ADDALY · THE AI CLASSROOM

Motion, Sound and Music

Why nothing in the world starts at full speed, and why a blanket beats a better microphone.

---

8 modules · 72 lessons · 25 figures · 8 case studies · 69,137 words · about 12 hours

Dr Mustafa Mun
Author and editor

ABOUT THIS BOOK

Motion, Sound and Music, published by Addaly, 2026. Author and editor: **Dr Mustafa Mun**.

The manuscript was drafted with the assistance of a large language model and was edited and published under his name. That is stated plainly here rather than left to be discovered, because a reader is entitled to know how a book they are trusting was made.

This book is the printed form of the course of the same name at addaly.com/learn/motion-and-audio. The website is the living version and is corrected first; where the two differ, the website is right.

Free to read, free to download, free to print, and free to teach from. There is no paid edition and there never will be. If you paid for this, somebody has sold you something that was given away.

Every diagram in it is drawn from data by code, not generated as a picture, so the numbers on the page are the numbers in the argument. Answers to every exercise, test and examination question are at the back, with a note on why each wrong option attracts people — those notes are the teaching, not the marking.

Contents

2. Motion that carries meaning

Specify the motion in an interface as something a second developer could build from a written brief — naming which element moves, for how many milliseconds, on which curve, in what order relative to its neighbours, what it does when the viewer has asked for reduced motion, and which CSS properties the browser can animate without re-running layout — and justify each number from the perceptual or rendering limit it comes from

1. The three jobs an animation can do
2. How long an animation should take
3. The transition is the explanation
4. Staggering without making people wait
5. Why your lower-third looks like a template
6. Naming the motion so it stops drifting
7. Spinners, skeletons and honest waiting
8. Motion that makes people ill
9. Why transform is cheap and width is not
10. Case study: Three hundred milliseconds, measured on the phone the invigilators actually have
11. Module test

3. Shipping the motion

Choose a delivery format for a given piece of motion by reasoning from whether it must respond to state, whether its content is vector or photographic, and who will edit it next — then produce that file with the right encoder settings, verify it on the target platform rather than in the authoring tool, and hand it over with a written specification covering trigger, interrupt, exit, responsive behaviour and reduced motion

1. Five ways to put motion on a screen
2. What Lottie carries and what it drops
3. Animating a drawing that is also a document
4. Why a GIF costs twenty times what a video costs
5. Transparent video, and why it is two files
6. Frames in a grid, and the memory they cost
7. Motion that responds to state
8. Send the developer a file, not a video
9. The specification a developer can build from
10. Case study: The loader that arrived as a video
11. Module test

4. Putting two images together

Combine two pieces of footage that were never filmed together so the seam does not show — building the matte by the cheapest route the shot allows, diagnosing an edge fringe from its colour, repairing spill and edge interaction rather than the matte's interior, attaching the insert with a track or solve whose error you have read, and matching black level, grain, focus and lens distortion in a defensible order

1. The greyscale image that decides everything
2. The dark fringe and what causes it
3. Every blend mode is one line of arithmetic
4. A green screen is fixed on set, not in the edit
5. Spill is real light, and light wrap is an honest fake
6. Drawing the matte by hand
7. What a tracker can and cannot see
8. Solving for a camera, and when you cannot
9. Black level, grain, focus, and the order to do them in
10. Case study: Forty minutes of lecture on a phone-shot green screen
11. Module test

5. Codecs, colour and delivery

Export a piece of video against a stated constraint — a size budget, a platform's requirements, an archival master, an editable intermediate — by choosing the codec class, rate-control mode, bit depth and colour tags from what the file is for; and diagnose a delivered file that looks milky, crushed, banded, juddery or twenty times too large by naming which of those decisions produced it

1. What a codec throws away
2. Frames that stand alone, and frames that do not
3. Quality target or size target
4. Why your export looks washed out
5. Counting the steps in a gradient
6. Log footage and what a LUT is not
7. Frame rates, shutter and judder
8. Your hero video is why the page feels slow
9. The tool under most of the others
10. Case study: Two hundred lessons, rejected for milky blacks
11. Module test

6. Capturing sound

Record a usable voice in an ordinary room — choosing pattern and distance from what the room is doing rather than from a specification sheet, setting gain so peaks land with headroom you can justify, capturing room tone as a matter of routine, and bringing a second microphone or a second device in without comb filtering or drift — and say for any given capture fault whether it can be repaired afterwards or not

1. Pressure, gradient, and everything that follows
2. People forgive a soft picture and leave over bad sound
3. Direct sound, reverberant sound, and the distance between them
4. Where the level should sit
5. Two numbers doing two different jobs
6. What the noise floor costs you later
7. Two microphones and a comb filter
8. Latency, and why a speaker starts stumbling
9. Two devices, one event
10. Case study: A twelve-thousand-rupee grant and a tiled kitchen
11. Module test

7. Shaping and mixing sound

Take a raw voice recording to a delivered mix — filtering and cutting before boosting, ordering the chain so each stage is not re-creating work for the next, placing the voice in a space without losing intelligibility, sitting music under it by arrangement rather than by ducking alone, and hitting a named integrated loudness and true-peak ceiling with a gain change rather than a limiter — and predict what the mix will do on a phone speaker before hearing it there

1. Three EQ moves that cover most voices
2. Compression does not make anything louder
3. Wind, hiss, and why both are placement problems
4. Reverb is information about where
5. Music under speech
6. Three boring sounds beat any filter
7. The number every platform measures
8. What survives a phone speaker
9. The chain, and why the order is not arbitrary
10. Case study: The music bed that sounded right in the office
11. Module test

8. Machines that make sound and motion

Use a generative or analytic model inside a motion and sound pipeline — synthesised speech, a cloned voice, automatic captions, source separation, generated music, interpolated or upscaled frames — and state for each what it is estimating, which characteristic failure to look for and where, what consent or licence the use requires, and which parts of the surrounding law are unsettled rather than merely complicated

1. What a text-to-speech system cannot read
2. The vector that is supposed to be you
3. A cloned voice needs a yes you can point to
4. Five per cent, and which five per cent
5. Taking a mixture apart
6. Generated music, and three separate legal questions
7. What a music licence actually covers
8. Frames that were never photographed
9. Provenance, watermarks and what to tell people
10. Case study: Forty more lessons in a voice that had left
11. Module test

Motion, Sound and Music — final examination

The paper that opens the next course. Answers to everything follow it.

MODULE 1

How motion reads as real

Before any software, the part that decides whether animation looks expensive or amateur: what a frame rate actually commits you to, how to read a speed graph, why the default ease is a compromise nobody chose, what a spring buys and what it costs, and which of the old animation principles survive on a screen somebody looks at four hundred times.

> BY THE END OF THIS MODULE YOU CAN
>
> Diagnose a piece of flat or floaty animation by reading its timing and spacing — naming whether the fault is the easing curve's influence, the spacing between frames, a missing arc, or an absent anticipation — and correct it by editing the value graph rather than adding an effect on top

---

1 · 9 min

Nothing in the world starts at full speed

THE ONE THING TO KEEP

Ease-out for anything arriving or responding, ease-in for anything leaving, and linear only for motion that never starts or stops.

The thing that makes homemade animation look homemade

Someone animates a logo sliding in from the left. The logo is fine, the typography is fine, and the whole thing looks cheap. It is almost never the design. It is

that the logo travelled at exactly the same speed for every frame of the move, and nothing in the physical world does that.

Fix the speed curve and the same animation, unchanged in every other respect, stops looking amateur. This is the highest-return single idea in motion, so it is worth understanding rather than just switching on.

What a keyframe actually stores

A keyframe is a value at a moment. "At 0 seconds, x = -400. At 0.4 seconds, x = 0." That is two facts. Everything between them is invented by the software, and the rule it uses to invent them is called interpolation.

The default rule in almost every tool is linear: divide the distance evenly across the frames. At 24 frames per second, a 0.4-second move is about ten frames, and linear puts the logo 40 pixels further along on every single one. Constant speed. It starts instantly at full pace and stops dead.

Easing means reshaping that curve so the value changes slowly at one end and quickly at the other. The object accelerates and decelerates, which is what things with mass do.

The three curves worth knowing

- **Ease-out** — fast at the start, slowing into the finish. Use it for anything arriving, and for anything responding to a click or tap. It moves immediately, so it feels responsive, and it settles gently, so it feels controlled. If you learn one curve, learn this one.
- **Ease-in** — slow at the start, fast at the finish. Use it for things leaving. It gathers pace and is gone.
- **Ease-in-out** — slow at both ends. Use it for something crossing the screen from one visible position to another, where the viewer watches the whole journey.

Where to find them:

- **After Effects**: select the keyframes and press F9 for Easy Ease, then open the Graph Editor (the graph icon in the timeline) and drag the bezier han-

dles. Pulling the outgoing handle far to the right makes a long, slow tail. Nothing teaches easing faster than watching that curve while the preview loops.

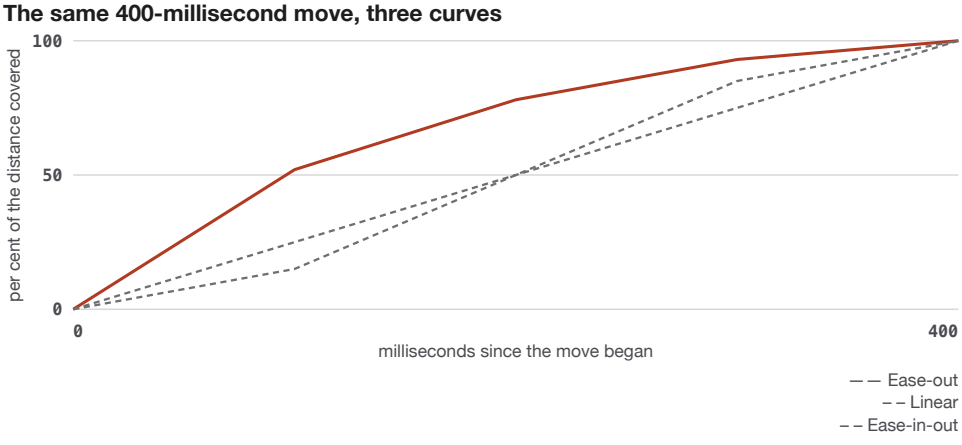
- **DaVinci Resolve** (free): right-click a keyframe in the Inspector for ease options; the Fusion page has a proper spline editor with the same handles.
- **Blender** (free): the Graph Editor, with Bezier interpolation and handles you drag.
- **CSS**: `transition: transform 240ms cubic-bezier(0.2, 0, 0, 1);`. That particular curve is a strong ease-out and is a good default for interface motion.
- **Canva** and **CapCut** give you named presets rather than curves. Less control, and for a title card that is often enough.

Duration, and why distance is not proportional

Rough starting points, then adjust by eye:

- Small interface changes — a toggle, a hover state, a colour change: 120–200 ms.
- A panel, a sheet, a modal appearing: 250–400 ms.
- A logo sting or a title card: 600 ms to a second, because you want it noticed.

Anything the user triggers repeatedly should sit at the short end. A 500 ms animation is delightful once and a tax on the fortieth use.



Same start, same end, same duration. At the halfway moment the linear move is halfway across, the ease-out is roughly three-quarters of the way, and the ease-in-out is also halfway but has spent its first hundred milliseconds barely moving — which is what a user reads as lag after they have clicked. Step your own animation to its middle frame and read off which of the three you actually made.

Distance matters, but not proportionally. Double the travel and you want roughly 1.4 times the duration, not double. A thing crossing twice the distance in the same time simply moves faster, which is what your eye expects of something further away.

Where this goes wrong

Ease-in-out on everything. It is the pleasant-sounding default, and on anything the user initiates it is wrong. A dropdown with a 400 ms ease-in-out spends its first hundred milliseconds barely moving. The user has clicked and seen nothing. They register that as lag, not as elegance. Use ease-out so it starts the instant they act.

Easing something too short to see it. Below about 100 ms there are not enough frames for a curve to read, and every curve looks the same. If you want a snappy feel at 80 ms, just make it linear and stop fiddling.

Assuming linear is always wrong. It is right whenever there is no start and no stop: a loading spinner rotating forever, a scrolling ticker, a looping background. Easing those makes them lurch.

Easing the wrong property. A fade has no mass. Opacity going from 0 to 1 with a heavy ease-in-out reads as hesitant rather than weighty. Ease move-

ment and scale; keep opacity simple.

Check your own work today

Screen-record any animation you have made and step through it a frame at a time. Find the frame at the exact middle of the move. If the object is halfway across, the motion is linear, whatever the software told you. If it is roughly three-quarters of the way, you have a real ease-out, and it will already look better than it did.

BEFORE YOU MOVE ON

A designer gives a dropdown menu a 400 ms ease-in-out on click. Testers say the interface feels sluggish, even though a 400 ms panel slide elsewhere in the same app feels fine. What is the most likely cause?

A. 400 ms is too long for any interface animation, and every transition in the app should be brought under 200 ms.

B. The menu is animating its height and position rather than a transform, so the browser is dropping frames.

C. Ease-in-out starts slowly, so almost nothing visibly moves for the first hundred milliseconds after the click; that gap before any feedback is what reads as lag. Ease-out would move immediately.

D. The curve is too gentle to signal that the click registered, and a small overshoot at the end would make it feel snappier.

2 · 9 min

Motion blur will not fix a floaty logo

THE ONE THING TO KEEP

Weight is uneven distance-per-frame plus a small anticipation and settle; effects added afterwards cannot supply it.

Motion blur will not fix a floaty logo

The usual sequence: an animation feels weightless, so the animator switches on motion blur, then lengthens the duration, then adds a glow. It still feels weightless, because none of those things is what creates weight.

Weight comes from spacing — how much distance the object covers on each individual frame. The classic animation principles were worked out for pencil drawings decades before computers, and they transfer directly to a logo sting or an interface transition. They are not about cartoons. They are about how eyes read movement.

Timing and spacing are two different things

Timing is how many frames the move takes. Spacing is how the distance is distributed across those frames. Easing, from the last lesson, is spacing seen from the curve's side.

A heavy object covers little ground on the first frames and a lot in the middle. A light object snaps. A thing landing covers most of its distance early and then takes a long, quiet settle. If you draw the object's position on each frame down a strip of paper, weight is visible as the gaps between the marks. Uneven gaps look alive. Even gaps look mechanical, no matter how long the animation runs.

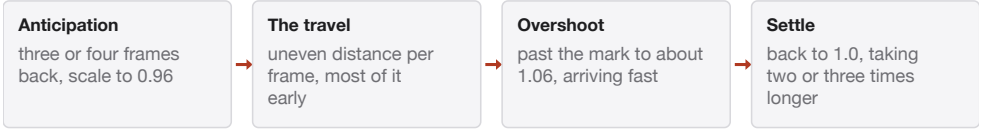
The three moves that create weight

Anticipation

Before a thing moves forward, it moves back. A person about to jump crouches. A bat swings back before it swings through.

On a logo, this is tiny and lasts three or four frames: scale down to 0.96, then up past 1.0. On a button press, the button dips 2 pixels before the panel opens. On a card sliding in from the right, it slides 8 pixels further right first.

The four parts of a logo move that has weight



None of these is an effect. Weight is the distribution of distance across frames plus a wind-up before and a settle after, and a piece that has none of them stays weightless however much blur and glow are added. Motion blur has one real job — stopping fast motion strobing — and supplying mass is not it.

The reason it works is that it tells the eye where to look before the movement happens. Without it, motion arrives unannounced and the viewer spends the first frames finding it.

Overshoot and settle

Things with momentum do not stop exactly on their mark. They go past and come back.

A workable logo scale: 1.0 → 1.06 → 0.99 → 1.0, with the overshoot arriving fast and the settle taking two or three times as long as the overshoot did. For interface elements keep it far smaller — 1 or 2 per cent — or skip it entirely.

That last point is the honest limitation. Overshoot is decoration. On a checkbox someone ticks forty times a day it is charming once and irritating thereafter, and on text it is nearly always bad, because a bouncing glyph is measurably harder to read. Save springiness for once-per-session moments: a launch screen, a payment succeeding, a celebration.

Follow-through and overlapping action

Parts do not all stop at the same instant. A tagline under a logo should start moving three to five frames after the logo does, and arrive after it. A list of items should stagger by 30–50 ms each, not 200 — beyond about 60 ms the list stops reading as one gesture and starts reading as a queue.

- **After Effects**: select layers and use **Animation → Keyframe Assistant → Sequence Layers**, or offset the layers in the timeline by hand.
- **CSS**: `animation-delay: calc(var(--i) * 40ms)` with an index set per item.
- **Resolve Fusion** (free): offset the keyframes in the Spline editor.

A useful test: if you switch off every element except one, does that one element's motion still make sense on its own? If the whole thing only works as a synchronised block, it will feel stiff.

What motion blur is actually for

Motion blur is a consequence of a camera shutter being open while something moves. It does two real jobs: it stops fast motion from strobing, and it matches synthetic motion to live footage.

At 24 or 30 frames per second, an object crossing a large part of the frame in two or three frames will strobe — the eye sees separate copies rather than a movement. Blur solves that. It does not create mass, and adding it to a floaty animation gives you a floaty animation that is now slightly soft.

In After Effects, motion blur has to be enabled twice: on the layer and on the composition. Resolve's Fusion page has a MotionBlur node and per-transform blur settings. Blender renders it from the shutter setting. CSS has no motion blur, which is fine, because web motion is usually short enough not to need it.

One exercise

Take any twelve-frame move you have already made. Do not add a single effect. Move the middle keyframe so that the object is three-quarters of the way

across at frame six instead of half. Add a four-frame anticipation at the start. Watch both versions back to back.

That is the entire difference between motion that reads as designed and motion that reads as a slide transition.

BEFORE YOU MOVE ON

A client says a logo animation feels floaty and weightless. The animator switches on motion blur and extends the duration from 0.6 to 1.2 seconds. It still feels weightless. What is the actual fix?

A. Redistribute the distance across the frames — cover most of the travel in the first few frames and give it a long settle — and add a few frames of anticipation before the move.

B. Extend the duration further, since a heavier object should take longer to complete the same move.

C. The motion blur was applied at the wrong shutter angle, so it is not reading as real camera blur.

D. Add a soft drop shadow and a slight perspective tilt so the logo reads as a physical object in space.

3 · 9 min

Frames, not seconds

THE ONE THING TO KEEP

Frame rate is a commitment made before the first keyframe, because a duration in milliseconds only becomes a real animation once it has been rounded to whole frames — and changing the rate afterwards silently resamples everything you spaced by hand.

The software lies to you politely

Every animation tool shows a timeline in seconds, and almost everybody types in seconds. Underneath, nothing is stored in seconds. A frame is the unit, and a second is whatever number of frames you decided on before you started.

This matters the first time you specify a 300ms transition and hand it to somebody working at a different frame rate.

- At 24 fps, 300ms is 7.2 frames. You get 7, and the real duration is 292ms.
- At 25 fps, it is 7.5 frames. You get 7 or 8 depending on which way the software rounds.
- At 60 fps, it is 18 frames exactly.

None of these are wrong. They are just not the same animation, and if you approved one and shipped another, the difference is real.

The rates, and what each one is for

- **24 fps** — cinema, and anything that wants to read as film. The lowest rate at which camera motion is tolerable.
- **25 fps** — broadcast in India, the UK, Europe, Australia. If your work touches television anywhere on 50 Hz mains, this is the number.
- **30 fps (really 29.976)** — North American broadcast heritage, and the safest default for social video because most platforms re-encode towards it

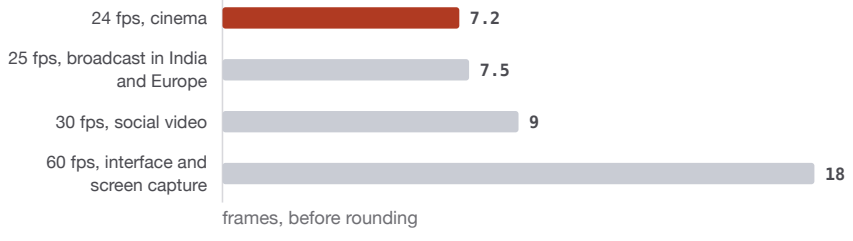
anyway.

- **50 and 60 fps** — sport, screen recordings, and interface work where you want the motion to feel like the device rather than like a film.
- **12 fps** — not a delivery rate, a working method. Hand-drawn animation holds each drawing for two frames inside a 24 fps timeline. Twelve distinct images a second is enough for drawn motion and nowhere near enough for a camera pan, which is why traditional animation works "on twos" and live action cannot.

29.97, and where the missing time went

When colour was added to American black-and-white television, the frame rate was slowed by exactly one part in a thousand to keep the colour subcarrier out of the audio. That 0.1% never went away. So 30 fps is usually 29.97, and 24 fps in a broadcast chain is usually 23.976.

What 300 milliseconds is, in frames



A frame is the unit and a second is however many frames you chose before you started. Each of these rounds to a whole frame, so the duration you actually get is 292 ms at 24, either 280 or 320 at 25 depending on which way the software rounds, and exactly 300 at 30 and 60. None of them is wrong and none of them is the same animation, which is why a brief says seven frames at 24 rather than about a third of a second.

The consequence is that a programme that runs for ten minutes of real time does not run for ten minutes of timecode. Over an hour, the gap is about 3.6 seconds. Drop-frame timecode fixes the *labels* by skipping two frame numbers every minute except every tenth minute — it does not drop any picture. Non-drop timecode counts honestly and drifts from the clock.

If you have ever wondered why an editor's delivery spec says `29.97 DF` and refuses anything else, that is why: a broadcast slot is measured by the clock, not by the frame count.

Set it first, and do not change it after

Pick the delivery frame rate before you place a single keyframe. Changing it later is not a neutral operation, and different tools disagree about what it means:

- After Effects keeps keyframes at their *time* values, so they land on fractional frames and get sampled to whichever frame is nearest. A carefully spaced move loses its spacing.
- Blender's frame-rate change keeps the *frame numbers*, so the animation changes duration instead.
- Kdenlive will not let you change a project's profile frame rate after the fact without re-interpreting every clip.

None of these is a bug. They are two reasonable answers to a question you should never have to ask.

Higher is not better, it is different

A common assumption is that 60 fps is a quality upgrade over 24. It is a different look, and the mechanism is exposure, not resolution.

With a conventional 180-degree shutter, the camera is open for half the frame interval. At 24 fps that is 1/48 of a second of blur per frame. At 60 fps it is 1/120 — half as much. Less blur per frame means each frame is crisper and the strobing between them is more visible, which is exactly the "looks like a soap opera" reaction people report and cannot explain. Rendered animation has the same property: if you do not add motion blur in the render, a fast move at any frame rate will strobe.

So the honest statement is: 60 fps buys you temporal resolution and costs you the blur that hides the gaps. For interface motion, where elements are flat and travel short distances, that trade is usually good. For a camera move across a landscape, it usually is not.

What to write down

On every brief, before anything else: **frame rate, resolution, and duration in frames.** "Seven frames at 24" is a specification. "About a third of a second" is a conversation you will have again.

BEFORE YOU MOVE ON

The same 300ms slide is built twice: once in a 24 fps composition and once at 60 fps, both travelling 600 pixels. Played back, the 24 fps version strobes during the fastest part of the move while the 60 fps version does not. What is the mechanism?

A. At 24 fps the object covers roughly 83 pixels between consecutive frames instead of 33, so the gaps between its positions become visible unless motion blur fills them.

B. The 24 fps file has a lower bitrate, so the encoder is discarding detail during fast motion.

C. 24 fps rounds 300ms down to 7 frames, and the shorter duration is what makes it look rougher.

D. The easing curve is evaluated once per frame, so a composition with fewer frames gives the software fewer opportunities to apply the ease, and the motion degrades towards linear as the frame rate drops.

4 · 9 min

The panel most people never open

THE ONE THING TO KEEP

The speed graph is the derivative of the value graph, so the shape of one tells you what the motion is doing between keyframes — and a rectangular speed graph is the signature of the linear default that makes animation read as wrong.

Two graphs, and they answer different questions

Every animation tool has a curve editor, and most people animating in one have never opened it. It is the single highest-leverage panel in the application, because it is the only place the software tells you what it is actually doing between your keyframes.

There are two views of the same data.

- **The value graph.** Vertical axis is the property's value — pixels, degrees, percent. Horizontal axis is time. The *slope* of this line is the speed.
- **The speed graph.** Vertical axis is velocity — pixels per second. Horizontal axis is time. A flat line here means constant speed.

They are not alternatives; one is the derivative of the other. Blender calls the first an F-curve and gives you handles on it. After Effects gives you the value graph for one-dimensional properties like opacity and rotation, and the speed graph for position, because position has two or three dimensions and three overlapping value curves are unreadable.

What a default keyframe looks like

Drop two keyframes on position, 600 pixels apart, twenty frames apart, and leave them linear. The value graph is a straight diagonal. The speed graph is a

rectangle: velocity jumps from zero to 720 px/s at the first keyframe, holds, and drops to zero at the second.

Those two vertical edges are the problem. Nothing in the physical world accelerates from rest to 720 px/s in zero time and then stops dead. Your eye has never seen it and reads it as wrong before you have finished watching.

The arithmetic is worth doing once by hand. Twenty frames, 600 pixels, linear: exactly 30 pixels of travel every frame, from the first to the last. Now ease the second keyframe out: the same journey might be 90 pixels on the first frame and 2 pixels on the last. Same distance, same duration, completely different reading. That difference lives entirely in the graph.

How to read a bad animation

Open the speed graph and look at the shape.

- **A flat top.** The move has a cruise — it reaches a speed and holds it. Fine for a conveyor belt, wrong for anything that started because somebody pressed a button.
- **A sharp spike.** All the distance is covered in a couple of frames. Reads as a snap. Often exactly what you want for a small UI element and never what you want for a large one.
- **A hump that peaks late.** Slow start, fast finish. This is an ease-in, and it reads as something falling or accelerating away. If you applied it to an arriving element, that is your bug.
- **Two humps.** Something between your keyframes is being overshot and pulled back. Usually an accidental spatial bezier.
- **The curve dipping below zero.** The property is briefly moving backwards. On position this is a bezier handle overshooting; on scale it is a value going negative and flipping the layer.

Spatial and temporal are separate

This catches everybody once. In After Effects, a position keyframe has *two* independent interpolations: how the value moves through space (the path on the canvas) and how it moves through time (the speed graph). You can have a

perfectly eased speed graph on a path that loops in a curve you never asked for, because the spatial interpolation defaulted to bezier and the software auto-smoothed a corner.

If a layer travels in an arc you did not draw, the fix is in the composition viewer — select the keyframes and set spatial interpolation to linear — not in the graph editor.

The free path

You do not need a paid tool to get a proper curve editor.

- **Blender** has the best free graph editor of any of them. Every animatable property becomes an F-curve; handles are fully editable; there are built-in modifiers for cycles and noise. It animates 2D shapes, text and grease pencil, not only 3D.
- **Glaxnimate** is a small, fast vector animator with per-keyframe easing, and it exports Lottie.
- **Synfig** gives each waypoint an interpolation type — TCB, ease, linear, constant — rather than free handles, which is more restrictive but readable.
- **Kdenlive** exposes keyframe curves on effects, with a limited set of easing options. Enough for a title, not enough for character work.

Name After Effects in a portfolio because job listings name it. Learn the ideas in Blender if that is what you have, because the ideas are identical and the F-curve editor is more honest about what it stores.

The limitation

A graph editor is a measuring instrument. It shows you what is happening and never tells you what should happen. Two animators looking at the same curve will disagree about whether it is right, and both can be correct for different pieces. What it removes is the category of problem where you keep adding effects because you cannot see that the underlying motion is a rectangle.

The habit worth building: when something feels off, open the graph *before* you add anything.

BEFORE YOU MOVE ON

A layer eases in and out correctly according to its speed graph, but on the canvas it travels in a gentle curve rather than the straight line intended. What is going on?

A. The ease has been applied to too many frames, so the software is compensating by curving the path to preserve the total distance.

B. The speed graph is a fundamentally different kind of animation from the value graph, and choosing it commits the layer to curved spatial motion that can only be straightened by switching back to value curves.

C. The spatial interpolation is bezier while the temporal interpolation is eased; they are independent settings, and the path has been auto-smoothed in the viewer.

D. The anchor point is offset from the layer centre, so rotation is being introduced into the position channel.

5 · 10 min

Why pressing Easy Ease still looks wrong

THE ONE THING TO KEEP

A keyframe handle has an angle and a length, and the one-key ease sets both to a symmetric default nobody chose — dragging the outgoing handle's influence to roughly three-quarters is what turns a mushy move into one that commits and then settles.

The handle has two properties and you only used one

Select two keyframes, press the ease key, and the motion improves. It also stops improving there, and most people never find out why.

A bezier handle on a keyframe has two independent properties:

- **Angle** — the direction the curve leaves or arrives at the keyframe. Setting it flat is what makes the speed zero at that point.
- **Influence, or length** — how far into the interval that flatness persists before the curve turns.

Pressing the ease shortcut sets both, to a symmetric default. In After Effects that default is 33.33% influence on each side. In CSS, `ease` is `cubic-bezier(0.25, 0.1, 0.25, 1)`. Neither number was chosen for your shot. They are a compromise that makes everything a little better and nothing actually good.

The asymmetry rule

Most motion that reads well is asymmetric, and the reason is physical. A thing arriving has to shed energy, which takes time. A thing departing under its own power can accelerate hard and then leave the frame at speed — the deceleration happens off-screen or not at all.

So:

- **Something arriving or settling**: little or no ease on the way in, a long ease on the way out. Try 10% in, 75% out.
- **Something leaving**: a long ease in, little on the way out, because you never see it stop.
- **Something moving between two positions and staying**: ease both, but not equally — around 30% in, 60% out reads as intentional rather than symmetrical.

The symmetric default is the one shape that never happens in the world: an object that decelerates exactly as gradually as it accelerated, having apparently decided both in advance.

Why symmetric easing feels mushy

The distance is fixed and the duration is fixed. If both ends are slow, all the travel has to be crammed into the middle. You get slow, then a burst, then slow. The eye reads that centre burst as the real motion and the slow ends as hesitation, so the whole move feels indecisive without anybody being able to say why.

Move the influence to one side and the picture changes: the object commits, then relaxes. Same two keyframes, same duration, different reading.

The numbers people actually ship

If you want a starting point rather than a principle:

```
/* browser defaults */
ease:        cubic-bezier(0.25, 0.10, 0.25, 1.00);
ease-out:    cubic-bezier(0.00, 0.00, 0.58, 1.00);
ease-in-out: cubic-bezier(0.42, 0.00, 0.58, 1.00);

/* the curve most design systems converge on for UI */
standard:    cubic-bezier(0.40, 0.00, 0.20, 1.00);
decelerate:  cubic-bezier(0.00, 0.00, 0.20, 1.00);
accelerate:  cubic-bezier(0.40, 0.00, 1.00, 1.00);
```

The four numbers are the x and y of the two control points on a unit square: time across, progress up. `0.40, 0.00` says the curve leaves the origin flat and slightly late — a short, firm acceleration. `0.20, 1.00` says it arrives at full progress early and coasts — a long settle.

Notice that the standard curve is exactly the asymmetry described above, expressed as four decimals.

Overshoot, and how much

An object that arrives and stops dead at its target looks correct and dull. An object that passes the target and comes back reads as having mass.

With bezier keyframes this needs a third keyframe: target at 105–110%, then back to 100%. The return should take roughly twice as long as the overshoot, because settling is slower than travelling.

- **103–106%** — barely conscious. Right for anything you will see hundreds of times.
- **108–115%** — noticeable, playful, fine for a one-off title or a logo sting.
- **125% and up** — cartoon. Correct for cartoons.

The common failure is overshooting on scale and position together, which produces something that looks like it has been thrown at the screen.

Where beziers run out

A bezier curve between two keyframes is a single smooth arc. It cannot express a bounce that decays over three or four diminishing hops without you stacking keyframes and setting each one by hand. It also cannot be interrupted: if the user taps again halfway through, a keyframed tween has to be cancelled and restarted, and the restart is visible as a jump.

Those two limits — decay, and interruptibility — are the whole argument for springs, which is the next lesson. Beziers are for motion you fully control from start to finish. Springs are for motion that has to respond to somebody.

The practical move

Next time an animation is nearly right, do not add a blur or a glow. Open the graph, grab the outgoing handle, and drag it from a third of the way across to about three-quarters. Watch what happens to the settle. That one drag is most of the difference between animation that looks made and animation that looks defaulted.

BEFORE YOU MOVE ON

A designer applies symmetric easing to a card sliding 400px into place over 400ms. It reads as hesitant. Why does making both ends equally gradual produce that specific impression?

A. Symmetric easing halves the effective duration of the move, so the card spends only around 200ms of its 400ms actually travelling, and the eye registers the two stationary-looking stretches at either end as pauses.

B. The distance and duration are fixed, so slowing both ends forces all the travel into a fast middle — the eye reads the burst as the motion and the slow ends as indecision.

C. Symmetric curves have no overshoot, and without overshoot no motion can read as having mass.

D. Cubic beziers cannot hold a constant velocity, so a symmetric curve is always accelerating or decelerating and never settles.

6 · 10 min

Springs, and what you give up for them

THE ONE THING TO KEEP

A spring's duration is an output of stiffness, damping and mass rather than something you set, which is why it can be redirected mid-flight without a visible restart — and why it is the wrong tool for anything that must land on a beat or must never exceed its target value.

Duration becomes an output

A bezier tween is defined by where it starts, where it ends, and how long it takes. A spring is defined by physics, and the duration falls out of the physics. You do not set it.

Three numbers control a spring simulation:

- **Stiffness** — how hard it pulls towards the target. Higher is faster and snappier.
- **Damping** — how much energy is removed each step. Higher settles sooner with less wobble.
- **Mass** — inertia. Higher is slower to start and slower to stop.

The relationship that matters is the **damping ratio**: damping divided by twice the square root of stiffness times mass.

- Ratio **below 1** — underdamped. It overshoots and oscillates. This is what most people want and what "springy" means.
- Ratio **exactly 1** — critically damped. It arrives as fast as possible with no overshoot at all.
- Ratio **above 1** — overdamped. It creeps in and never overshoots. Usually feels sluggish.

A useful starting set for interface work: stiffness 200, damping 20, mass 1. That gives a ratio of about 0.7, a small single overshoot, and a settle in roughly half a second. Drop damping to 12 and it wobbles three times. Raise it to 28 and the overshoot disappears.

The real reason frameworks use springs

It is not that springs look nicer. It is that they can be interrupted.

Consider a sheet being dragged up, released halfway, and grabbed again before it finishes. A keyframed tween has a fixed start value, a fixed end value, and a clock. To redirect it you must cancel it and start a new one, and the new one starts from zero velocity — so the sheet momentarily stops dead in the user's hand. Everybody has felt this and blamed the phone.

A spring carries velocity as state. Change the target mid-flight and the simulation continues from wherever it is, at whatever speed it had. The motion stays continuous because it never restarted. This is why SwiftUI, Framer Motion, React Spring and the Android motion libraries all default to springs for gesture-driven work.

Where a spring is the wrong choice

Overshoot means going past the value. Sometimes the value means something.

- **Progress indicators.** A bar that springs to 100% visibly passes 100%. You have just told the user the download is 107% done.
- **Anything reading a real quantity** — a gauge, a counter, a temperature. Overshoot is a false reading held for 200ms.
- **Motion that must land on a beat or a word.** You cannot specify a spring's duration, so you cannot make it finish at frame 74. For sound sync, use keyframes.
- **Anything rendered to a video file for a fixed-length edit.** Same reason. Bake the spring to keyframes first if you need both.

There is also a subtler one: springs make everything feel like the same product. When every element in an interface uses the house spring, motion stops carrying information because nothing is distinguishable from anything else.

Doing it without a framework

CSS gained the `linear()` easing function, which takes a list of sampled progress values. A spring is just a function of time, so you can sample it and paste the result:

```
.sheet {
  transition: transform 0.6s linear(
    0, 0.14, 0.42, 0.72, 0.94, 1.06, 1.09, 1.06, 1.02, 1, 1
  );
}
```

Those eleven numbers are a spring evaluated at eleven points in time. You can see the overshoot in the data: it passes 1 at the sixth sample, peaks at 1.09, and comes back. Any spring calculator will generate the list, and the result runs on the compositor with no JavaScript.

In Blender, add a **Damped Track** or simply bake the physics: animate the target, run the simulation, then convert to keyframes so the curve becomes editable. Glaxnimate has no spring, so you approximate with three keyframes — target, overshoot, settle — which is the bezier method from the previous lesson.

The honest limitation

A spring simulation is deterministic but not predictable in the way a designer needs. "How long does it take?" has no clean answer, because mathematically it never fully arrives — implementations stop when the displacement and velocity both fall under a threshold. Two libraries with the same stiffness and damping can settle at visibly different times because they chose different thresholds and different integration step sizes.

So if you specify "stiffness 200, damping 20" to a developer using a different library than the one you prototyped in, you have specified less than you think. The thing to hand over is the *feel plus the numbers plus a reference recording*, and a willingness to tune once it is running on the real device.

BEFORE YOU MOVE ON

A bottom sheet animated with a keyframed tween feels like it 'sticks' when the user grabs it again mid-animation, while the spring version does not. What is the mechanism behind the difference?

A. Spring animations are handed to the GPU compositor while keyframed tweens are evaluated on the main thread, so the tween is dropping frames at exactly the moment a touch event arrives and needs to be processed.

B. The spring uses a shorter duration, so there is less time for an interruption to land in the middle of it.

C. A tween is defined by fixed endpoints and a clock, so redirecting it means cancelling and restarting from zero velocity, while a spring carries velocity as state and continues from it.

D. Tweens cannot overshoot, so the sheet has to stop exactly at the boundary before it can begin moving the other way.

7 · 9 min

Straight lines and the things that lag behind

THE ONE THING TO KEEP

Arcs and small two-to-four-frame offsets between parts are what separate animated motion from interpolated motion, but the element under the user's finger must never lag, because perceived responsiveness is measured from input to first visible change.

Almost nothing travels in a straight line

Default interpolation between two positions is a straight line, because a straight line is the simplest thing a computer can do. It is also close to the rarest path in the physical world.

The reason is joints. A hand moving across a desk pivots at the elbow and the shoulder, so its path is a shallow arc. A head turning to look at something sweeps. A thrown object follows a parabola. Even a car changing lanes traces an S, because steering is a rate of change of direction, not a direction.

So when animation reads as mechanical and the easing is already correct, the next thing to check is the path.

Making an arc

In most tools the position path is editable directly on the canvas, and adding an arc means dragging the spatial handle at the midpoint. A useful amount is small: for a 600-pixel horizontal move, a 40 to 80 pixel vertical bow is usually enough to remove the mechanical reading without anybody consciously noticing a curve.

Which way the arc bends carries meaning. Bowing upward reads as light — a thing lifted and set down. Bowing downward reads as heavy, or as something falling into place. Pick deliberately; the default is neither.

For interface motion the rule is narrower. Material Design's guidance is that elements moving in two dimensions arc, and elements moving in one dimension do not. A card that travels both across and down arcs. A drawer sliding horizontally should not, because a curve implies a freedom the drawer does not have.

Follow-through, overlap and drag

Three related ideas, often confused, and each is a small offset in time.

- **Overlap**: parts start at different times. The body starts, then the arm.
- **Follow-through**: parts keep going after the main mass stops. The body lands, the coat keeps travelling for a few frames.
- **Drag**: a trailing part lags behind the whole way, never leading.

The numbers are small. At 24 fps, an offset of **2 to 4 frames** is the working range for most secondary elements. Below 2 it is invisible. Above 6 it stops reading as attachment and starts reading as two separate animations.

A concrete rig you can build in ten minutes:

1. Parent a text layer to a null object.
2. Animate the null. This is your main mass.
3. Select the text layer's own keyframes and drag them three frames later.
4. Add a 4% scale overshoot to the text only, settling two frames after the null stops.

The null arrives, the text catches up and settles. Nobody watching will describe it as follow-through. They will say it looks more expensive.

Where this is actively wrong

Follow-through on the element the user just touched is a bug, not a flourish.

The mechanism: the perceived responsiveness of an interface is measured from the input to the *first* visible change. If you delay the response of the thing under the finger by three frames to make it feel weighty, you have added 125ms to the perceived latency of every tap. The research on this is old and

consistent — under about 100ms reads as instantaneous, and past roughly 300ms people start looking for a second thing to do.

So the split is: **the control responds immediately, the scenery follows through.** A button's own press state has no lag. The sheet it opens can absolutely have a trailing header.

Secondary action

Separate from follow-through, and often what is missing when a piece feels thin. Secondary action is a *different* motion happening because of the main one, not a delayed copy of it.

- A card lifting: the shadow spreads and softens as it rises. That is the shadow doing a different thing, driven by height.
- A logo landing: a small ripple in the surface, or a two-frame flash of a highlight.
- A list re-ordering: the items that did not move dim very slightly so the one that did is readable.

The test for whether a secondary action is earning its place: remove it and see whether anything becomes harder to understand. If not, it is decoration, which is allowed once and not four hundred times.

Doing it in free tools

Blender: parent to an Empty, then use the Graph Editor to offset the child's F-curves — select the keys and press G, X, and type the frame offset. It is exactly the four-step rig above.

Glaxnimate: group the layers and move the group's keyframes; per-layer offsets are done by dragging keyframes in the timeline.

Kdenlive: harder, because its keyframing is per-effect rather than per-layer. For anything with more than two moving parts, do the motion elsewhere and bring in a transparent video or an image sequence with alpha.

The habit

When animation feels flat, ask two questions in order. Is anything travelling in a perfectly straight line that should not be? Is every part starting and stopping on the same frame? Those two faults account for most of the gap between animation that was keyframed and animation that was animated.

BEFORE YOU MOVE ON

A designer adds a three-frame follow-through to a button's own press state at 24fps, reasoning that it gives the button weight. Users report the app feels slower. What has actually changed?

A. Three frames of extra animation per tap accumulates across a session, so total time in animation has increased measurably.

B. The overshoot contained in the follow-through means the button briefly renders larger than its pressed size, and users are interpreting that momentary size change as a rendering glitch rather than as intentional weight.

C. The follow-through is applied to the wrong layer, and the correct fix is to apply it to the button and remove it from the scenery.

D. Perceived responsiveness is measured from the input to the first visible change, so delaying the touched element's response adds about 125ms of apparent latency to every tap.

8 · 10 min

The old principles, filtered for a screen seen four hundred times

THE ONE THING TO KEEP

Film motion is watched once and interface motion is watched hundreds of times, so anticipation, exaggeration and squash-and-stretch — the principles that make a single viewing memorable — are precisely the ones that must be reduced or dropped on a screen somebody uses every day.

A different viewing contract

The twelve principles of animation were written at Disney for film. Film is watched once, in the dark, with attention. An interface is watched hundreds of times, on a bus, by somebody trying to do something else. A motion graphic on social sits in the middle: watched once by most people, forty times by whoever made it.

That difference is the filter. A principle designed to hold attention on a single viewing can be actively hostile at the four-hundredth.

Here is each one with a verdict.

Survives without modification

Slow in and slow out. This is easing. It is not a stylistic choice; it is the difference between motion and teleportation with a delay.

Timing. How long something takes is still the primary carrier of weight and meaning. Nothing has replaced it.

Arcs. Still true, still mostly ignored, still free.

Staging. Arguably more important on a screen than on film, because the viewer can look away and there is no director controlling the cut. If two things

move at once and neither is dominant, both are missed.

Solid drawing. Originally about drawing volume convincingly. It survives as: respect the geometry. If a card has a perspective, keep it consistent; if a shape has a light source, do not let it move independently of the shape.

Survives with a restriction

Follow-through and overlapping action. Yes, for secondary elements. Never for the control under the finger, for the reason in the previous lesson.

Secondary action. Yes, when it carries information. A shadow that spreads as a card rises tells you about height. A sparkle tells you nothing.

Appeal. Survives as brand fit. It is a real constraint and it is not a personal preference — a bank and a game want different motion, and the difference is measurable in duration and overshoot before it is anything else.

Straight-ahead and pose-to-pose. Pose-to-pose is what keyframing is; you set the extremes and let the machine fill in. Straight-ahead — drawing every frame in order — survives in frame-by-frame work, and is exactly what Blender's grease pencil and Krita's animation timeline are for. It produces motion no interpolation would ever generate, at a cost of hours.

Dangerous, and here is the mechanism

Anticipation. In film, a character winds up before a throw, and the wind-up is what makes the throw read. In an interface, anticipation means moving *away* from where the user asked you to go before going there. That is time added between the input and the answer, and the answer is what they wanted. Anticipation on a tap response is a bug.

Where it still works on a screen: in motion that nobody is waiting on. A logo sting can wind up. A page transition triggered by the system can. A button cannot.

Squash and stretch. Designed for bodies made of flesh, which deform. A card in an interface represents a rigid object, and stretching it reads as rubber rather than as speed. But the principle underneath — that a change of shape

communicates force — survives at small amplitudes. A 2 to 4% squash on a button press is felt and not seen; a 20% squash is a cartoon.

The other risk is that squash and stretch applied to text makes the letterforms wrong. Scaling type non-uniformly is a typographic error that happens to be animated.

Exaggeration. The principle most damaged by the viewing contract. Exaggeration is what makes a single viewing legible and memorable. It is also what makes the four-hundredth viewing unbearable. The practical rule: the more often a motion will be seen, the closer to literal it should be. A splash screen can be exaggerated; a keyboard cannot.

The test worth applying

Before shipping any interface motion, watch it twenty times in a row. Not twice — twenty. Almost every motion that is too long, too bouncy, or too clever fails this and nothing else catches it. The first viewing tells you whether it is charming. The twentieth tells you whether you can live with it.

For motion graphics destined to be watched once, invert the test: show it to somebody who has never seen it, once, and then ask them what it said. If they can tell you, exaggeration did its job.

What the principles do not cover

They were written before screens that talk back. Nothing in the twelve addresses interruption, gesture tracking, reduced-motion preferences, or a state machine with six states and twenty transitions between them. Those are genuinely new problems, and the honest position is that the field has not settled on answers for them the way it settled on easing. Treat the twelve as a good foundation with known gaps, not as a complete theory.

BEFORE YOU MOVE ON

Why is anticipation, which makes a character's throw read convincingly on film, treated as a defect when applied to a button's response to a tap?

A. Anticipation requires squash and stretch to read, and rigid interface elements cannot squash.
B. It inserts movement away from the requested result before the result, adding time between the user's input and the answer they asked for.
C. Interface animation durations are far too short for an anticipation move to become visible at all, so adding one is simply wasted effort on the animator's part rather than something that harms the experience.
D. Anticipation was designed for hand-drawn frames and does not survive interpolation between keyframes.

9 · 8 min

Where the eye already is

THE ONE THING TO KEEP

A viewer needs roughly 150–250ms to decide to look somewhere and another 20–40ms to get there, so a new element that appears far from where the last one left is missed for its first several frames — place the next point of interest where the eye already is, or hold a beat.

The viewer has one fovea and it is slow

Sharp vision covers about two degrees of your visual field — roughly a thumbnail at arm's length. Everything else is peripheral: good at detecting motion and contrast, bad at detail, useless for reading.

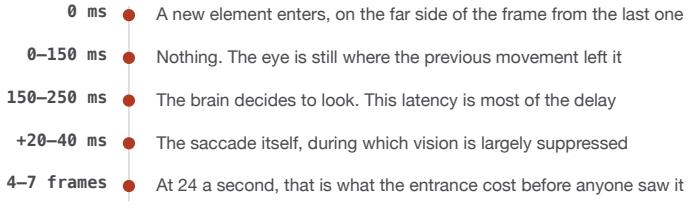
Moving the sharp part somewhere else is a saccade. A saccade takes 20 to 40 milliseconds to execute, and there is a latency of 150 to 250ms before it starts,

during which the brain decides where to go. Vision is largely suppressed during the movement itself.

At 24 frames per second, that is a total of four to seven frames between something appearing and the viewer actually seeing it. On a piece of animation that lasts two seconds, you have just spent a quarter of it on eye travel.

This is the entire subject. Everything below follows from it.

From an element appearing to a viewer seeing it



This is why the ease-in you spent an hour on is frequently never watched. Either bring the next element in near where the last one left, or hold a beat of six to ten frames so there is time to travel. On a two-second piece, eye travel alone can take a quarter of the running time.

Eye-trace: put the next thing where the eye is

The technique comes from film editing, where it is called eye-trace, and it applies exactly to motion graphics.

If an element exits screen-right on frame 40 and the next element enters screen-left on frame 42, the viewer misses the entrance. They were looking right. They will catch the new element somewhere around frame 48, having missed its first six frames — which, if those frames contained the ease-in you spent an hour on, is a poor return.

The fix costs nothing. Either bring the new element in near where the last one left, or hold a beat — six to ten frames — so there is time to travel. Hitchcock's version of this was that the cut should land where the audience is already looking.

The same rule governs transitions between scenes in a title sequence, between slides in a deck that animates, and between screens in an app.

What pulls the eye, in order

When several things compete, the periphery resolves it roughly like this:

1. **Motion**, especially onset of motion. A thing that starts moving beats a thing already moving.
2. **Faces**, and anything face-like. Reliable to the point of being a nuisance.
3. **High luminance contrast** — a bright thing on a dark field, more than a saturated thing on a grey one.
4. **Text**, once it is fixated, because reading captures attention and holds it.

This ordering is why a small animated element in a corner destroys a piece: it wins against the large static thing you wanted read.

It is also the argument against animating two things at once. If both move, one wins by accident rather than by your decision, and you cannot predict which.

The reading problem

Text and motion compete for the same resource. A person reading is fixating and re-fixating along a line; they are not available to watch anything move. So the standard sequencing for anything with words:

1. Motion brings the text in.
2. Motion stops completely.
3. The text is held, still, for the reading time.
4. Motion takes it out.

The held section is not dead time; it is the only part where the text does its job. The most common fault in amateur motion graphics is that the animation runs for the whole duration and the reading time was never budgeted.

Testing it without eye-tracking equipment

You do not need a lab.

- **Point as you watch.** Play the piece at full speed and physically point at where you are looking. You will catch your own misses.
- **Ask one fresh person, once.** Play it once, then ask what it said. You only get one first viewing per person, so spend it deliberately.
- **The single-frame test.** Scrub to any frame and ask: if a viewer arrived at exactly this moment, where would they look, and is that where the meaning is?
- **Watch it small.** Shrink the preview to phone size on your desk. Most competing-element problems become obvious when everything is within the fovea at once — and then you know the full-size version is worse, not better.

The limitation

All of this describes a first viewing by somebody who does not know what is coming. You, having built it, are the worst possible judge: you know where the logo appears, so your eye is already there and the piece plays perfectly for an audience of one.

There is no technique that restores a naive viewing to the person who made the thing. The only real instrument is somebody else's first look, and you get one of those per person. Use them on the version you think is finished, not on the rough cut.

BEFORE YOU MOVE ON

A title sequence has an element exit screen-right on frame 40 and the next enter screen-left on frame 42, both at 24fps. The client says the second element 'appears from nowhere'. What is happening?

A. Two frames is below the flicker fusion threshold, so the second element is not registered as a separate event at all.

B. The eye is still fixated screen-right and takes several frames to travel, so the entrance and its ease-in happen outside the viewer's sharp vision.

C. The second element needs a longer ease-in; two frames of overlap is not enough time for an entrance to read.

D. Motion onset in the peripheral field is neurologically suppressed during a saccade, so peripheral vision is unable to detect a new element beginning to move and the entrance is never registered.

CASE STUDY · MODULE 1

The sting the client kept calling cheap

A two-person motion studio in Indore, three days before a co-operative bank's rebrand goes live.

Ritu and Anand had five days and forty thousand rupees to make a five-second logo sting. It would run at the head of every branch video and on the idle screen of a hundred and forty ATMs, so it would be seen by the same tellers several hundred times a year. The contract included two revisions. A third would be at the studio's cost.

The first cut was competent. The logo slid in from the left over twenty frames at twenty-five frames a second, the tagline faded up underneath, the colours matched the brand sheet and the typography had already been approved by the bank's agency. The marketing head watched it twice and said it looked cheap. Asked what he meant, he could not say. Asked what would fix it, he could: motion blur on the logo, a light sweep across the letters, and a few drifting particles behind them.

Anand could do all three in an afternoon. He had the presets.

Ritu opened the graph editor instead, which she admits she did not usually do. The speed graph was a rectangle. The logo covered exactly twenty-four pixels on every one of the twenty frames, went from nothing to seven hundred and twenty pixels a second in no time at all, and stopped dead. The tagline started and stopped on the same frames as the logo, so the two elements moved as one rigid block.

That gave them a decision with three days left, and both answers cost something.

Doing what the client asked was four hours. It was visible, it was pointable-at in the next review, and it would have produced a floaty animation that was now slightly soft. Motion blur exists to stop fast motion strobing and to match synthetic motion to live footage; it does not create mass, and neither does a glow.

The other answer was to tell a paying client that the three things he had asked for would not fix his problem, and then to hand back something that looked, in the timeline, almost identical to the version he had rejected. The changelog would say "adjusted easing". There would be nothing to point at. If it came back a third time, the studio would be working for nothing against a live date.

Ritu re-spaced the move: a four-frame anticipation in which the logo scaled down to ninety-six per cent, most of the travel in the first third, an overshoot to about a hundred and six per cent arriving fast, and a settle back to a hundred taking two and a half times as long as the overshoot. She offset the tagline three frames behind the logo and let it arrive after it. Nothing else changed — same colours, same type, same five seconds.

Then she hedged. She rendered three versions: the original with the requested effects, the re-spaced version with no effects at all, and the re-spaced version with the effects on top. She took all three to the review and did not say which was which.

There was one more decision buried in this, and it was about where the sting would live. On a branch video it is watched once. On an ATM idle screen it loops in front of a teller for six hours a day. Anticipation and overshoot are exactly the principles that make a single viewing memorable and the four-hundredth unbearable, so Ritu built the ATM cut with the same spacing and the overshoot reduced from six per cent to two — small enough to be felt rather than seen. Two files, one decision, and it added twenty minutes.

STOP HERE

Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

The marketing head picked the re-spaced version with no effects, and described it as "the one where the logo has some weight to it" — a word nobody in the studio had used in front of him. He then asked whether the shimmer

could go on top of that one. Ritu played the third version, he watched it twice, and he dropped the idea himself. The invoice went out at two revisions. Six months later the bank commissioned the branch-induction series and named the sting when they did. Anand now opens the graph editor before he opens the effects panel; he estimates the habit costs him thirty seconds a shot and has saved him about a week a year.

The client asked for motion blur, a light sweep and particles. Why would none of the three have fixed the sting?

All three are applied on top of a move whose distance is spread evenly across every frame. Weight comes from spacing — uneven distance per frame, plus a wind-up before the move and a settle after it. Motion blur has two real jobs, stopping fast motion strobing and matching synthetic motion to filmed footage, and supplying mass is not one of them. Adding it to a floaty animation gives you a floaty animation that is slightly soft.

Ritu showed three versions without labelling them. What did that buy her that an explanation would not have?

The client had a real perception — it looks cheap — and a wrong diagnosis. Explaining the diagnosis puts the studio's authority against the client's, three days before a live date. Showing the versions unlabelled let him keep the perception and drop the diagnosis by himself, and it tested the claim rather than asserting it. If he had picked the effects version, Ritu would have been wrong and would have known in ten seconds.

What would have made the whole argument unnecessary?

Opening the speed graph before the first review. A rectangular speed graph is the signature of the linear default and it is visible in two seconds. The disagreement came from shipping a review cut that nobody had checked in the one panel that says what the software is doing between the keyframes.

MODULE 1 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 You step through a twenty-frame move and find the object exactly halfway across at frame ten. What does that tell you?

A. The motion is linear, whatever easing the software reports

B. The motion has a correct ease-out and will read as responsive

C. The frame rate is wrong for the duration you chose

D. Motion blur is disabled on the layer or the composition

2 A dropdown opens with a 400 ms ease-in-out and users report that the app feels laggy. What is the mechanism?

A. Four hundred milliseconds is past the one-second limit for thought

B. Ease-in-out is defined only for opacity, and behaves unpredictably on position changes

C. It barely moves for its first hundred milliseconds, so nothing answers the tap

D. The dropdown animates a property the compositor cannot handle

3 Why is a spring the wrong animation for a download progress bar?

A. A spring cannot be interrupted once it has started

B. An underdamped spring overshoots, so the bar passes one hundred per cent

C. Springs are main-thread only and cannot run on the compositor

D. Spring duration is fixed by the library and cannot be matched to the download time

4 A finished 24 fps composition is changed to 25 fps in After Effects. What happens to the spacing you set by hand?

A. Keyframes keep their frame numbers, so the animation gets shorter
B. Nothing changes, because keyframes are stored in seconds and a second is a second
C. The software refuses the change until every keyframe is re-timed by hand
D. Keyframes keep their time values and get resampled to the nearest frame

5 Why is a three-frame lag on the button a user has just pressed a bug rather than a flourish?

A. Secondary elements must always move before the primary one, never after
B. A three-frame lag desynchronises the button from its own press sound effect
C. Follow-through only works on elements that travel more than 600 pixels
D. Perceived responsiveness is measured from input to first visible change

6 An element exits screen-right at frame forty and the next enters screen-left at frame forty-two. What is missed, and why?

A. Nothing, because peripheral vision detects motion onset immediately
B. The new element's first frames: a saccade takes 150 to 250 ms to begin
C. The exit, because the eye has already travelled to the left of the frame
D. Both, because two animations cannot be processed in the same second

MODULE 2

Motion that carries meaning

Animation in an interface as a functional layer rather than decoration: what a transition claims, how long it should take, which properties the browser can animate cheaply, and what to do for the substantial number of people who are made ill by large-field motion.

> BY THE END OF THIS MODULE YOU CAN
>
> Specify the motion in an interface as something a second developer could build from a written brief — naming which element moves, for how many milliseconds, on which curve, in what order relative to its neighbours, what it does when the viewer has asked for reduced motion, and which CSS properties the browser can animate without re-running layout — and justify each number from the perceptual or rendering limit it comes from

10 · 9 min

The three jobs an animation can do

> THE ONE THING TO KEEP
>
> Interface motion earns its place only by answering what just happened, where a thing came from, or whether the system is alive — and the causal reading depends on the movement starting within roughly a tenth of a second of the action that caused it.

Motion in an interface is not decoration

An animation on a screen is doing one of three jobs, and if it is doing none of them it is costing time and giving nothing back.

1. **It says what just happened.** A row slides out and the rows below close the gap: deleted. The same row vanishing instantly could have been deleted, filtered, or a rendering bug.
2. **It says where a thing came from.** A panel that grows out of the button you pressed is understood as belonging to that button. The same panel fading in at the centre of the screen is a new, unexplained object.
3. **It says the system is alive.** Something moving is the cheapest possible evidence that your tap was received, and it buys the server a few hundred milliseconds of goodwill.

Everything else — the logo that spins on load, the card that wobbles on hover — is being paid for out of the user's attention with no return.

The numbers that make this concrete

Three thresholds have survived four decades of testing and are worth memorising.

- **Around 100ms.** Below this, a response feels instantaneous — the result feels caused by your own action rather than reported to you afterwards.
- **Around 1 second.** The limit for uninterrupted thought. Up to a second, attention stays on the task; past it, people notice they are waiting.
- **Around 10 seconds.** The limit for keeping attention on a dialogue at all. Past this, people switch to something else and come back.

These come from Jakob Nielsen's 1993 summary of much older human-factors work, and they are about human perception rather than about any particular device, which is why they have not moved.

There is a fourth number worth knowing: the **Doherty threshold**, from an IBM study in 1982. When a system responded in under 400ms, operators did not just work faster — they entered a different working state, kept going, and productivity rose out of proportion to the time saved. This is the honest argument for keeping interface motion short: past roughly 400ms of total response, you have pushed the user out of a flow state to show them an animation.

The trap: animation time is added to waiting time

Most people building interfaces think of an animation as filling time that was already being spent. It usually is not.

If a modal opens in 300ms and the data inside it arrives in 200ms, the user waited 300ms, not 200ms. Your animation was the slow part. The fix is not a faster animation; it is starting the network request when the button is pressed rather than when the animation finishes, and letting the content arrive during the transition.

Test this the honest way: record your screen at 60fps, then step through the recording frame by frame and count from the frame where your finger touches to the frame where the content is readable. Most interfaces come in 200 to 400ms slower than their developers believe. On a phone, a 240fps slow-motion video of a real device is better still, because it includes the touch latency the simulator hides.

Causality is the thing you are actually buying

There is a well-documented perceptual effect behind all of this. When object A moves, stops next to B, and B immediately moves off, people do not see two events — they see A cause B to move. This is Michotte's launching effect, and it holds only within a tight timing window. Insert a gap of more than roughly 100 to 150ms between the two movements and the impression of causation collapses into two unrelated events.

That is the whole mechanism behind "the panel should come from the button". If the panel's growth starts within about a tenth of a second of the press and starts from the button's position, the two are perceived as one causal event. Delay it or move its origin and the user has to reason about the connection instead of seeing it.

What to do when you have nothing to animate

Sometimes the honest answer is no motion.

- **A state change with no spatial relationship** — a toggle flipping a setting somewhere else — is better served by a cross-fade or by simply updating, not by sliding something across the screen to imply a journey that did not happen.
- **A destructive action** wants a pause, not a flourish. Motion that reads as smooth and pleasant on a delete confirmation is working against you.
- **Data that changes because of somebody else** — a live figure updating — deserves a small, non-competing highlight, not the same treatment as the user's own action.

The free path

Every idea here is free to practise.

- **The browser's own DevTools** show you frame timing, with no plugin and no licence. Chrome and Firefox both let you slow animations down and inspect the easing curve applied to a running element.
- **Penpot** and **Figma's** free tier both do enough prototyping to test whether a transition reads. **Penpot** is open source and self-hostable if you need that.
- **Motion** (formerly Framer Motion) and **GSAP** are both usable free for most work, and plain CSS transitions cover the majority of interface motion without any library at all.

The limitation

Motion cannot fix a structure that does not make sense. If people cannot find the settings page, animating the drawer that contains it changes nothing — you have made a confusing journey smoother, and smoothness is not the same as comprehension. Motion is very good at explaining a relationship that genuinely exists in your model, and very bad at inventing one that does not. When a transition is hard to design, that is usually the interface telling you the two states are not actually related.

BEFORE YOU MOVE ON

A modal takes 300ms to animate open, and the data inside it arrives from the server 200ms after the button is pressed. The team argues about whether to shorten the animation. What is the actual situation?

A. The animation is free because the network request is already running at the same moment, so the user is only ever waiting the 200ms the server takes and shortening the transition would gain nobody anything at all.

B. The user waits 300ms, not 200ms, because the animation is the slower of the two — the fix is to start the request on press so the data arrives during the transition, not to cut the animation.

C. The user waits 500ms, because an animation and a network request cannot overlap in a browser and must run one after the other.

D. The wait is irrelevant because 300ms is below the Doherty threshold of 400ms, so no perceptible cost exists and the team should stop discussing it and ship whichever version looks better.

11 · 9 min

How long an animation should take

THE ONE THING TO KEEP

Durations come from a small named set rather than from feel, exits run at roughly three-quarters of entrance time, and distance scales duration sub-linearly — but perceived length is set by the shape of the curve's tail more than by the number.

The question everybody guesses at

"How long should this animation be?" gets answered by feel, and feel drifts. The person who has watched their own transition four hundred times finds it too slow at 250ms and cuts it to 120ms, at which point it is a flicker for everybody who has never seen it before.

There are defensible numbers. They come from two facts: the eye needs a minimum exposure to register that something moved, and attention has a budget.

Working durations

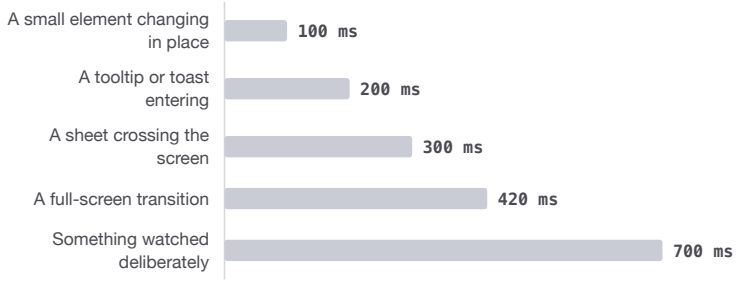
These ranges match what the large design systems publish, and more importantly they match what survives testing.

- **70 to 120ms** — a small element changing in place. A checkbox filling, a button's background shifting on press, an icon swapping. Below about 60ms the change is perceived as instantaneous rather than as a movement, which is often exactly what you want for a press state.
- **150 to 250ms** — a small element entering or leaving. A tooltip, a dropdown, a toast.
- **250 to 350ms** — a panel, sheet, or dialogue crossing a meaningful part of the screen.
- **350 to 500ms** — a full-screen transition, a page change, something covering most of the viewport.
- **Above 500ms** — reserve for something the user is watching deliberately: an onboarding illustration, a data visualisation building, a celebration. Not for anything that sits between a person and their task.

Two adjustments that matter more than the base number:

Exits are faster than entrances. Roughly 70 to 80 per cent of the entry duration. An arriving element is new information and the eye needs time to find it. A leaving element is already understood; lingering just makes the interface feel sticky.

Durations that survive testing



Two adjustments matter more than the base number. Exits run at roughly three-quarters of the entrance, because a leaving element is already understood. And distance scales duration sub-linearly — about 1.4 times the time for each doubling of travel — so an 800-pixel move at 500 ms feels the same speed as a 200-pixel move at 250 ms.

Distance scales the duration, but not linearly. Doubling the travel distance does not mean doubling the time — that produces a crawl on large screens. Material Design's published approach is a distance-based curve where duration grows roughly with the square root of distance; in practice, scaling time by about 1.4× for each doubling of distance keeps perceived speed constant. A 200px move at 250ms and a 800px move at 500ms feel like the same speed even though the second one is four times the distance at half the speed-per-pixel.

Why the same number feels wrong on different screens

A 300ms sheet transition tested on a 6-inch phone can feel sluggish on a 27-inch monitor and frantic on a television across a room. The reason is angular, not physical: what your eye tracks is degrees of visual field per second, and the same pixel distance subtends a very different angle at arm's length than at three metres.

This is why published duration guidance almost always names a device class. Material's numbers are phone numbers. Apple's are tuned per platform, and tvOS motion is noticeably slower and larger than iOS motion for exactly this reason. Take any table of durations as calibrated for the screen it was measured on, then re-test.

Perceived duration is not measured duration

Two animations of identical length can feel different lengths.

- An animation with a **long slow tail** — a curve that spends its last third creeping the final few pixels — reads as longer than its clock time, because the end is the part you wait for. This is the single most common cause of "it feels slow" on a transition that measures 250ms.
- An animation that **starts immediately** reads as shorter, because the response latency is what people actually judge. A 300ms transition that begins on the same frame as the tap feels faster than a 200ms one that begins 100ms later.
- An animation **you have seen before** feels longer. Repeat exposure is why the durations you love on day one feel slow on day thirty, and why you should not trust your own judgement on something you have built.

The practical consequence: cut the tail before you cut the duration. Changing the curve so the last 10 per cent of the distance happens in the last 15 per cent of the time rather than the last 40 per cent will fix most "too slow" complaints without touching the number.

How to set your own numbers

1. Pick three durations, not twenty. Something like 150, 250 and 400ms covers nearly everything.
2. Write them down as named values, not as literals scattered through the code.
3. Test each on the slowest device you support, over a throttled connection, with someone who has not seen the product.
4. Record the result at 60fps and count frames. A 250ms transition is 15 frames at 60fps; if you count 28, something in your stack is adding time you did not budget for.

The free path

- **ffmpeg** will pull a screen recording apart into numbered frames so you can count: `ffmpeg -i capture.mp4 -vf fps=60 frame_%04d.png`. Counting frames is the only way to know the real duration rather than the intended one.
- **Chrome DevTools** and **Firefox Profiler** both show animation timing directly, free.
- **OBS Studio** records the screen at a known frame rate on Linux, Windows and macOS, free, and is the standard tool for this.

The limitation

None of these numbers are laws. They are a starting point calibrated on hand-held screens for people with typical vision and typical reaction times, and they say nothing about a user with a motor impairment who needs more time to track a moving target, or about a power user who runs the same flow two hundred times a day and wants everything instant. A configurable speed multiplier serves both better than a perfect single value, and almost nobody ships one.

BEFORE YOU MOVE ON

A 250ms panel transition is reported as feeling sluggish. Measuring confirms it really is 250ms and begins on the same frame as the tap. What is the most likely cause and fix?

A. The device is dropping frames, so the animation takes longer in practice than the specification claims; the fix is to promote the panel to its own compositor layer.

B. 250ms is simply above the perceptual limit for a panel; the fix is to cut it to around 120ms so it lands inside the window where responses feel instantaneous.

C. The easing curve has a long slow tail, so the last few pixels take a disproportionate share of the time; compress the tail before touching the duration.

D. The exit duration has been set equal to the entrance duration, and an exit that matches its entrance always reads as slow regardless of what the entrance does.

12 · 10 min

The transition is the explanation

THE ONE THING TO KEEP

A shared-element transition measures an element before and after, inverts the difference with a transform and plays it off — which is why it can only describe a relationship that genuinely exists between the two states.

Two screens, or one screen changing

When a user taps a thumbnail in a grid and lands on a detail page, there are two stories the interface can tell.

The first: this grid is gone, and here is a different page. The second: this item you tapped is the thing now filling your screen. The second story is nearly al-

ways the true one, and a transition is how you tell it.

The technique has a name in every framework — shared element transition, hero animation, container transform, view transition — and the same mechanism underneath. You identify one element that exists in both states, measure where it is in each, and animate between the two measurements while the rest of the interface cross-fades.

FLIP: the mechanism, in four steps

The standard implementation is called FLIP, from Paul Lewis, and it is worth understanding even if a framework does it for you, because when a shared-element transition goes wrong the fault is always in one of these four steps.

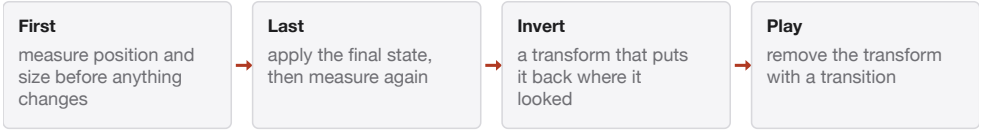
1. **First.** Before anything changes, measure the element's position and size. In the browser this is `getBoundingClientRect()`.
2. **Last.** Apply the final state — swap the DOM, navigate, change the layout — and measure again, immediately, before the browser paints.
3. **Invert.** Compute the difference and apply a `transform` that puts the element visually back where it started. The element is now in its final position in the layout, but looks as though it never moved.
4. **Play.** Remove the transform with a transition. The element travels from the inverted position to its real one.

The reason for this dance rather than simply animating width, height, top and left: steps 3 and 4 use only `transform`, which the browser can animate on the compositor without re-running layout. Animating the geometry properties directly means a full layout calculation on every frame, which is where dropped frames come from.

What makes one look wrong

Three failures cover nearly all of them.

FLIP, and where a shared-element transition goes wrong



Steps three and four use transform alone, which the browser animates on the compositor without re-running layout. Animating top, left, width and height directly forces a full layout calculation on every frame, and that is where the dropped frames come from. When one of these transitions looks wrong, the fault is in one of these four steps.

The aspect ratio changes during the move. A 1:1 thumbnail becoming a 16:9 header will squash visibly if you animate width and height independently. The fix is to animate a uniform scale and crop, or to cross-fade between two correctly-cropped images at the midpoint where the distortion is least visible. Most frameworks that do this well are cross-fading and hoping you do not look.

The text comes along for the ride. Scaling a container scales the type inside it, so a 14px caption becomes a blurry 32px caption mid-flight and then snaps back. The standard fix is a counter-scale: scale the child by the inverse of the parent's scale each frame, so the text stays at its true size throughout.

Nothing corresponds. If the two screens genuinely share no element, a shared-element transition has to invent one, and the result is a shape morphing into an unrelated shape. This reads as a special effect rather than an explanation. A cross-fade is the honest answer.

The rule underneath

The transition should describe a relationship that is already true in your data model.

- Tapping an item in a list and seeing that item expand: the item is the same object. Shared element is correct.
- Moving from step 2 to step 3 of a form: these are positions in a sequence. A horizontal slide is correct, and the direction should reverse when going back.
- Switching between two unrelated top-level tabs: there is no journey. A cross-fade, or nothing, is correct. Sliding implies the tabs sit side by side in

a space, which is a claim you probably do not want to make — and which breaks the moment somebody navigates to the fourth tab from the first.

When a transition is hard to design, it is usually because you are trying to describe a relationship that does not exist.

Doing it without a framework

The browser now has `document.startViewTransition()`, which handles the capture-and-interpolate work natively for same-document changes and, with the multi-page variant, across navigations. You tag the element in both states with the same `view-transition-name` and the browser does the FLIP for you. It degrades to an instant change in browsers that do not support it, which is the correct fallback.

For everything else, FLIP in about thirty lines of plain JavaScript covers it, no library required. GSAP's Flip plugin and Motion's layout animations are convenience on top of the same four steps.

The free path

- **View Transitions API** — built into the browser, nothing to install.
- **Motion** (the open-source successor to Framer Motion) — free, and its `layout` prop is FLIP with the measurement handled.
- **GSAP** — now free for all use including commercial, Flip plugin included.
- **Penpot** — open-source design tool, enough prototyping to test a transition before building it.

The limitation

A shared-element transition costs frames, and it costs them at the worst moment: while a new screen is mounting and probably fetching data. On a low-end device, the transition and the render compete, and the result is a janky animation followed by a late-arriving screen — worse than no transition at all.

The defensible rule is to make the transition cancellable and to skip it under load. If the element's destination is not measurable within one frame, do not

animate; just change. A transition is an explanation, and an explanation delivered badly is worse than none.

BEFORE YOU MOVE ON

In a FLIP transition, why is the element moved back to its starting position with a transform in step three rather than by animating its top and left values directly?

A. Because transform is applied after layout, so the browser can run the animation on the compositor without recalculating layout on every frame.

B. Because top and left are only valid on absolutely positioned elements, and a shared element is usually in normal document flow where those properties are ignored entirely.

C. Because transform interpolates in a different colour space from the geometry properties, which is what keeps the movement visually smooth through the middle of the transition.

D. Because the transform is what gets captured by getBoundingClientRect in step two, so using top and left would mean the measurement taken in the Last step returns the pre-change position and the whole calculation inverts itself.

13 · 9 min

Staggering without making people wait

THE ONE THING TO KEEP

A stagger works by overlapping heavily — 20 to 60ms of offset per item with a capped total window — and starting the next element when the previous is roughly halfway, because the eye reads intent long before a movement settles.

One thing at a time, but not one after the other

A list of eight cards appearing. The naive implementation animates all eight identically at the same instant, which reads as a single block appearing — you have gained nothing over showing them instantly. The over-corrected implementation animates them one at a time, each waiting for the last to finish, and eight cards at 300ms each takes 2.4 seconds, which is an eternity.

The technique in between is the stagger: each item starts a fixed offset after the previous one, and they overlap heavily.

The numbers

- **20 to 60ms of offset per item** is the useful range for interface work. Below about 15ms the eye cannot separate them and it reads as simultaneous. Above about 80ms it reads as a queue and the last item feels late.
- **Cap the total.** With a 40ms stagger, twenty items means the last one starts 760ms after the first — too slow. The fixes are either a maximum total stagger window (divide the window by the item count) or a cap on how many items stagger at all, with the rest appearing together.
- **Only stagger what is visible.** Staggering items below the fold wastes the budget on things nobody is looking at, and on a long list it can mean animating hundreds of elements.

In CSS this is usually a custom property set per item:

```
.card { animation: rise 260ms var(--ease-enter) both;
        animation-delay: calc(var(--i) * 40ms); }
```

with `--i` set inline from the index. In the Web Animations API it is a `delay` per element; in GSAP and Motion it is a `stagger` option that does the same arithmetic.

Direction carries meaning

A stagger has an order, and the order is a statement.

- **Top to bottom** for a list: matches reading order, feels like the content filling in.
- **From the point of interaction** outward: a menu opening from the button it belongs to, items nearest the button first. This reinforces the causal link.
- **Reverse on exit.** If items entered top-first, they should leave bottom-first, so the exit is not a replay of the entrance. Most libraries let you reverse the stagger direction with one flag, and almost nobody does it.
- **Never random.** A randomised stagger reads as broken rather than organic, because the eye is looking for the pattern and cannot find one.

Overlap, not sequence

For a compound transition — a sheet rising while its header fades in while its content slides up — the amateur version runs these in series and takes 900ms. The professional version overlaps them so the whole thing takes 400ms and still reads as three distinct events.

The rule that works: start the next element when the previous one is about 40 to 60 per cent through, not when it finishes. The eye has already registered the first movement's direction and intent by then; waiting for the settle adds time and communicates nothing.

This is easier to reason about with a timeline than with delays. GSAP's timeline, Motion's sequence array, and After Effects' layer bar view are all the

same idea: positions on a shared clock rather than numbers added up in your head. Once you have more than three overlapping elements, delay arithmetic becomes unmaintainable — change one duration and every downstream number is wrong.

One focal point

The strongest constraint in orchestration is attention: people can attend to one moving thing. If four elements move in four directions at once, the viewer picks one and misses the rest.

So decide what the transition is *about*. If a dialogue opens, the dialogue is the event; the background dimming is support and should be slower, smaller and less contrasty. If a card expands into a page, the card is the event and everything else should cross-fade. When everything is animated with equal energy, nothing is emphasised — the same failure as a document where every word is bold.

When staggering is wrong

- **A list the user already knows.** Returning to a list you have seen a hundred times and watching it assemble itself is friction, not delight. Consider staggering only on first load.
- **Anything the user is about to interact with quickly.** If the third item is the one they always tap, a 120ms delay before it is tappable is a cost with no benefit. Make items interactive immediately even while animating.
- **Search results.** People scan these fast. Staggering delays the scan.
- **Under reduced motion.** Collapse the stagger to zero and cross-fade the set.

The free path

- **GSAP** — free for all use now, and its `stagger` object handles grids, distance-from-a-point ordering and reverse.
- **Motion** — open source, `staggerChildren` and `delayChildren` on a parent variant.

- **Plain CSS** — an index custom property and `calc()` covers most of it with no dependency at all.
- **Blender**, for non-web work, has an offset-per-object approach through drivers or the NLA editor, free.

The limitation

Stagger is the effect most likely to be praised in a portfolio and disliked in daily use. It photographs well and it tests badly, because a reviewer watches it once and a user meets it forty times a day. The honest position is that it belongs in moments that happen rarely — a first load, an empty state filling, a result arriving after work — and does not belong in the paths people repeat. Very few teams measure this, and most ship the same stagger everywhere because it looked good in the review.

BEFORE YOU MOVE ON

A list of 24 cards uses a 60ms stagger. Reviewers like it; users call the screen slow. What is the mechanism behind the complaint?

A. Each card's own animation is being lengthened by the delay, so the twenty-fourth card animates over a much longer duration than the first.

B. Twenty-four items at 60ms means the last card starts nearly 1.4 seconds after the first, which is past the limit for uninterrupted attention — and reviewers watch once while users meet it repeatedly.

C. The stagger is running on the main thread rather than the compositor, so with 24 simultaneous animations the browser cannot maintain the frame budget.

D. The cards are being revealed in reading order from the top rather than outward from the point the user tapped, and any ordering other than outward-from-the-tap is read as arbitrary by the viewer and therefore reported as slowness.

14 · 10 min

Why your lower-third looks like a template

THE ONE THING TO KEEP

Animation time is not reading time — budget two and a half to three seconds of still text first, then add the entrance and exit on top.

Why your name plate looks like a template

Because everything in it moves. The bar wipes in, the name flies up, the role fades from the side, a shine sweeps across, and a particle drifts past. Each choice was fine on its own. Together they say: this was assembled from parts.

Motion is a way of pointing. If every element moves, nothing has been pointed at. The professional-looking version usually animates one thing and holds the rest.

Reading time is the constraint nobody budgets for

A viewer needs roughly two and a half to three seconds of *still, settled* text to take in a name and a job title, longer if the name is unfamiliar to them or if they are reading in a second language — which, on the internet, is most of your audience.

Here is where it goes wrong. A three-second lower-third with a 1.2-second animated entrance and a 0.8-second exit leaves one second of readable time. The name was on screen for three seconds and nobody read it. The animation is not reading time.

Budget backwards: decide the hold first (three seconds), then add the in and out on top. Six seconds of lower-third feels long in the timeline and correct to a viewer.

Making the words readable

Type reveals, and how each one feels

- **Opacity fade.** Neutral, safe, invisible. Good when the graphic is not the point.
- **Mask or wipe reveal.** A rectangle scaling across the text, revealing it. Reads as printing or typing. Crisp and expensive-feeling. Animate the mask, never the letters.
- **Position plus fade.** The most common. Keep the distance small: 8–16 pixels of travel, not 200. Long travel on text is the single clearest amateur signature.
- **Per-character stagger.** Energetic, and illegible past about eight characters. Use it on one short word.
- **Scale from 1.04 down to 1.0** with a fade. Subtle, and it reads as authority.

Kinetic typography — text as the main event — follows one rule: the word that changes is the word that moves. And when you cut text to a voiceover, cut on the stressed syllable rather than the word boundary. The stress is where the ear expects the beat, and matching it is the difference between text that dances with the audio and text that merely keeps up.

Where the frame is actually cut

Whatever you design will be reframed. A 16:9 video becomes a 9:16 vertical clip. The platform lays its own interface over the bottom of the frame — captions, a username, a description, a row of buttons down the right edge. The lower third of a 16:9 frame is precisely where a caption bar lands.

Practical rules:

- Keep essential text inside the middle 80 per cent horizontally.
- Keep it above the bottom 20 per cent, even though the name of the graphic tells you to do the opposite.
- If the piece will be cut vertical, check it in a 9:16 crop before you build the graphics, not after.

For legibility, a drop shadow at default settings does very little. What works is a scrim: a soft dark gradient behind the text, tall and low in contrast, or a wide, very soft shadow at low opacity combined with a 1–2 pixel dark outline. Test it against the brightest frame the text will sit over, not an average one.

Fonts carry licences, and a licence for print does not always cover embedding in video. Design foundations covers choosing type; the licence question is worth a minute before you deliver a client's video.

Tools, and the caption style everyone uses

Tools, including the free ones

- **DaVinci Resolve** (free): the Text+ tool is genuinely powerful — real controls, and the Fusion page underneath it for anything custom. This is the best free option by some distance.
- **Premiere Pro**: Essential Graphics, with saved templates. Paid, and what most job ads name.
- **Kdenlive** and **Shotcut** (free): basic title animation. Fine for a fade and a slide.
- **Canva** (free tier): animated text presets. Limited easing, no custom curves, and completely adequate for a simple title card.
- **CapCut** on a phone: auto-captions and text presets, free, and the fastest path if you have no laptop.
- **CSS** for anything on the web, where a lower-third is just a positioned element with a transition.

The caption style, honestly assessed

One word at a time, large, centred, popping and colour-changing in time with the speech. It works — on a muted feed it holds attention, and the platforms' own tools generate it in one tap.

It is also hard work for anyone reading English as a second or third language, because it removes the ability to read ahead. Chunks of two to four words, held for a beat, keep most of the retention benefit and are considerably kinder. Pick deliberately rather than by default.

And read your auto-captions. They get names, place names, technical terms and any non-English word wrong, reliably, every time.

Today

Mute one of your videos and watch it. If you cannot tell who is talking, what the piece is about, or what you are meant to do at the end, the graphics are decorative rather than working.

BEFORE YOU MOVE ON

A three-second lower-third has a 1.2-second animated entrance and a 0.8-second exit. In testing, viewers consistently say they missed the guest's name. What is the real problem?

A. The type is too small for phone viewing, and increasing the point size will fix it.

B. Only about one second of the three is settled, readable text, and a name plus a role needs roughly two and a half to three seconds of stillness — more for someone reading in a second language.

C. There is not enough contrast between the text and the footage behind it, so a heavier shadow is needed.

D. The entrance is too slow to catch the eye, and a faster, flashier animation would draw attention to the name.

15 · 8 min

Naming the motion so it stops drifting

THE ONE THING TO KEEP

Motion tokens are named by role rather than by value so the number can change without the name becoming a lie, and redefining them inside a reduced-motion query is the cheapest way to make every future component inherit the right behaviour.

The problem a token solves

Open any mature codebase and search for `cubic-bezier`. You will find eleven different curves, three of which differ in the third decimal place, none of which anybody chose deliberately. Search for `transition:` and you will find 180ms, 200ms, 0.2s, 250ms and 0.3s, all meaning "a short one".

This is not untidiness. It is the reason the product feels like several products. Consistent motion is one of the strongest signals that an interface was made by one team, and inconsistent motion is noticed before it is identified — people report that something "feels cheap" without being able to say why.

A motion token is the fix: a named value, defined once, referenced everywhere.

What a motion system contains

Three sets of tokens, and a fourth thing that is not a token.

Durations. Three to five, no more.

```
:root {
  --duration-instant: 100ms;
  --duration-fast:    180ms;
  --duration-normal:  260ms;
  --duration-slow:    400ms;
}
```

Easing curves. Four, named by what they are for rather than by their shape.

```
:root {
  /* decelerate: things arriving. Fast start, settle in. */
  --ease-enter: cubic-bezier(0.05, 0.7, 0.1, 1);
  /* accelerate: things leaving. Slow start, fast exit. */
  --ease-exit:  cubic-bezier(0.3, 0, 0.8, 0.15);
  /* both ends: things moving within the screen. */
  --ease-move:  cubic-bezier(0.2, 0, 0, 1);
  /* emphasis: a small overshoot for something that wants at-
tention. */
  --ease-snap:  cubic-bezier(0.3, 1.4, 0.5, 1);
}
```

Combined recipes. The pairing is the real decision. `--motion-sheet-enter: var(--duration-normal) var(--ease-enter)` means nobody has to remember which curve goes with which duration.

The fourth thing, which is not a token and must be written down anyway: **which property moves**. A system that says "sheets use 260ms ease-enter" without saying "sheets translate on Y and fade, they do not scale" will still produce four different sheets.

Naming: by role, not by value

Name a token `--duration-fast`, not `--duration-180`. The second one becomes a lie the first time somebody decides 180 was too slow, and then you have `--duration-180: 200ms` in your codebase, which is worse than a raw number.

Role names also survive platform differences. The same `--duration-normal` can be 260ms on the web and 300ms on a television, and the calling code

does not change.

Why the curves look like that

A cubic-bezier easing function in CSS takes four numbers: the x and y of two control points, with the start fixed at (0,0) and the end at (1,1). The x axis is time, the y axis is progress.

- Both control points low on y early, then rising: slow start, fast finish — acceleration, for exits.
- Both high on y early: fast start, slow finish — deceleration, for entrances.
- A y value above 1: the curve goes past the destination and comes back — overshoot.
- A y value below 0: it pulls backwards first — anticipation.

CSS lets you exceed the 0–1 range on y but not on x, because time cannot run backwards. This is why you can make a bounce with a bezier but not a full spring with multiple oscillations; for that you need `linear()` with sampled points, or a JavaScript spring.

The token nobody defines and everybody needs

Add a reduced-motion variant in the same place:

```
@media (prefers-reduced-motion: reduce) {
  :root {
    --duration-instant: 1ms;
    --duration-fast:    1ms;
    --duration-normal:  120ms;
    --duration-slow:    120ms;
    --ease-snap: var(--ease-move); /* no overshoot */
  }
}
```

Redefining the tokens rather than overriding each component means every component inherits the behaviour, including the ones written next year by somebody who has not read this lesson. This is the single highest-value thing tokens buy you.

Making it real

Tokens only work if the raw values become unusable. In practice:

- A lint rule that fails a build on a literal `cubic-bezier(` or a bare `ms` value outside the token file. **Stylelint** does this free with `declaration-property-value-allowed-list`.
- One file, in version control, as the only definition. Design tools and code read from the same source if you can manage it; **Style Dictionary** is the open-source tool that compiles one token file into CSS, iOS, Android and JSON outputs.
- A page in the product that renders every token as a live example. Not a document — a page you can open on a real device and watch. Motion cannot be reviewed in a static specification.

The free path

Nothing here needs a paid tool. CSS custom properties are native. Style Dictionary is Apache-licensed. Stylelint is MIT. **Penpot** is open source and supports tokens directly. Figma's variables do the same thing on the free tier for small teams, and the important part — the shared vocabulary — costs nothing either way.

The limitation

Tokens make motion consistent. They do not make it good, and they can entrench a bad decision across an entire product faster than inconsistency ever did. A wrong `--duration-normal` is now wrong in 400 places.

They also cannot express the thing that matters most in a complex transition: orchestration. Which element moves first, how much the second one overlaps the first, what waits for what. A token file has no vocabulary for that, so it gets encoded in each component separately and drifts anyway. The honest summary is that tokens solve the vocabulary problem and leave the composition problem entirely open.

BEFORE YOU MOVE ON

Why is it better to redefine the duration tokens inside a prefers-reduced-motion media query than to add a reduced-motion override to each component?

A. Because custom properties defined inside a media query take priority over component rules regardless of specificity, which is the only way to reliably override a third-party component.

B. Because components written later inherit the behaviour without their authors having to know the rule exists, so the coverage does not decay as the codebase grows.

C. Because the media query is evaluated once at page load, so the browser can skip creating compositor layers for every animation in the document and save GPU memory.

D. Because per-component overrides would need a JavaScript listener on the matchMedia object, and a listener that is registered per component will leak memory as components mount and unmount over a long session.

16 · 9 min

Spinners, skeletons and honest waiting

THE ONE THING TO KEEP

Delay the indicator by 200 to 300ms so fast responses never flash a spinner, use an indeterminate indicator whenever you do not genuinely know the progress, and treat a skeleton as a promise about layout that will cost you twice if it is wrong.

The indicator is a claim, and it can be false

A spinner says: something is happening, and it will finish. Both halves of that are claims about your system, and both are frequently untrue. A spinner that keeps turning after the request has failed is lying, and users learn fast that spinners lie, which is why people reload pages that were about to succeed.

Designing a waiting state starts with the duration, not the graphic.

Three bands

Under about 100ms. Show nothing. The response arrives inside the window where the action still feels instantaneous, and a spinner that appears and disappears within 100ms is a visible flash that makes a fast interaction feel broken. If you cannot know the duration in advance, delay the indicator: show nothing for the first 200 to 300ms, then show the spinner if the work is still running. Most people are surprised by how much this alone improves a product's perceived speed — you have removed flicker from every fast case and lost nothing on the slow ones.

Roughly 100ms to 1 second. Keep the user's context on screen and indicate locally. A button that shows a small inline spinner in place of its label. A section that dims. Do not blank the screen: replacing a full page of content with a centred spinner for 400ms is a worse experience than leaving the stale content visible with a subtle busy state.

Beyond 1 second. Now you need a real indicator, and past roughly 4 or 5 seconds you need one that shows progress or explains itself. Past 10 seconds, the interaction is over as far as attention is concerned — the user has switched tasks, and your job is to be findable when they come back, which means a persistent result rather than a modal they have to be present for.

Determinate, indeterminate, and the honest choice

A **determinate** progress bar claims you know how far along you are. Use it only when you do: a file upload knows its byte count, a multi-step job knows its steps.

An **indeterminate** indicator — the looping bar, the spinner — claims only that work is happening. It is the right choice far more often than it is used, because most web requests genuinely have unknown duration.

The failure people remember is a determinate bar that stalls at 90 per cent. Perceived duration research is consistent on this: a progress bar that decelerates toward the end feels much longer than one of identical total duration that

accelerates. If you must fake it, fake it in the direction of speeding up. The more defensible answer is to stop faking and use an indeterminate indicator, which makes no promise it cannot keep.

Skeleton screens: useful and frequently misused

A skeleton screen shows grey blocks in the shape of the content that is about to arrive. It works for two reasons: the layout does not jump when content lands, and the user can start orienting before the data exists.

It stops working when the skeleton does not match. If your skeleton shows three cards and four arrive, the page re-flows at exactly the moment the user started reading — you have replaced one jolt with two. A skeleton is a promise about layout, and a wrong promise is worse than a spinner.

The shimmer that sweeps across a skeleton is doing one honest job: distinguishing "loading" from "empty". Without it, grey blocks can read as a broken render. Keep it slow — a 1 to 1.5 second sweep — and remember that it is continuous motion, so it belongs inside your reduced-motion handling.

Optimistic updates, and what they cost

The fastest waiting state is no waiting state: apply the change locally, send the request, and reconcile afterwards. A "like" button should fill instantly.

This is correct for actions that almost always succeed and are cheap to reverse. It is wrong for anything where a silent rollback would mislead. If a payment optimistically shows as sent and then quietly reverts, you have built something worse than slow. The test: if this fails and the user has already moved on, what do they believe?

Measuring rather than guessing

Do not design waiting states against your own network. Throttle to a slow connection in DevTools and watch what your users see: which parts of the page arrive first, how long the gap is, whether the layout settles once or four times.

`ffmpeg -i capture.mp4 -vf fps=60 f_%04d.png` on a screen recording lets you count frames from click to first content. Chrome's Performance panel gives you Largest Contentful Paint and Cumulative Layout Shift directly, both free, and CLS in particular is the number that catches a mismatched skeleton.

The free path

Everything here is free. CSS animations for spinners and shimmers need no library. The `<progress>` element is native. For skeletons, a handful of divs with a background gradient and one keyframe animation is the entire implementation; the packaged libraries exist to save fifteen minutes, not to do anything you cannot.

The limitation

No waiting state makes waiting shorter. There is a real, measurable effect from good indicators — people report shorter perceived durations and abandon less — but it is worth a few seconds at most, and it does not compound. If your page takes eight seconds, the honest fix is in the eight seconds, not in the animation covering them.

There is also a documented perverse case: adding an artificial delay with a progress animation can make a result feel more trustworthy, which is why some products slow themselves down deliberately. That works, and it is manipulation. Knowing the mechanism is the point; whether you use it that way is a decision you should make out loud rather than by accident.

BEFORE YOU MOVE ON

A team adds a skeleton screen to a feed. Load feels smoother in testing, but the page's Cumulative Layout Shift score gets worse. What has most likely happened?

A. The shimmer animation on the skeleton is being counted as layout movement, because any animated element contributes to the layout shift metric while it is in motion.

B. Skeletons always increase layout shift because they add a render pass, and the correct trade is to accept a worse score for a better perceived experience.

C. The skeleton's shape does not match the content that arrives, so the page reflows when real data lands — replacing one jolt with two.

D. The skeleton is rendered before the fonts have loaded, so the metric is capturing the font swap rather than the skeleton, and the fix is font-display rather than anything to do with the loading state.

17 · 10 min

Motion that makes people ill

THE ONE THING TO KEEP

Large-area, parallax and zoom motion cause a visual-vestibular conflict that produces real nausea in a large minority of adults, so the media query should substitute a cross-fade rather than switch animation off — and the effects most likely to trigger it should not be in the default experience at all.

Motion makes some people ill, and the mechanism is specific

This is not a preference about taste. A significant number of people get dizziness, nausea, headache or disorientation from certain kinds of screen motion, and it is a physiological response with a known cause.

Your sense of balance comes from combining three signals: what your eyes see, what your inner ear's vestibular system reports about acceleration and head position, and proprioception from muscles and joints. When these disagree — your eyes report large-field movement while your inner ear reports that your head is still — the brain treats the conflict as a problem. The same conflict, in the other direction, causes seasickness.

The scale is larger than most developers assume. Analysis of US National Health and Nutrition Examination Survey data found that roughly 35 per cent of American adults aged 40 and over had some form of vestibular dysfunction. Not all of them are affected by screen motion, but this is not a rare edge case, and it overlaps with migraine, concussion recovery, and a long tail of conditions people do not disclose.

Which motion triggers it

The trigger is not "animation". It is a narrow set of properties:

- **Large area.** Motion that fills a large part of the visual field. A 40px toast sliding in is fine; a full-screen background that zooms is not.
- **Parallax.** Layers moving at different speeds is the strongest trigger in the set, because it is exactly the depth cue your visual system uses to judge self-motion. A page that parallaxes as you scroll is telling your visual system that you are moving through a space.
- **Scale over distance.** Something rushing toward the viewer, or the whole interface zooming in and out on navigation.
- **Spinning and rotation**, especially sustained.
- **Continuous, unstoppable motion** — an autoplaying background video, an infinite carousel.

Small, short, local motion is generally safe. That is the useful news: honouring reduced motion does not mean shipping a static product.

The switch exists and you can read it

Every major operating system exposes a reduce-motion setting, and the browser exposes it as a media query.

```
.card {
  transition: transform 250ms cubic-bezier(0.2, 0, 0, 1);
}

@media (prefers-reduced-motion: reduce) {
  .card {
    transition: opacity 150ms linear;
    transform: none;
  }
}
```

On iOS it is Settings → Accessibility → Motion → Reduce Motion. On Android, Settings → Accessibility → Remove animations. On macOS, System Settings → Accessibility → Display → Reduce motion. On Windows, Settings → Accessibility → Visual effects → Animation effects. In JavaScript: `window.matchMedia('(prefers-reduced-motion: reduce)').matches`, and listen for changes rather than reading it once at load.

Reduced is not removed

The most common mistake is treating the query as an off switch and disabling everything. That throws away the feedback function covered earlier in this module — the user who asked for less motion still needs to know their tap registered and still needs to know where the panel came from.

The substitutions that work:

- **Slide → cross-fade.** An opacity change carries no motion signal but still marks a state change.
- **Large travel → small travel.** Keep the motion, cut the distance to a few pixels.
- **Parallax → nothing.** This one genuinely has no safe reduced form; remove it.
- **Autoplay → a play button.** Anything running without being asked for should stop.
- **Long → short.** Halving durations reduces exposure without removing the cue.

Never set `* { animation: none !important; }`. It breaks CSS-driven functionality that depends on animation events firing, and it removes loading indicators, which is a usability regression for everyone.

What the standards actually require

Two WCAG success criteria are relevant and are often confused.

- **2.2.2 Pause, Stop, Hide (Level A)** — any motion that starts automatically, lasts more than five seconds, and is presented alongside other content, must have a mechanism to pause, stop or hide it.
- **2.3.3 Animation from Interactions (Level AAA)** — motion animation triggered by interaction can be disabled, unless it is essential.

The second one is AAA, which means it is not in the compliance bar most organisations target. That does not make it optional in any moral sense; it means you will not be forced to do it by a procurement checklist, and you should anyway.

Testing it

Turn the setting on in your own OS and use your product for a day. It is the only test that finds the animations you forgot about — the ones inside a third-party component, the ones in a video that autoplays, the scroll-linked header you stopped noticing months ago.

Browsers also let you emulate it: Chrome DevTools has it under Rendering → Emulate CSS media feature prefers-reduced-motion, and Firefox has the same under the Inspector's accessibility settings. Free, and it takes one click.

The limitation

The media query only reports an operating-system setting, and most people who would benefit have never found it. Somebody who gets migraines from parallax will usually leave your site rather than search their OS settings for a toggle they do not know exists. So the query is a floor, not a solution: it catches the people who already know what to ask for.

The stronger design position is to keep the large-field, parallax and zoom effects out of the default experience entirely, and reserve the media query for the smaller adjustments. A product that is comfortable by default needs the query least.

BEFORE YOU MOVE ON

A site honours prefers-reduced-motion by applying a global rule that disables all animations and transitions. Why is this a worse implementation than substituting cross-fades?

A. Because the operating-system setting is intended only to reduce the speed of animations, and disabling them entirely goes beyond what the user asked the system to do.

B. Because a global disable rule cannot reach animations defined inside a shadow DOM or a third-party iframe, so the effects most likely to cause harm are precisely the ones it misses.

C. Because it removes the feedback that tells a user their action registered, and it breaks loading indicators and any code waiting on animation events.

D. Because vestibular symptoms are caused by the abruptness of a change rather than by its extent, so an instant state change is a stronger trigger than the slide it replaced and the rule makes the symptom worse.

18 · 10 min

Why transform is cheap and width is not

THE ONE THING TO KEEP

Transform and opacity describe how an already-painted layer is placed, so animating them skips layout and paint and can run on the compositor thread even while the main thread is blocked — which is why they survive a heavy script task and a width animation does not.

Why the same animation is smooth in one property and jerky in another

Animate `transform: translateX(300px)` and it runs at the display's full rate on a five-year-old phone. Animate `left: 300px` and the same movement stutters. The distance is identical, the duration is identical, and the difference is entirely about which stages of the rendering pipeline each one forces the browser to repeat.

The pipeline

Every frame, a browser can do up to five things:

1. **Style** — work out which CSS rules apply and what the computed values are.
2. **Layout** — work out where every box goes and how big it is. Changing one element's geometry can move everything after it, so this is global work.
3. **Paint** — fill in pixels: text, colours, borders, shadows, into one or more layers.
4. **Composite** — take the painted layers and combine them into the final image, applying each layer's transform and opacity.
5. **Present** — hand it to the display.

The frame budget is the display's refresh interval: 16.7ms at 60Hz, 8.3ms at 120Hz. Everything above must finish inside that, and the browser has other work to do in it too — roughly 10ms is a realistic budget at 60Hz.

Layout is the expensive stage and it is not local. Changing `left` on one absolutely positioned element is cheap; changing `width` on an element in normal flow can force the browser to re-measure hundreds of siblings. Paint is expensive in proportion to area and to how complex the content is — text and shadows are dear.

The five stages a browser can do in one frame

1 Style	which rules apply, and what the values are
2 Layout	where every box goes — not local work
3 Paint	fill in pixels; costly by area
4 Composite	place and blend the painted layers
5 Present	hand the result to the display

The budget is 16.7 ms at 60 Hz, and the browser has other work inside it. Changing width re-runs stage two and everything after it, on the main thread. Transform and opacity are properties of an already-painted layer, so the browser skips stages one to three and can do the rest on the compositor thread — which keeps running while a 200-millisecond script task blocks the main one.

Transform and opacity are different. They are properties of a composited layer, not of its contents. Changing them does not change what was painted, only how the already-painted layer is placed and blended. So the browser can skip stages 1 to 3 entirely and, critically, can do the work on the compositor thread — which keeps running even when the main thread is blocked by JavaScript.

That last point is the one that matters on real devices. A transform animation survives a 200ms script task. A `left` animation freezes.

The practical list

Animate freely: `transform` (translate, scale, rotate, skew), `opacity`, and `filter` — though filters cost GPU time and can be slow on weak hardware.

Avoid animating: `width`, `height`, `top`, `left`, `right`, `bottom`, `margin`, `padding`, `border-width`, `font-size` — all trigger layout.

Avoid animating: `box-shadow`, `background-position`, `border-radius`, `color` — no layout, but a repaint every frame. Sometimes acceptable for a small element; noticeably expensive on a large one. A moving `box-shadow` on a full-width card is a common cause of a slow-feeling page, and the fix is to put the shadow on a pseudo-element and animate its opacity instead.

Layers are not free

`will-change: transform` promotes an element to its own compositor layer ahead of time, so the browser does not have to promote it mid-animation and drop a frame doing it.

It is also a memory allocation. A full-screen layer at 1920×1080 with 4 bytes per pixel is about 8MB of GPU memory, and on a low-end phone with a handful of such layers you will hit memory pressure and get *worse* performance, or crashes. The rules that follow from the mechanism:

- Add `will-change` shortly before the animation and remove it after.
- Never put it on a long list of elements "just in case".
- If you find yourself adding it everywhere, the problem is elsewhere.

Seeing it rather than guessing

Open DevTools → the three-dot menu → More tools → Rendering, and turn on **Paint flashing**. Any region that repaints flashes green. Run your animation. If large areas flash, you are painting every frame. This takes ten seconds and settles most arguments.

Then record in the Performance panel and look at the flame chart. Purple bands are layout and paint; a well-built transform animation shows almost none of them during the animation, just compositor work. The panel also marks long tasks over 50ms, which are what break main-thread animations.

The one exception worth knowing

The Web Animations API with `element.animate()` can hand a transform or opacity animation straight to the compositor, exactly like CSS. A `requestAnimationFrame` loop that sets `style.transform` each frame cannot — it is main-thread by definition, because your JavaScript has to run to produce each value.

This is the real reason to prefer declarative animation over a manual rAF loop: not elegance, but which thread the work lands on. When you genuinely need per-frame JavaScript — scroll-linked effects, physics — modern CSS scroll-driven animations and `ScrollTimeline` move a large part of that case back onto the compositor.

The free path

Chrome DevTools, Firefox Profiler and Safari's Web Inspector all do everything described here, free. There is no paid tool that tells you more about a frame budget than Paint flashing and a performance recording.

The limitation

"Animate transform and opacity" is good advice that has hardened into a superstition. Plenty of real interfaces need a height to animate — an accordion opening is the obvious one — and the answer is not to refuse. It is to know the cost, keep the animating subtree small, and consider the alternatives: a scaleY with a counter-scaled child, a clip-path, or CSS `grid-template-rows: 0fr → 1fr`, which does trigger layout but on a much smaller subtree than the whole page.

The other limitation is that the compositor is GPU work, and on some low-end Android devices the GPU is the bottleneck rather than the CPU. Transform-only animation does not help there, and the only fix is fewer and smaller layers.

BEFORE YOU MOVE ON

During a transform animation, a 200ms JavaScript task blocks the main thread. The animation continues smoothly. During an identical animation on the width property, it freezes. Why?

A. Width animations are interpolated in JavaScript by the browser's animation engine, while transforms are interpolated in CSS, and only the CSS path is exempt from main-thread blocking.

B. A width animation needs layout recalculated every frame, and layout runs on the main thread; a transform only changes how an existing layer is composited, which the compositor thread can do alone.

C. The transform animation is being run by the GPU, which caches the whole animation in advance, so it would continue even if the page were closed mid-flight.

D. The width animation has triggered a repaint of the element's text, and text rasterisation is the only operation in the rendering pipeline that is pinned to the main thread while every other stage can be parallelised onto worker threads.

CASE STUDY · MODULE 2

Three hundred milliseconds, measured on the phone the invigilators actually have

A two-person team maintaining the attendance app used by nineteen coaching centres in Kota.

The app does one thing that matters. An invigilator taps a batch, a sheet rises from the row they tapped carrying that batch's student list, and they mark absences before the test starts. The sheet enters over three hundred milliseconds on an ease-out, the rows arrive with a forty-millisecond stagger, and a scrim dims the list behind. On the team's laptops, and on the one recent phone in the office, it is pleasant.

The centres reported that the app hung when you opened a batch. Neither developer could reproduce it. The tickets kept coming, and one centre head said his staff had gone back to paper for the morning slot.

They borrowed two of the phones the invigilators use — four-year-old Androids on the cheapest tariff — recorded the screen at sixty frames a second, and counted frames from the finger touching the row to the student list being readable. Seven hundred and eighty milliseconds. The animation they had budgeted for was three hundred.

The extra came from three places, and none was the animation being slow. The network request for the student list started when the animation finished rather than when the finger landed, which added two hundred milliseconds that were purely arithmetic. The row stagger was uncapped, so on a batch of twenty-four the last row began nine hundred and twenty milliseconds after the first — and although the top rows were readable early, the sheet was not interactive until the whole set had settled. And the sheet carried an animating box-shadow across its full width, which has no layout cost and repaints every frame, over a large area, on a weak GPU.

One developer proposed deleting every animation in the app. It was an afternoon's work and it would measure fastest.

The other pulled the support log. Before the sheet existed, the commonest call had been "I marked the wrong batch" — and fixing one of those means a person editing rows in the database and, twice, driving to Kota. The sheet growing out of the row the teacher tapped is what tells them which batch they are in; a sheet that fades in at the centre of the screen is a new, unexplained object. Removing the motion would trade a complaint about speed for a complaint about data, and only one of those costs a day.

So the decision was between an afternoon with a known regression and a week of measurement and re-engineering, on a two-person team, with the exam season four weeks out.

They took the week. The request now starts on touch-down and the content arrives during the transition. The stagger is capped at six rows, with the rest appearing together, and every row is tappable while it is still animating. The shadow moved to a pseudo-element whose opacity is animated, which is a compositor property. They also added a reduced-motion path, because one centre head in his fifties had said the app made him feel "swimmy" and nobody had thought to write that down as a bug.

The reduced-motion work was smaller than either developer expected, because they had put their durations and curves in four custom properties rather than scattering them through the components. Redefining those four inside the media query, rather than overriding each screen, meant the behaviour reached parts of the app neither of them had opened in months — including two screens written by a contractor who had never heard of the setting.

STOP HERE

Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

Touch to readable list fell from seven hundred and eighty milliseconds to three hundred and ten on the four-year-old phone. The hanging reports

stopped within a fortnight and the paper registers came back to the office. The stagger cap turned out to matter more than the network change, because the rows a teacher looks at first are the top six and they were now all present at once. The shadow change was worth about forty milliseconds and took twenty minutes. The team keeps one old phone on the desk now, and treats a frame count taken from it as the only number that counts — the laptop figure, they say, was never wrong; it was just a measurement of a device none of their users owns.

Why did a three-hundred-millisecond animation produce a seven-hundred-and-eighty-millisecond wait?

Because animation time is added to waiting time rather than overlapping it. The request started when the transition ended, so the two ran in series instead of together. On top of that the uncapped stagger pushed the last row past nine hundred milliseconds and the sheet was not interactive until it settled, and a full-width box-shadow repainted every frame. None of those is the transition being too slow.

Deleting all the motion would have measured fastest. Why was it the wrong fix?

The transition was doing one of the three jobs that earn an animation its place: saying where a thing came from. A sheet that grows out of the row you tapped is understood as belonging to that row, and the causal reading depends on it starting within about a tenth of a second of the tap and starting from that position. Removing it would have restored a class of error — marking the wrong batch — that costs far more to fix than two hundred milliseconds costs to lose.

Why was the box-shadow expensive when it does not trigger layout?

It forces a repaint every frame, and paint cost scales with area and with how complex the content is. A shadow across a full-width sheet is a large area on a weak GPU. Putting the shadow on a pseudo-element and animating that element's opacity moves the work onto a compositor-only property, so the already-painted layer is simply blended differently.

MODULE 2 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 A panel grows from the button that opened it, but the growth starts 300 ms after the tap. What has been lost?

A. The reading that the tap caused it, which collapses past about 150 ms
B. The frame budget, because 300 ms exceeds 16.7 ms
C. Nothing; the delay makes the panel feel more substantial
D. The shared-element transition, which needs a measurement within one frame

2 A modal's opening animation takes 300 ms and its data arrives 200 ms after the press. The request fires on the press. How long is the wait?

A. 200 ms — the animation is free because it runs while the data loads
B. 500 ms — the two durations add together
C. 300 ms — the animation is now the slow part of the response
D. 100 ms — the durations cancel because the data arrives mid-animation

3 A list of twenty cards is staggered at 40 ms per item. What goes wrong, and what is the standard fix?

A. Nothing; forty milliseconds per item is inside the useful range
B. The stagger reads as simultaneous; raise the offset to 120 ms per item
C. The last card starts 760 ms late — cap the window or the count
D. The cards collide, so the stagger direction must be reversed on entry

4 Under prefers-reduced-motion, what should replace a full-screen parallax hero?

A. Nothing — parallax has no safe reduced form
B. The same parallax at half the scroll speed and half the depth
C. A cross-fade between the two layers
D. A slower version, 120 ms instead of 600 ms

5 Why does a transform animation survive a 200 ms JavaScript task while an animation of left freezes?

A. Transform animations are given higher priority by the browser's script engine
B. Left triggers a repaint, and repaints are slower than layout work
C. The browser caches the transform values and replays them after the task
D. Transform is a property of an already-painted layer, not of its contents

6 Why do many teams delay a spinner by 200 to 300 ms rather than showing it at once?

A. It gives the network time to fail before anything is drawn
B. A spinner that appears and vanishes inside 100 ms reads as broken
C. Spinners must not animate during the first frames of a page load
D. It keeps the spinner inside the 400 ms Doherty threshold for the request

MODULE 3

Shipping the motion

Getting an animation out of the tool it was made in and onto somebody else's screen: which of the five delivery technologies fits, what Lottie and SVG cannot carry, why a GIF costs twenty times what a video costs, how transparency actually works on the web, and what a developer needs in writing.

> BY THE END OF THIS MODULE YOU CAN
>
> Choose a delivery format for a given piece of motion by reasoning from whether it must respond to state, whether its content is vector or photographic, and who will edit it next — then produce that file with the right encoder settings, verify it on the target platform rather than in the authoring tool, and hand it over with a written specification covering trigger, interrupt, exit, responsive behaviour and reduced motion

---

19 · 10 min

Five ways to put motion on a screen

> THE ONE THING TO KEEP
>
> The delivery format follows from three questions — does it respond to state, is the content vector or photographic, and how often will it change — and the first of those eliminates more options than the other two combined.

Five ways to put motion on a screen, and they are not

interchangeable

Before any of the format detail, there is a decision to make, and getting it wrong costs weeks. The five options, and the question each one answers:

1. **Code — CSS, the Web Animations API, or a library.** The motion is generated at runtime from values. Infinitely adjustable, scales to any size, costs nothing to download, and requires a developer for every change.
2. **Vector data — Lottie, SVG, Rive.** The motion is authored in a design tool and exported as a file the runtime plays back. Scales without loss, small, and can be swapped without a code change.
3. **Raster frames — sprite sheet, animated WebP, APNG, GIF.** Every frame is stored as pixels. Handles anything you can draw, including textures and photographic content, at the cost of size and memory.
4. **Video.** The same as raster frames but with interframe compression, which is an enormous saving. Handles photographic and filmed content. Weak at transparency, awkward to control precisely.
5. **A real-time renderer — Canvas, WebGL, a game engine.** Anything is possible and everything is your problem.

The three questions that decide it

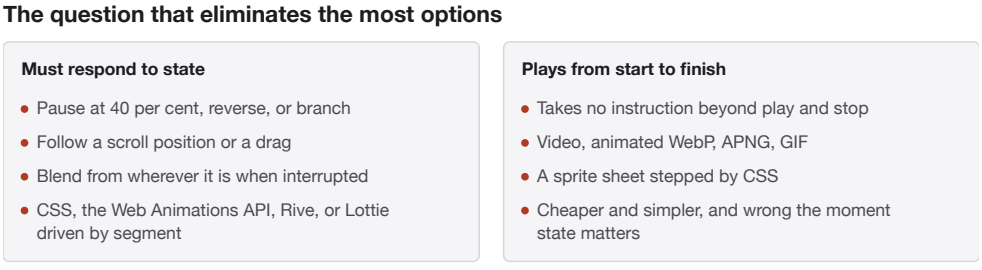
Does the motion need to respond to state? If it must pause at 40 per cent, reverse, branch depending on whether the user is signed in, or track a scroll position, you need code, Rive, or Lottie with segment control. A video or an animated image plays from start to finish and takes no instruction beyond play and stop. This is the question that eliminates the most options and is usually asked last.

Is the content vector or photographic? A logo, an icon, a character drawn in flat shapes: vector. Anything filmed, anything with a photographic texture, anything with a complex gradient mesh or a blur-heavy effect: raster or video. Trying to force photographic content into a vector format produces a file with thousands of paths that is larger and slower than the video would have been.

How many times will it change? A hero animation that marketing will revise monthly should be a file a designer can replace. A button press state that will never change should be six lines of CSS. The cost of a format is not only bytes; it is who can edit it and what they need in order to ship the change.

The sizes, roughly

For a three-second 512×512 animation of a simple illustrated scene, the orders of magnitude are consistent enough to plan with:



Three questions decide a delivery format: does the motion respond to state, is the content vector or photographic, and how often will it change. The first eliminates more options than the other two together, and it is the one usually asked last — after a week has been spent in the wrong tool.

- **Lottie JSON**: 10 to 60KB, often smaller gzipped than a single frame as PNG.
- **SVG with CSS**: 5 to 40KB for simple shapes.
- **MP4 (H.264)**: 100 to 400KB.
- **Animated WebP**: 300KB to 1.5MB.
- **APNG**: 1 to 4MB.
- **GIF**: 2 to 6MB, and it will look worse than all of the above.

The interesting gap is between the video and the raster-frame formats. It is not a small optimisation; it is a factor of five to twenty, and the reason is covered in the lesson on GIF.

The runtime costs nobody budgets for

Download size is the number everybody checks. Two others matter more on a weak device.

Decoded memory. A raster animation has to exist in memory as uncompressed pixels while it plays. A 24-frame 500×500 sprite sheet is 24 × 500 × 500 × 4 bytes — about 24MB — regardless of the fact that the PNG on disk was 300KB. On a phone with a handful of these, you are the reason the browser tab reloads.

Per-frame CPU. A Lottie file with 600 shape layers is small to download and expensive to render, because the player is rebuilding vector geometry every frame. The SVG renderer in lottie-web creates a DOM node per shape; at a few hundred nodes, animating them 60 times a second is real main-thread work. The canvas renderer is usually faster for complex files and loses crisp scaling.

A useful check: if the file is under 50KB and still stutters, the problem is shape count, not bytes.

A decision path that works

- Two states, one element, no branching → **CSS transition**.
- A loop or a multi-step sequence in code, still simple shapes → **CSS keyframes or the Web Animations API**.
- An illustrated animation a designer owns and will revise → **Lottie**.
- The same, but it must respond to hover, drag, or application state → **Rive**, or Lottie with segment playback.
- A drawn line that writes itself on, or an icon that morphs → **SVG**, with `stroke-dashoffset` or a path interpolation.
- Filmed or photographic content → **video**, with the MP4 and WebM pair.
- Filmed content that must sit over an arbitrary background → **alpha video**, and read the transparency lesson before promising it.
- Fewer than about twenty frames of hand-drawn raster art → **sprite sheet with `steps()`**.
- None of the above → **Canvas or WebGL**, and budget accordingly.

The free path

Every format here has a free authoring route. **Glaxnimate** authors and exports Lottie and SVG, open source, runs on a laptop with no GPU. **Inkscape** draws SVG and its output can be animated with CSS. **Blender** exports frame sequences for sprite sheets and renders video with alpha. **ffmpeg** converts between every raster and video format discussed. **Krita** does frame-by-frame animation and exports sequences. After Effects and Rive are named in job adverts and worth naming in a portfolio; nothing in this module requires either.

The limitation

There is no format that is small, scalable, photographic, interactive and universally supported. Every choice here trades one of those away, and most arguments about formats are two people optimising different corners of that trade without saying which one they are defending. Write down which of the three questions above your project answers "yes" to before the argument starts, and most format debates end in a minute.

BEFORE YOU MOVE ON

A team needs an illustrated character that waves when a user hovers, freezes mid-wave if the pointer leaves, and will be revised by a designer every few weeks. Which delivery route fits, and why?

A. A looping MP4 with a transparent HEVC track, because video compression makes it the smallest option and Safari's alpha support covers the hover case.

B. A sprite sheet with a steps() animation, because random access to any frame is instant and the designer can regenerate the sheet from new drawings.

C. Rive, or Lottie driven by segment playback, because the motion must respond to state and a designer needs to replace the file without a code change.

D. A Canvas or WebGL renderer written by a developer, because only a real-time renderer can interpolate from an arbitrary mid-animation position back to a rest pose when an interaction is cancelled.

20 · 10 min

What Lottie carries and what it drops

THE ONE THING TO KEEP

A Lottie file is a list of shapes with keyframed properties, so anything that is not expressible that way — effects, most expressions, motion blur, cameras — does not survive the export, and imported raster images are base64-inflated into the JSON rather than compressed.

What a Lottie file actually is

A Lottie file is JSON. Open one in a text editor and you will find a list of layers, each with a set of animated properties, each property a list of keyframes with values and bezier handles. There are no pixels in it unless an image was imported, in which case that image sits inside the JSON as a base64 string and the file size jumps accordingly.

It is produced by exporting from After Effects with the **Bodymovin** plugin, or from Glaxnimate, Haiku, or any of several open-source writers. It is played back by a runtime — `lottie-web` in a browser, `lottie-ios`, `lottie-android`, and ports for Flutter, React Native and others. The runtime reads the keyframes and redraws the shapes every frame.

That is the whole idea, and it explains both its strengths and every one of its failures. The strengths: a 40KB file that renders crisply at any size, on any platform, with the same timing, and that a designer can replace without a developer. The failures: anything that is not expressible as "shapes with animated properties" does not survive the export.

What does not export

This is the list that costs people a day each time they rediscover it.

- **Effects.** Almost all of them. Glow, drop shadow as an effect, blur, distortions, generate effects — gone, or approximated badly. Build glow with actual shapes and opacity.
- **Expressions.** Newer runtimes support a limited subset, but an expression-driven rig is not portable. Bodymovin can bake expressions into keyframes on export, which works and produces a much larger file.
- **Merge paths.** Supported in some renderers and not others, which is worse than unsupported because it works on your machine.
- **Layer styles**, **3D layers** with real depth, **cameras**, **motion blur**, **time remapping**, **adjustment layers**, **track mattes beyond the simple cases**.
- **Auto-orient**, **parenting past certain depths**, and most expression controls.

The rule the export gives you: **shape layers, solids, precomps, masks, simple track mattes, and transform properties**. Build inside that box and the file behaves everywhere.

The two renderers

`lottie-web` can draw with SVG, canvas, or HTML. The choice matters more than most people realise.

SVG renderer (the default) creates a DOM element for every shape, group and mask. Crisp at any scale, correct anti-aliasing, works with CSS. It also means a file with 400 shapes creates roughly 400 nodes that are re-laid-out every frame. Past a few hundred, this becomes visible main-thread work, and on a low-end phone it stutters.

Canvas renderer draws to a single element. Far faster on complex files. You lose per-shape CSS control and you have to redraw on resize to keep it sharp, because the canvas is a fixed pixel buffer.

The practical test: render your file, open the DevTools performance panel, and look at the main thread during playback. If you see a solid block of scripting work at 16ms intervals, switch renderer.

The sizes and the traps

A typical illustrated Lottie is 15 to 60KB. Two things blow that up:

Imported raster images. They are base64-encoded into the JSON, which adds about 33 per cent to their size on top of losing all compression benefits. A Lottie with a photograph in it is not a vector file with a small extra; it is a badly packed image file with some vectors attached. Keep images external where the runtime supports it, or do not use them.

Baked expressions and very dense keyframes. An expression sampled per frame for ten seconds at 60fps is 600 keyframes on that property. Reduce the frame rate of the export — 24 or 30 is almost always enough — and the file halves.

dotLottie (`.lottie`) is a zip containing the JSON plus any assets. It is typically 50 to 80 per cent smaller than the raw JSON for image-bearing files and has a cleaner story for multiple animations in one bundle. Support is now good across the official runtimes.

Controlling it

This is where Lottie beats a video. The runtime exposes the timeline:

```
const anim = lottie.loadAnimation({
  container: el, renderer: 'svg', loop: false,
  autoplay: false, path: '/motion/checkout.json'
});
anim.playSegments([0, 40], true);   // the "loading" portion
anim.setSpeed(1.2);
anim.goToAndStop(38, true);          // frame-accurate hold
```

Marking sections of a single file and playing segments is how you build a state-driven animation without a separate file per state. Name your frame ranges in the handover, because a developer cannot find them by looking.

Honouring reduced motion

A looping Lottie is continuous motion and belongs inside the reduced-motion rules from the previous module. The runtime cannot know; you have to check the media query and either call `goToAndStop` on a representative frame or not load the animation at all. Shipping a static poster frame as an SVG for that case is a five-minute job that almost nobody does.

The free path

- **Glaxnimate** — open source, authors and exports Lottie directly, no After Effects required.
- **python-lottie** — a library that reads, writes and converts Lottie files, useful for batch edits and for stripping unsupported features.
- **LottieFiles** has a free tier with a preview and optimiser; the file format itself is open and none of the tooling is mandatory.
- **Blender** with grease pencil for authoring, exported through an SVG sequence, is a viable if awkward free route for frame-by-frame vector work.

The limitation

Lottie's compatibility story is not one thing. The web, iOS and Android runtimes have historically differed in exactly which features they implement, and a file that looks right in one can be subtly wrong in another — a mask that clips differently, a trim path that runs backwards, a gradient that bands. There is no conformance suite that settles it.

So the only safe process is to preview the exported file on every target platform before sign-off, not in the design tool. The design tool is not the renderer, and the difference between them is where the day goes.

BEFORE YOU MOVE ON

A Lottie file is 1.4MB and stutters on a mid-range phone. The designer insists the composition is simple. What are the two most likely causes, given how the format works?

A. A raster image has been embedded as base64 inside the JSON, and the SVG renderer is creating a DOM node per shape that is re-laid-out every frame.
B. The animation was exported at 60fps instead of 24, which quadruples the rendering cost per second because the runtime must interpolate four times as many intermediate values.
C. Expressions in the composition are being evaluated live by the runtime on every frame, which is the main source of per-frame cost in any Lottie file.
D. The file is using the canvas renderer, which rasterises the entire composition at device pixel ratio on every frame and is therefore always slower than the SVG renderer for complex artwork.

21 · 10 min

Animating a drawing that is also a document

THE ONE THING TO KEEP

SVG elements sit in the DOM, so CSS and JavaScript animate them directly — and the two techniques worth memorising are stroke-dashoffset for a self-drawing line, normalised with pathLength, and transform-box: fill-box to stop a rotation swinging across the viewBox.

SVG is a document, which is the whole trick

An SVG is markup. Every shape in it is an element with attributes, sitting in the same DOM as the rest of your page. That means the entire CSS and JavaScript animation toolkit applies to it directly — no runtime, no plugin, no export step. A designer draws a path in Inkscape, you give it a class, and CSS animates it.

This is why SVG remains the right answer for icons, logos, diagrams, line illustrations and data graphics, despite Lottie existing.

The line that draws itself

The single most useful SVG technique, and it looks like magic until you know the mechanism.

A stroked path can have a dash pattern: `stroke-dasharray: 10 5` means 10 units of ink, 5 of gap, repeating. Set the dash length equal to the *entire length of the path* and the gap equal to it too, and you have one dash covering the whole line and one gap after it. Then `stroke-dashoffset` slides that pattern along the path. Offset it by the full length and the ink is pushed off the end — the line is invisible. Animate the offset back to zero and the ink slides into place, which reads as the line being drawn.

```
.signature path {
  stroke-dasharray: 1000;
  stroke-dashoffset: 1000;
  animation: draw 2s ease-out forwards;
}
@keyframes draw { to { stroke-dashoffset: 0; } }
```

The number 1000 has to be at least the path's length. Get the real one with `path.getTotalLength()` in JavaScript — or, much better, put `pathLength="100"` on the path itself. That attribute tells SVG to pretend the path is 100 units long for all dash arithmetic, whatever its real geometry, so your CSS can use 100 everywhere and never needs measuring. Almost nobody knows about `pathLength` and it removes the one awkward step in the technique.

Morphing one shape into another

Two routes, with a sharp limitation on both.

Animating the `d` attribute. Modern browsers can interpolate a path's `d` between two values, in CSS or the Web Animations API. It only works when both paths have the same number of segments, of the same types, in the same

order. A four-point square morphs to a four-point diamond; a square does not morph to a circle unless you build the circle out of the same four cubic segments.

A shape library. GSAP's MorphSVG and the open-source `flubber` both solve the mismatch by resampling one path to match the other's point count. This is the honest way to morph arbitrary shapes, and the results are only as good as the correspondence the algorithm guesses.

The practical workflow: draw both shapes from the same base in Inkscape or Figma so the point counts already match, and the native route just works.

SMIL, and whether to use it

SVG has its own animation elements — `<animate>`, `<animateTransform>`, `<animateMotion>` — known as SMIL. Chrome announced their deprecation in 2015 and reversed it; they work today in Chrome, Firefox and Safari and are not going away in practice.

They have one genuine advantage: an SVG file animated with SMIL animates when opened on its own, in an `<img>` tag, or as a CSS background image, where CSS-in-the-page cannot reach it. If you need a self-contained animated SVG, SMIL is the only way inside the format.

Two reasons to prefer CSS or JavaScript in a page: SMIL does not honour `prefers-reduced-motion` without extra work, and its timing model is a separate thing to learn for no gain when you already have the platform's.

`<animateMotion>` deserves a mention: it moves an element along an arbitrary path, which CSS only gained recently through `offset-path`. If you need something to follow a curve, `offset-path: path("M ...")` with an animated `offset-distance` is the modern equivalent and it is compositor-friendly.

The transform trap

The most common SVG animation bug: an element rotates around the wrong point.

CSS `transform-origin` on an HTML element defaults to the element's centre. On an SVG element it is resolved against the element's own bounding box in user units, and `transform-box` decides which box. The fix is one line:

```
.gear { transform-box: fill-box; transform-origin: center; }
```

Without it, a gear in the middle of a 1000-unit viewBox rotates around the viewBox origin and swings across the screen. Most "my SVG animation flies off" questions are this.

Size and performance

SVG is text, so it gzips extremely well — a 30KB SVG is often 4KB on the wire. Keep it small by cleaning the exported file: Illustrator and Figma both emit enormous amounts of redundant markup. **SVGO** removes it, typically cutting 40 to 70 per cent, and it is free.

Be careful with filters. `feGaussianBlur` and friends are rasterised per frame and are among the slowest things a browser does. An animated blur on a large SVG will drop frames on hardware where the same animation with no filter runs cleanly.

The free path

- **Inkscape** — a full SVG editor, open source, and unusually it edits the actual SVG rather than an internal model it exports from.
- **SVGO** — MIT-licensed optimiser, runs as a CLI or a build step.
- **Glaxnimate** — draws and animates SVG and Lottie together.
- **flubber** — MIT, path interpolation for morphs.
- **GSAP** — free for all uses now, including MorphSVG and DrawSVG.

The limitation

SVG animation scales badly with element count in exactly the way Lottie's SVG renderer does, and for the same reason: each shape is a DOM node and the browser does style and layout work on all of them. A hundred animated

nodes is comfortable; a thousand is not, and the fix is to move to canvas, at which point you have given up the thing that made SVG attractive.

It is also genuinely bad at anything photographic. An SVG containing a traced photograph is thousands of paths, larger than the JPEG, and slow. The format is for drawings, and it is excellent at drawings.

BEFORE YOU MOVE ON

Why does setting pathLength="100" on an SVG path make a self-drawing line animation easier to write?

A. It resamples the path to exactly 100 points, so any two paths with the attribute can be morphed into each other without a resampling library.

B. It caps the animation at 100 discrete steps, which prevents the browser from interpolating sub-pixel dash offsets and removes the shimmer on diagonal strokes.

C. All dash arithmetic on that path is then expressed in units of a hundredth of its length, so the CSS can use 100 without measuring the real geometry.

D. It tells the browser to precompute the path's geometry at parse time and cache it, so getTotalLength no longer forces a synchronous layout when it is called from JavaScript during the animation's first frame.

22 · 9 min

Why a GIF costs twenty times what a video costs

THE ONE THING TO KEEP

GIF stores each frame as a palette of 256 colours with one bit of transparency and no motion compensation between frames, so a clip that a video codec describes as block movement has to be stored almost in full every frame.

Why the same three seconds is 4MB as a GIF and 180KB as a video

GIF was finalised in 1989. It has three properties that make it unsuitable for the job it is most often used for, and all three follow from that date.

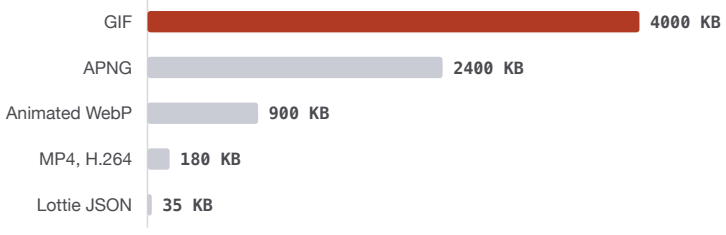
256 colours, chosen from a palette, per frame. The format stores an index into a colour table, not a colour. Anything with a gradient, a soft shadow, film grain or skin tone has to be quantised down and then dithered, which is why GIFs of real footage look speckled and banded. The dithering also destroys the compressor's job, because dither noise is by definition unpredictable.

One bit of transparency. A pixel is either fully opaque or fully transparent. There is no partial alpha, so an anti-aliased edge over an unknown background gets a hard, jagged fringe in whatever colour the file was matted against. This is the "why does my logo have a white halo" problem, and it cannot be fixed in GIF.

No motion compensation. Modern video codecs describe a frame as "the previous frame, with this block moved 14 pixels left". GIF can only say "these pixels changed" — a rectangular difference region, compressed with LZW. Pan the camera and every pixel changes, so every frame is stored nearly in full.

Put together: a 640×360 three-second clip that an H.264 encoder stores in about 180KB will commonly be 3 to 6MB as a GIF, and will look worse. That is a factor of twenty to thirty, paid on every page load, by every visitor.

The same three seconds, five ways



GIF stores 256 palette colours per frame, one bit of transparency, and no motion compensation at all, so a pan means storing nearly every pixel again. The factor of twenty against the video is paid on every page load by every visitor, and the GIF is the one that looks worst. Where the destination demands a GIF anyway, generate the palette from the whole clip rather than one frame.

Replace it with a video element

The replacement is a muted, looping, inline video. Autoplay is permitted by every modern browser under the same conditions: the video must be muted and, on iOS, carry `playsinline`.

```
<video autoplay loop muted playsinline
       poster="/motion/preview.jpg"
       width="640" height="360">
  <source src="/motion/demo.webm" type="video/webm">
  <source src="/motion/demo.mp4"  type="video/mp4">
</video>
```

Two sources because VP9 in WebM is usually 20 to 40 per cent smaller than H.264 at the same quality and Safari's support has historically lagged; the MP4 is the guaranteed fallback. Always set `width` and `height`, or the page will shift when the video loads.

Three things this buys you beyond size: the user can pause it, you can respect reduced motion by not autoplaying, and the browser will not decode it while it is off-screen if you add `preload="none"` and start it when it scrolls into view.

If you are stuck with GIF

Sometimes the destination demands it — an email client, a chat platform, a forum that strips video tags. The quality difference between a careless GIF and a careful one is large, and it comes down to the palette.

The default in most converters is a single palette computed from one frame. The better approach generates a palette from the whole clip, then applies it with dithering:

```
ffmpeg -i in.mp4 -vf "fps=15,scale=480:-1:flags=lanczos,palet-
tegen=stats_mode=diff" palette.png
ffmpeg -i in.mp4 -i palette.png -lavfi
"fps=15,scale=480:-1:flags=lanczos,paletteuse=dither=bayer:bay-
er_scale=3" out.gif
```

`stats_mode=diff` weights the palette toward the parts of the frame that move, which is where the eye is. Bayer dithering produces a regular pattern that LZW compresses far better than the default error-diffusion dither, usually cutting the file by a third for a small visible cost.

The other two levers, in order of effect: **drop the frame rate** to 12 to 15, and **drop the resolution**. Both cut size roughly linearly. Trimming a second off the clip is worth more than any encoder setting.

The middle options

Between GIF and video there are two raster formats worth knowing.

Animated WebP — full 8-bit alpha, far better compression than GIF, supported everywhere that matters now. Typically 30 to 50 per cent of the GIF size at better quality. It is the correct answer when you need an animated *image* — something that works in an `<img>` tag, as a CSS background, or anywhere a video element is not allowed.

APNG — full alpha, lossless, and very large. Useful when you need exact pixels and transparency and size is genuinely not a concern, which is rare.

Both are images, so neither can be paused by the user and neither honours reduced motion on its own. That is a real accessibility cost and the reason a video element is preferable where it is allowed.

The free path

ffmpeg does all of it — the conversion, the palette work, the WebP and APNG output, the frame-rate and scale changes — and it is the tool the paid converters are usually wrapping. **Gifski** is an open-source encoder that produces noticeably better GIFs than the default by using a per-frame palette and a smarter quantiser, at the cost of speed. **ImageMagick** handles APNG assembly. None of this needs a licence.

The limitation

GIF persists for a reason that has nothing to do with quality: it is universally accepted. Email clients, older forums, messaging apps and some content management systems will take a GIF and nothing else, and a technically superior file that the destination rejects is worth nothing.

So the honest advice is not "never use GIF". It is: never use GIF on a web page you control, always use it where the destination demands it, and when you do, spend the two minutes on the palette because the difference is visible and the file is often a third of the size.

BEFORE YOU MOVE ON

Converting a screen recording to GIF, a team finds that switching the dither from error-diffusion to Bayer cuts the file by about a third with only a small visible cost. Why does the dithering method change the file size at all?

A. Bayer dithering uses fewer palette entries, and GIF stores one byte per distinct colour used in each frame, so a smaller palette is directly a smaller file.

B. Error-diffusion scatters noise unpredictably across the frame, while Bayer produces a regular repeating pattern that the LZW compressor can encode efficiently.

C. Bayer dithering is applied once to the whole clip rather than per frame, so the difference regions between frames contain no dither changes and compress to almost nothing.

D. Error-diffusion increases the effective colour depth of the image beyond the 256-colour limit, forcing the encoder to split each frame into several sub-images with their own local colour tables and their own headers.

23 · 10 min

Transparent video, and why it is two files

THE ONE THING TO KEEP

No single video file carries alpha everywhere: VP9 in WebM covers Chrome and Firefox, HEVC with the hvc1 tag covers Safari, and the luma-matte workaround doubles the frame width and requires WebGL — so the first question is whether the background is genuinely arbitrary.

The request that sounds simple

"Can we have the animated logo float over the photograph, with the background showing through?" This is a transparent video, and it is the single most annoying request in web motion, because the format support is genuinely poor and the workarounds all have costs.

The reason is historical. Video codecs were built for broadcast and cinema, where there is no background behind the frame. Alpha was bolted on afterwards, by different people, in incompatible ways.

What actually exists

VP9 with alpha, in WebM. The pixel format is `yuva420p` — the extra `a` is the alpha plane. Supported in Chrome, Firefox and Edge. Not in Safari. Encode with:

```
ffmpeg -i in.mov -c:v libvpx-vp9 -pix_fmt yuva420p -b:v 0 -crf
32 out.webm
```

Your source has to carry alpha to begin with: ProRes 4444, QuickTime Animation, or a PNG sequence. Export from your editor with "RGB + Alpha" and a codec that supports it, or the alpha never existed.

HEVC with alpha, in MP4. Safari only, on macOS and iOS. Apple documents the requirement precisely: the video track must use the `hvc1` tag rather than `hev1`, and the alpha is carried as an auxiliary layer. It is encodable from Apple's own tools and from ffmpeg builds with `videotoolbox`, and it is not encodable everywhere.

So the working answer on the web today is **two files and a source-order fallback**: HEVC-alpha MP4 first for Safari, VP9-alpha WebM second for everyone else. Both have to be produced, both have to be tested, and if either is missing a large share of users see a black box.

Animated WebP and APNG both carry real 8-bit alpha and are supported everywhere. They are images, so they are far larger than video and cannot be paused, but for a short loop under a couple of seconds they are frequently the pragmatic answer.

The luma-matte trick

When the two-file route is unacceptable, the standard workaround is to put the colour and the transparency in the same ordinary video, side by side: the

image on the left half of the frame, a black-and-white matte of its alpha on the right. One file, any codec, universal support.

At playback, a small WebGL shader — or a canvas loop, more slowly — reads both halves each frame and writes the matte's luminance into the output's alpha channel. Several open-source implementations exist; the technique is old and comes from broadcast, where a "fill and key" pair has always been two signals.

The costs are real and worth stating: the frame is twice as wide, so you are paying for more pixels; the matte is compressed lossily along with everything else, so soft edges pick up ringing; and you now require WebGL, which excludes anyone with it disabled and adds a code path to maintain.

The option people forget

Ask whether the background is genuinely arbitrary.

Very often the "transparent" video is going over one known colour, or one known photograph. If so, composite it in the edit, export an ordinary opaque MP4, and the whole problem disappears along with 60 per cent of the file size. The number of transparent-video requests that survive this question is smaller than you would expect, and asking it takes ten seconds.

If the content is vector — a logo, an illustrated character, a line animation — the right answer is almost always **Lottie or SVG**, which have transparency for free because they are drawings rather than rectangles of pixels. Transparent video is for filmed or rendered raster content specifically.

Checking that alpha survived

The failure mode is silent: your file plays, the background is black, and you cannot tell whether the alpha is missing or being ignored. Check the file rather than guessing.

```
ffprobe -v error -select_streams v:0 \
  -show_entries stream=pix_fmt,codec_name -of default=nw=1
out.webm
```

If `pix_fmt` is `yuv420p` rather than `yuva420p`, the alpha was discarded at encode — usually because the source did not have it, or because a filter in the chain dropped it. `-vf format=yuva420p` early in the chain fixes a surprising number of these.

The other common cause is **premultiplied versus straight alpha**, covered properly in the compositing module. If your edges have a dark or light fringe rather than being cleanly transparent, the alpha survived and is being interpreted with the wrong convention.

Performance notes

An alpha video is decoded with an extra plane, so it costs more memory and more decode time than the same clip opaque — roughly 30 to 50 per cent in practice. On a low-end phone, a full-screen alpha video at 60fps is not realistic, and the fallback should be a static image rather than a stuttering one.

Set `poster` on the video element so that something sensible is visible before the first frame decodes, and remember that a looping decorative video is continuous motion: it belongs behind the reduced-motion check.

The free path

ffmpeg encodes VP9 with alpha, converts sequences, builds side-by-side mattes with the `hstack` filter, and probes the result — all free. **Blender** renders with a transparent film and exports ProRes 4444 or a PNG sequence. **DaVinci Resolve**'s free edition exports alpha in several formats. **Kdenlive** and **Shotcut** both handle alpha sources. The gap in the free toolchain is HEVC-with-alpha encoding, which in practice requires an Apple machine.

The limitation

There is no single file today that carries video with alpha and plays everywhere. Every route above is a compromise: two files and double the testing, an image format and a size penalty, or a shader and a dependency. Anybody who tells you otherwise has tested on one browser.

This will probably improve — AVIF sequences carry alpha and support is growing — but treat any claim of universal support as something to verify on a real Safari and a real Android device before it goes in a plan.

BEFORE YOU MOVE ON

An alpha video plays but the transparent areas render black. ffprobe reports pix_fmt as yuv420p. What does that tell you?

A. The alpha is present but premultiplied, and the player is interpreting it as straight alpha, which multiplies the colour by zero in the transparent regions.

B. The browser is falling back to the MP4 source because the WebM failed a codec check, so you are seeing an opaque file that was never meant to carry alpha.

C. The alpha plane was discarded during encoding, because a pixel format without the trailing 'a' has no channel to store it in.

D. The alpha plane is present but stored in a separate auxiliary track that ffprobe does not list under the primary video stream, which is the normal arrangement for VP9 and means the file is correct and the fault is in the CSS compositing of the video element against its parent background.

24 · 9 min

Frames in a grid, and the memory they cost

THE ONE THING TO KEEP

A sprite sheet gives instant random access to any frame with steps() timing, but its cost is decoded memory — width times height times four bytes, regardless of how well the PNG compressed — which is why a 4096-square sheet can cost 67MB and evict everything else on the page.

When frames are the only honest representation

Some animation is not describable as shapes with animated properties. Hand-drawn frame-by-frame work, a painted effect, a cel-animated character with squash that no rig would produce, a rendered 3D loop with real lighting. For these, the animation *is* the sequence of pictures, and the format that admits this is a sprite sheet: every frame laid out in a grid in one image, shown one at a time.

It sounds primitive. It is also the fastest possible playback, because there is no decode work per frame and no geometry to rebuild — the browser or GPU is showing a different window onto an image it already holds.

The CSS, which is shorter than people expect

One image, 24 frames in a row, each 200px wide:

```
.flame {
  width: 200px; height: 200px;
  background: url(/motion/flame.png) 0 0 / 4800px 200px;
  animation: play 1s steps(24) infinite;
}
@keyframes play { to { background-position: -4800px 0; } }
```

The whole technique is in `steps(24)`. A normal timing function interpolates smoothly, which would slide the image continuously and show two half-frames at once. `steps()` holds each value and jumps, so the background position lands exactly on frame boundaries.

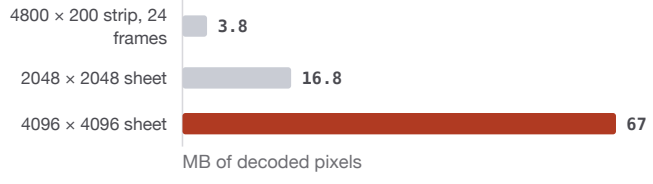
The end position is the *full* sheet width, not the width minus one frame. With `steps(24)`, the animation is divided into 24 equal holds, and the last hold shows frame 24 — the step after that would be the wrap. Getting this off by one is the classic bug: a flash of blank at the end of every loop means you subtracted a frame you should not have.

For a grid rather than a strip, animate `background-position` in two keyframe tracks, or use two nested animations — one stepping across columns fast, one stepping down rows slowly.

The number that matters is memory, not file size

A PNG sprite sheet might be 400KB on disk. That is not what it costs.

What a sprite sheet costs in memory, whatever the PNG weighed



Width times height times four bytes, held for as long as the element exists, regardless of how well the file compressed on disk. This is the usual way a sprite-sheet animation kills a phone, and the symptom is not a slow animation — it is the tab reloading, or unrelated images flickering as they are evicted and decoded again.

To display it, the browser decodes it to uncompressed pixels: width × height × 4 bytes. A 4800×200 strip is 3.8MB in memory. A 4096×4096 sheet — the size people reach for when they have a hundred frames — is 67MB, and that is per sheet, held for as long as the element exists.

This is the single most common way a sprite-sheet animation kills a phone. The symptoms are not a slow animation; they are the whole tab reloading, or other images being evicted and re-decoded, which looks like unrelated flickering elsewhere on the page.

Constraints that follow:

- **Cap the total pixel area.** Many mobile GPUs cap a single texture at 4096×4096; some older ones at 2048. Exceed it and the image may silently fail to render, or be downscaled.
- **Keep the frame count honest.** Twelve frames looping at 12fps reads as animation. Ninety-six frames at 60fps costs eight times the memory for a difference most viewers will not name.
- **Use WebP for the sheet.** Lossy WebP with alpha is typically a third of the PNG size on disk. The decoded memory is identical, so this only helps download, but download is still worth a third.

Choosing a frame rate deliberately

Frame-by-frame animation does not have to run at the display rate. Traditional animation has always worked "on twos" — a new drawing every second frame, so 12 drawings a second at 24fps — and it reads as smooth for most subjects while halving the work and the memory.

Fast action sometimes needs ones. Slow drifting motion can go on threes or fours. Choosing per shot rather than globally is how hand-drawn animation has always been budgeted, and it applies directly here: the frame count is your file size and your memory.

Where sprite sheets are still the right answer

- **Short, looping, drawn effects** — a flame, a sparkle, a loading character.
- **Scroll-driven sequences** where the user controls the frame. Showing frame 37 of 60 as a background-position is instant; seeking a video to an exact frame is not, because a video decoder must decode from the last keyframe forward.
- **Anything inside a game engine**, where texture atlases are the native idiom.
- **Interfaces where the animation must not depend on a runtime library.**

That second case is worth underlining. The product-page effect where an object rotates as you scroll is almost always a frame sequence, not a video, precisely because video seeking is unreliable and a sprite sheet's random access is free.

Building one

```
# frames out of a source clip, 12 per second
ffmpeg -i source.mov -vf fps=12,scale=200:-1 frames/f_%03d.png

# assemble a single row
ffmpeg -i frames/f_%03d.png -filter_complex tile=24x1 sheet.png

# and a smaller version for download
cwebp -q 80 sheet.png -o sheet.webp
```

`tile=6x4` gives a grid instead. **ImageMagick**'s `montage` does the same job with more control over padding, and padding matters: leave none, or GPU texture filtering will bleed the neighbouring frame's edge pixels into yours.

The free path

Everything above is free. **ffmpeg** extracts and tiles. **ImageMagick** assembles and trims. **Krita** and **OpenToonz** are both open source and both do genuine frame-by-frame animation with onion skinning — OpenToonz is the tool Studio Ghibli's in-house system became. **Blender**'s grease pencil does drawn animation inside a 3D scene and renders sequences directly. **TexturePacker** is the paid standard for atlases and **free-tex-packer** does the same job open source.

The limitation

Sprite sheets do not scale — in either sense. You cannot display one larger than it was drawn without softness, so a responsive layout means either shipping the largest size to everyone or shipping several sheets. And the cost grows linearly with every frame and every pixel, with no compression across frames at all, which is exactly the weakness that video codecs exist to fix.

The moment your sequence is longer than a couple of seconds, or larger than a few hundred pixels square, the arithmetic turns against you and a video becomes the correct format — with all the seeking limitations that brings back.

BEFORE YOU MOVE ON

A 4096x4096 PNG sprite sheet compresses to 800KB on disk. A tester reports that on an older phone, opening the page causes unrelated images elsewhere to flicker and reload. What is the mechanism?

A. The 800KB download is saturating the connection, so other images are being fetched after a timeout and re-requested.

B. The steps() timing function forces a repaint of the whole document on each step, and repaints evict other decoded images from the browser's cache.

C. The sheet occupies about 67MB once decoded, and under memory pressure the browser discards other decoded images and re-decodes them when they are needed.

D. The sheet exceeds the maximum texture size on that device's GPU, so the compositor is refusing to upload it as a single texture and is instead re-uploading tiles of it on every frame, which starves the texture memory that other images are using.

25 · 10 min

Motion that responds to state

THE ONE THING TO KEEP

Interactive motion is a state machine, and the hard part is interruption — a tool like Rive blends from the current position automatically, CSS transitions get this free from the browser, and a play-a-clip approach gets it not at all.

Animation that has to know what is happening

Everything so far plays. A Lottie plays, a video plays, a sprite sheet plays. Real interface motion frequently has to *respond*: a button that is idle, then hovered, then pressed, then loading, then successful, with a sensible transition between any two of those, including the ones nobody planned for — pressed straight to error, hover interrupted halfway to idle.

Building that with play-a-file thinking produces a combinatorial mess. Five states means up to twenty transitions, and a separate clip for each is unmaintainable.

The state machine

The tool for this is a finite state machine: a set of named states, a set of inputs, and rules for which input moves you from which state to which other state. The animation for each state, and each transition between states, is authored once.

Rive is the product built around this idea. You draw and animate in the editor, then define a state machine in the same file: states, transitions, conditions, and inputs that the host application can set. The runtime loads a `.riv` file and you set inputs from code:

```
const input = stateMachine.findInput('isLoading');
input.value = true;   // the runtime handles the blend
```

The important part is what the runtime does between states. It blends — if you are 40 per cent through the hover animation when the pointer leaves, it does not snap to idle, it interpolates from where it is. That interruption handling is the thing that is tedious to write by hand and is the main reason to use a tool for it.

Rive files are small — commonly under 20KB for a complex interactive component, because it is vector data plus a state graph — and the runtimes cover web, iOS, Android, Flutter and several engines.

Be clear-eyed about the trade: this is a proprietary format with a commercial product behind it. The runtimes are open source and permissively licensed, and the editor has a free tier with limits that change. A file authored in Rive is editable in Rive.

Doing it without a new dependency

The same architecture works with tools you already have, and for many components it is the better answer.

Lottie with named segments. Author one file with the states laid out along the timeline — frames 0–20 idle loop, 21–45 hover-in, 46–70 press, 71–110 success — and drive it from code with `playSegments`. You write the state machine yourself, in about forty lines. You do not get automatic blending on interruption, so design the segments to be interruptible: return to a common pose at the boundaries.

CSS with a data attribute. For simple components this is the whole thing. Put the state on the element as `data-state="loading"`, write a rule per state, and let transitions handle the in-between. The browser's transition engine already blends from the current computed value, which means interruption handling is free — this is the one place the platform gives you for nothing what Rive charges for.

```
.btn { transition: transform 180ms var(--ease-move); }
.btn[data-state="pressed"] { transform: scale(0.96); }
.btn[data-state="loading"] { transform: scale(1); }
```

XState or a hand-rolled reducer for the logic, with any of the above for the visuals. Keeping the state machine in your application code rather than inside an animation file has a real advantage: it is testable, it is in version control next to the component, and a developer can read it without opening a design tool.

The question to decide it

Does the *designer* need to own the state machine, or does the *developer*?

If the motion is complex, the states are visual, and the person iterating on it is a designer who should not be opening a pull request for every timing tweak, a tool like Rive earns its place. If the states are really application states — loading, error, empty, authorised — they belong in code, because that is where the truth about them already lives, and duplicating them into an animation file guarantees the two will disagree.

The failure mode nobody warns about is the third option: the state machine exists in both places. The animation file knows about `isLoading` and so does the component, and six months later one of them has a state the other does not.

Scroll as an input

A common case of interactive motion is scroll position driving an animation. It used to require a JavaScript listener recalculating on every scroll event, which is main-thread work at exactly the moment the main thread is busiest.

CSS scroll-driven animations now express this declaratively:

```
.progress {
  animation: grow linear;
  animation-timeline: scroll(root block);
}
```

The browser runs it on the compositor, so it stays smooth while scripts run. Where support is missing it degrades to the animation not running, so design the default state to be correct.

The free path

- **CSS transitions plus a state attribute** — free, native, and handles interruption correctly by default.
- **Lottie** with segment control — free format, free runtimes, free authoring in Glaxnimate.
- **XState** — MIT-licensed state machines, with a free visualiser that draws the graph.
- **Rive** runtimes are open source; the editor has a free tier, and it is worth knowing because job adverts have started naming it.
- **Blender** with drivers for non-runtime interactive prototyping.

The limitation

A state machine makes the transitions explicit, which is its value and also its cost: you now have to decide what happens for every pair, including the fifteen nobody will ever hit. Teams routinely spend a week on transition polish for states that occur in 0.1 per cent of sessions.

And an interactive animation cannot be checked by watching it once. It needs a test that drives it through the awkward sequences — pressed during loading, hover during exit, two inputs changing on the same frame — because those are where the blending produces something nobody designed.

BEFORE YOU MOVE ON

A component's hover, press, loading and error visuals are driven by a state machine authored inside an animation file, while the application separately tracks the same four states in its own code. Six months later the two disagree. What was the structural mistake?

A. The state machine belonged in one place; duplicating application states into an animation file guarantees the two definitions will drift.

B. The animation file should have used named timeline segments instead of a state machine, because segment playback is the only approach that keeps the source of truth in code.

C. The states were not covered by tests, and any state machine without a test that drives every transition pair will eventually diverge from its specification.

D. Loading and error are application states rather than visual states, so they should have been kept out of the animation entirely and represented by swapping between two separate animation files, each with its own internal state machine covering only hover and press.

26 · 9 min

Send the developer a file, not a video

THE ONE THING TO KEEP

If a developer cannot recolour it, pause it, or place it on a dark background, you exported the wrong format.

Send the developer a file, not a video

A designer animates a neat loading spinner in After Effects, exports an MP4, and sends it to the developer. It cannot be used, and the reasons are worth listing because they are the whole argument.

A video has a fixed size, so it blurs when scaled. It has a background, so it cannot sit on a coloured surface. Its colours are baked in, so it cannot follow a dark theme. It cannot be paused at a meaningful state, cannot be looped seamlessly at an arbitrary point, and cannot be driven by what the user is doing. And a three-second 1080p MP4 is a few hundred kilobytes where the same animation described as vectors is around ten.

The animation was not wrong. The file was.

The formats, and what each one is for

Lottie. A JSON file that describes vector shapes and keyframes, played live by a small runtime on web, iOS and Android. It scales to any size, can be recoloured in code, can be played, paused, reversed and scrubbed, and is tiny. Exported from After Effects with the Bodymovin / LottieFiles plugin. This is the standard answer for icons, illustrations, loaders and empty-state animations.

CSS and SVG animation. For anything simple — a transition, a spinner, a hover state, a line drawing itself with `stroke-dasharray`. No library, no file, nothing to load. Build the SVG in **Inkscape** (free) or **Figma**'s free tier and animate it in the stylesheet.

Rive. A competing runtime built around state machines, so the motion responds to input rather than playing a fixed clip. Worth knowing about when the animation needs to react.

Video. Correct for anything photographic, anything long, and anything using real footage. Not a defeat, just a different job.

Animated WebP or APNG. When it must be raster and must loop without a player.

Two things that decide whether it works

The performance detail that decides whether it stutters

Animate `transform` and `opacity`. Those two can usually be handled by the browser's compositor without recalculating the page layout or repainting

pixels.

Animating `width`, `height`, `top`, `left`, `margin` or `box-shadow` forces the browser to redo layout or paint on every single frame. On your laptop it looks fine. On a four-year-old Android phone it drops to something visibly juddery. This is the most common cause of web animation that works in the demo and fails on real devices.

So: to move something, use `transform: translateX()`, not `left`. To resize it, use `transform: scale()`, not `width`.

Lottie's honest limits

Lottie exports shape layers and their keyframes. It is not a renderer for After Effects.

Most effects do not export — glows, blurs, distortions, layer styles, 3D layers, many blend modes, and expressions beyond a supported subset. They vanish silently. The export succeeds, the file plays, and the glow is simply not there. If you embed a raster image to work around this, the file size goes from kilobytes to hundreds of kilobytes and you have lost the main advantage.

The way to work with this is to keep the exporter's supported-feature list open while you animate, not to animate freely and export at the end. Build the glow as a shape with a gradient rather than as an effect. And test in the actual player on an actual device — the preview on the authoring site is not the same renderer as the one in the app.

Handover hygiene, and reduced motion

What to send

- Name your layers. The developer will see those names.
- Deliver at the size it will be used, or design it to be size-independent.
- State the frame rate and the duration in the filename or a note.
- Provide a static first frame as a fallback, for when the player fails or the animation is switched off.
- Say explicitly whether it loops, plays once, or waits for an event.

Reduced motion is not optional

A meaningful number of people turn on a system setting called "reduce motion" because animation makes them physically unwell. Vestibular disorders are real, and the triggers are large movements, parallax, and anything that zooms.

In CSS, wrap non-essential motion so it degrades to a fade:

```
@media (prefers-reduced-motion: reduce) {
  * { animation-duration: 0.01ms !important; transition-dura-
tion: 0.01ms !important; }
}
```

Better than that blanket version is to write reduced alternatives deliberately for the animations that matter. Lottie players can be gated on the same media query. Native apps expose an equivalent flag.

This is accessibility, and it costs you about five minutes.

One thing to try

Take an animation you have already delivered as a video. Ask whether a developer could change its colour, pause it halfway, or place it on a dark background. If the answer is no to all three, it was the wrong file, and rebuilding it as vectors is a short afternoon.

BEFORE YOU MOVE ON

A designer builds an icon animation in After Effects using a glow effect and a track matte, exports it as a Lottie JSON, and the developer reports that the glow is simply missing in the app. Why?

A. The JSON was exported at a frame rate the player does not support, so some layers are being skipped.
B. The Lottie player in the app is out of date and needs upgrading to a version that renders effects.
C. The phone's GPU cannot render the glow at the required frame rate, so the player is disabling it.
D. Lottie describes vector shapes and their keyframes; most After Effects effects are outside the format and are dropped silently at export, so the glow has to be built as a shape or shipped as raster.

27 · 9 min

The specification a developer can build from

THE ONE THING TO KEEP

A recording shows one playthrough of the happy path, so the specification has to supply what it cannot — trigger, interrupt behaviour, exit, loop count, responsive numbers and reduced-motion substitution — and it should be reviewed at quarter speed on the oldest device available.

The document that stops the fourteen questions

A designer sends a video of the intended animation. A developer builds it. It is wrong in six ways, none of which were discussed, all of which were invisible in the video. Three rounds of review later, it ships, subtly different from the design and impossible to reproduce elsewhere.

The fix is a written specification. It is not bureaucracy; it is the shortest route to the thing being right, and it takes about fifteen minutes to write.

What a developer cannot get from a video

A video shows one playthrough of one happy path. It does not show:

- **What triggers it.** Page load? Element entering the viewport? A click? A successful response? "On load" and "on scroll into view" look identical in a recording and are different code.
- **What interrupts it.** The user clicks again while it is running. The user navigates away. The data arrives early.
- **The exit.** Almost every recorded animation shows the entrance and stops. What happens when the element leaves is a separate decision and is usually forgotten until QA.
- **Whether it loops**, how many times, and whether it rests on a final pose.
- **What happens at other sizes.** The same animation on a 360px phone and a 1440px desktop is rarely the same numbers.
- **The reduced-motion behaviour.**
- **Which properties are moving.** Is that growth a scale or a width? They look similar and behave differently — a scale distorts the text inside it, a width re-runs layout.

The template

One of these per animation, in whatever tool you already use. Plain text is fine.

```
NAME            sheet-enter
TRIGGER         user taps a list row
INTERRUPT       tapping another row: cancel, reverse at current
position

ELEMENT         .sheet
  translateY    100% -> 0     360ms   ease-enter
  opacity       0 -> 1        160ms   linear
ELEMENT         .scrim
  opacity       0 -> 0.4      200ms   linear      (starts at
0ms)
ELEMENT         .sheet-content children
  translateY    12px -> 0     240ms   ease-enter  stagger 30ms,
max 6
                                                   (starts at
120ms)

EXIT            reverse of enter, 260ms, ease-exit; scrim fades
over 200ms
LOOP            none
RESPONSIVE      below 600px the sheet is full-height; above,
640px max-width,
                translateY distance is the sheet height either
way
REDUCED MOTION  no translate; opacity only, 120ms; no stagger
ASSETS          none
```

Anybody can build that, including you in a year. Note what it makes explicit that a video hides: the scrim and the sheet start together but the content waits 120ms, the exit is faster than the entrance, and the stagger is capped.

Naming, so the file is findable

For a delivered file — a Lottie, a video, a sprite sheet — the name is part of the specification.

`checkout-success-loop-512.json` tells a developer where it goes, what it does, whether it repeats, and its authored size. `Comp 1 (1).json` tells them nothing, and it is what they will receive unless the convention is written down.

Inside the file, name the layers too. A developer debugging a Lottie in the DOM inspector sees your layer names. `Shape Layer 14` is a cost you are passing on.

The handover checklist

Before sending anything:

1. Does it play correctly on a real device on each target platform, not in the authoring tool?
2. Is the file size stated, and is it what you expected? A 900KB Lottie means an image got embedded.
3. Is there a static fallback — a poster frame, an SVG, a first-frame PNG — for reduced motion and for load failure?
4. Are the frame ranges or state names listed, if the developer needs to play parts of it?
5. Does the specification say what happens on interrupt and on exit?
6. Has somebody who did not make it read the specification and built a mental model that matches yours?

That last one catches more than the other five together.

Review it as a moving thing

A motion specification cannot be signed off as a document. Build a page — a plain HTML file is enough — that renders every animation in the system, with controls to replay, to slow to a quarter speed, and to toggle reduced motion. Open it on the oldest phone in the office.

Quarter speed is where the problems are visible. At full speed, everything looks fine, because the eye forgives a great deal in 250ms. At 25 per cent, you can see the overshoot that is one frame too long, the element that starts three frames late, the opacity that finishes before the movement does.

Chrome DevTools has an animation speed control built in, free, under the Animations panel. Use it before the review, not after.

The free path

None of this needs tooling. A markdown file in the repository next to the component beats a specification in a design tool that developers do not have a seat for. If you want it rendered, a static site generator or a plain HTML page is enough. **Storybook** is free and is the usual home for the live version; a single `motion.html` works just as well for a small team.

The limitation

A specification describes intent, and motion is judged by feel. Two implementations matching the same numbers can feel different if one of them drops frames, or starts a frame later because of how it is triggered, or runs on a device with a 120Hz display where the curve resolves differently.

So the document is necessary and not sufficient. It removes the category of disagreement that comes from ambiguity, which is most of them, and leaves the category that comes from taste, which is the one worth arguing about.

BEFORE YOU MOVE ON

Why is reviewing an animation at 25 per cent speed more useful than reviewing it at full speed?

A. Slowing an animation reveals whether it was authored with the correct easing curve, because a linear curve is indistinguishable from an eased one at normal speed.

B. The eye forgives a great deal inside 250ms, so faults of a few frames — a late start, an overlong overshoot, an opacity that finishes early — are only separable when the duration is stretched.

C. At quarter speed the browser renders each frame four times, which exposes dropped frames and compositing errors that are hidden by the display's refresh rate at normal speed.

D. Reviewers watching at full speed are subject to the repeat-exposure effect and will judge a correct animation as too slow, whereas at 25 per cent the effect does not apply and their judgement of the duration becomes reliable.

CASE STUDY · MODULE 3

The loader that arrived as a video

A four-person design agency in Coimbatore and the sole developer of a textile exporter's new order portal.

Meera animated the upload loader in After Effects: three dots, a soft glow, a blurred trail behind the leading dot, a two-second loop. She exported a 1080p MP4, six hundred and forty kilobytes, and sent it to Sathish with the project files.

Sathish could not use it, and the reasons were not about quality.

The portal has a dark theme that buyers in Europe switch on and a light one the mill office uses. The loader has to sit on three different surface colours. An MP4 has a background, so it can sit on exactly one. The loader also has to stop on a "done" pose when an upload finishes, rather than loop until it is hidden; a video plays and takes no instruction beyond play and stop. Its colours are baked, so it cannot follow a theme. And it is a fixed size, so on the wide desktop layout it is soft.

That is four separate failures and none of them is a criticism of the animation. The animation was fine. The file was wrong.

They had two days before a customer demo, and three options.

Meera could rebuild it as a Lottie. The glow is an effect and effects do not export, so it would have to be reconstructed as shapes with gradients. The trailing blur is also an effect. The dot timing came from an expression, which either does not survive or is baked into hundreds of keyframes on export. Realistically two days, which was the whole margin, and she had another job booked for one of them.

Sathish could convert the MP4 to a GIF and ship it. Three hours. It would be around three megabytes on a portal whose users are frequently on mobile data, and because GIF has one bit of transparency the anti-aliased edges would carry a hard white fringe on the dark theme — the halo problem, which

cannot be fixed inside the format. It would loop forever, so he would hide it and show a tick instead of resting on the done pose.

The third option came up on the third phone call, from Sathish. Three flat dots with staggered opacity is twelve lines of CSS. It inherits the theme's colour because it uses `currentColor`, it weighs nothing, it scales, it stops whenever he wants, and it took an hour. It is not Meera's animation.

That was the uncomfortable part, and it was not really about pride. Meera's objection was that the agency had been paid for a designed loader and was about to ship a generic one, and that the next request would be for the developer to "just do it in code" every time. Sathish's objection to the Lottie was that the file would arrive with four layers called Shape Layer 14 and no note about which frames meant what, as the last three had.

Both objections were about handover rather than about format, and both were fixable in an afternoon of writing things down. The format argument was the one that had eaten two days, and it turned out to have an answer neither of them had asked for out loud: the loader and the empty-state illustration were two different jobs that had been given the same file type because one person owned both.

STOP HERE

Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

They shipped the CSS loader for the demo and commissioned the Lottie for the release. Meera rebuilt it over the following fortnight with the glow drawn as shapes, named her layers, wrote the frame ranges in the handover note — nought to twenty-four idle, twenty-five to forty-eight success — and supplied a first-frame SVG for reduced motion and for load failure. The CSS loader is still in production on the upload path eighteen months later, because nothing about it has ever needed changing. The Lottie went onto the dashboard's emp-

ty state, where it earns its place: it is illustrated, marketing revises it twice a year, and a developer does not have to be involved to swap it. The agency added three questions to its delivery checklist — does the motion respond to state, is the content vector or photographic, and who will edit it next — and Meera says the second one would have ended the argument before it started, because three flat dots were never a job for a video.

Beyond its size, list what the MP4 could not do that the portal needed.

It has a fixed size, so it blurs when scaled. It has a background, so it cannot sit on an arbitrary or theme-dependent surface. Its colours are baked, so it cannot follow a dark theme. It cannot be paused at a meaningful state, cannot rest on a final pose, and cannot be driven by what the application is doing.

What exactly would GIF's one bit of transparency have cost on the dark theme?

A pixel in a GIF is either fully opaque or fully transparent, so an anti-aliased edge drawn over an unknown background gets a hard fringe in whatever colour the file was matted against — a white halo around every dot on a dark surface. There is no partial alpha in the format, so no export setting or edge treatment removes it.

The CSS loader was not the designer's animation. What made it a defensible answer rather than a retreat?

The format follows from three questions, and this case answered all three toward code: it must respond to application state, the content is simple vector shapes rather than anything photographic, and it will almost never change. A CSS keyframe with a staggered delay is then the smallest thing that does the job, and it gets theme colour and interruption handling from the platform for nothing. The agency traded authorship for fitness deliberately, and kept the illustrated work where a designer's ownership actually pays — the empty state that marketing revises.

MODULE 3 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 Of the three questions that decide a delivery format, which one eliminates the most options?

A. Whether the content is vector or photographic
B. Whether the motion has to respond to state
C. How often the animation will be revised
D. Whether the target platform supports alpha

2 A designer's Lottie file arrives at 900 KB. What is the most likely cause?

A. Too many shape layers, which inflate the JSON
B. The export frame rate was set to 24 rather than 60
C. The animation was exported with the canvas renderer selected instead
D. A raster image was imported and base64-encoded into the JSON

3 A gear in the middle of a 1000-unit viewBox swings across the screen when rotated. What is the fix?

A. Wrap the gear in a group and animate the group instead
B. Use animateTransform rather than CSS, which resolves origins correctly
C. Set transform-box to fill-box and transform-origin to center
D. Add a pathLength attribute so the transform is normalised to the shape

4 A 4096 by 4096 PNG sprite sheet compresses to 400 KB on disk. What does it cost on the device?

A. About 67 MB, because it is decoded to four bytes per pixel
B. About 400 KB, because the browser keeps the compressed bytes
C. About 16 MB, because PNG decodes to one byte per pixel
D. It depends on the frame rate the animation plays at

5 A 24-frame sprite strip flashes blank at the end of every loop. What was done wrong?

A. The end position was set to the sheet width minus one frame

B. The timing function should have been steps of 23 for 24 frames

C. The animation duration is not an exact multiple of the frame count

D. The sheet needs padding between frames to stop GPU texture bleed

6 A hover animation is forty per cent through when the pointer leaves. What does a state-machine runtime do that playing a clip does not?

A. It plays the hover clip through to the end before it starts the exit clip

B. It caches the remaining frames so the exit costs nothing

C. It blends to the exit from the current position instead of snapping

D. It reverses the hover clip at double speed to catch up

MODULE 4

Putting two images together

Compositing from the arithmetic up: what a matte is and where one comes from, why a dark fringe means the alpha convention is wrong, what blend modes really compute, how a key is rescued by despill and light wrap rather than by better settings, and what a tracker and a camera solver can and cannot know.

> **BY THE END OF THIS MODULE YOU CAN**
>
> Combine two pieces of footage that were never filmed together so the seam does not show — building the matte by the cheapest route the shot allows, diagnosing an edge fringe from its colour, repairing spill and edge interaction rather than the matte's interior, attaching the insert with a track or solve whose error you have read, and matching black level, grain, focus and lens distortion in a defensible order

28 · 10 min

The greyscale image that decides everything

THE ONE THING TO KEEP

A composite is one weighted mix per pixel, so the interesting work is in the grey edge values — and a difficult key is solved by splitting the frame into a garbage matte, a solid core and a narrow edge band rather than by finding better keyer settings.

One greyscale image decides everything

Every composite in every film you have seen comes down to the same operation, repeated: two images, and a third greyscale image that decides how much of each one shows at every pixel. That third image is the matte.

White means show the foreground. Black means show the background. The grey values in between are the entire art of compositing, because they are the edges — hair, motion blur, glass, smoke — and edges are where a bad composite gives itself away.

The operation is called **over**, and in its straight form it is:

```
result = foreground × alpha + background × (1 − alpha)
```

Read it once and the whole discipline unpacks. At alpha 1 you get the foreground. At alpha 0 you get the background. At alpha 0.4 you get a weighted mix, which is exactly what a real semi-transparent edge looks like.

The vocabulary, which is not standardised

Different tools use different words for the same things, which is why documentation is confusing.

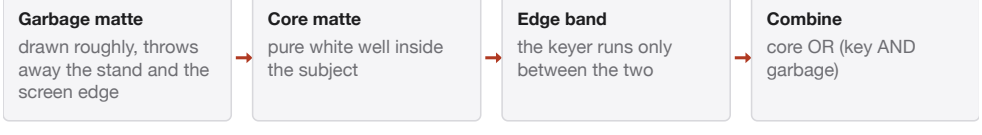
- **Alpha channel** — a matte stored as a fourth channel inside the image, alongside red, green and blue.
- **Matte** or **mask** — a matte held separately, often as a greyscale image or a drawn shape.
- **Key** — a matte generated automatically from the image's own content, usually colour.
- **Luma matte** — using another image's brightness as the matte.
- **Alpha matte** — using another image's alpha as the matte.
- **Garbage matte** — a rough hand-drawn shape that throws away everything you know you do not want: the edge of the green screen, a light stand, the boom.
- **Holdout matte** — the inverse, used to stop one element drawing over another.
- **Core matte** — a shape safely inside the subject, guaranteed solid.

The technique that solves most difficult keys

Almost every hard key is solved the same way, and it is not by finding better keyer settings.

You build **three mattes and combine them**:

Three mattes, and the keyer only has to do the third



This is why experienced compositors do not spend long on keyer sliders. Asked to solve a whole frame, a keyer must compromise between a dark jacket that goes see-through and a screen edge that will not leave. Asked to solve a narrow band of edge, it can be pushed much harder than it could globally.

1. A **garbage matte**, drawn by hand, roughly, that removes everything outside the subject. Keyers work far better when they are not being asked to deal with a C-stand in the corner.
2. A **core matte**, drawn inside the subject, which is pure white. This guarantees the interior is solid even if the keyer produces holes in it — a dark jacket against a dark-lit green screen will always produce holes.

3. An **edge matte** from the keyer, used *only* in the band between the core and the garbage matte, where the soft detail lives.

Combine: core OR (key AND garbage). The keyer is now responsible for a thin band of edge rather than the whole frame, and you can push it much harder in that band than you could globally.

This is the single most useful structural idea in compositing, and it is the reason experienced compositors do not spend long fiddling with keyer sliders.

Reading a matte properly

You cannot judge a matte by looking at the composite. Look at the matte on its own, and look at it twice.

Gain it up. Multiply the matte by 5 or 10. Areas that should be solid white and are actually 0.95 will now be visible as grey — those are the holes that will show as a faint transparency of the background through your subject, and they are invisible at normal exposure.

Gain it down, or view the inverse. Areas of the background that should be pure black but are 0.02 will show up. That is background contamination, which appears as a faint glow of the subject where there should be nothing.

In practice: a matte that looks correct at normal viewing and falls apart when gained is the normal state of a first attempt. Nuke, Fusion, After Effects and Blender's compositor all have a one-key matte view; use it constantly.

Where mattes come from

- **Keyed** — from colour, covered in the green-screen lesson.
- **Drawn** — rotoscoping, frame by frame, with splines.
- **Rendered** — a 3D render supplies a perfect alpha channel for free, plus ID mattes per object if you ask for them.
- **Depth-derived** — a depth pass or a depth-sensing camera can threshold into a matte. Consumer depth maps are low-resolution and noisy at edges, which is exactly where you need them to be good.

- **Machine-learned** — segmentation models now produce a usable person matte from arbitrary footage. They are astonishing at the body and poor at hair and motion blur, and they are temporally unstable: the matte wobbles between frames even when nothing moves. Useful as a garbage or core matte; rarely usable as an edge matte without work.

The free path

Blender's compositor does everything in this lesson — nodes, keyers, mask shapes with animation, matte viewing — and is free and scriptable. **Natron** is an open-source node compositor modelled on Nuke; be aware its development has been intermittent, so check the state of the project before committing to it. **DaVinci Resolve**'s free edition includes Fusion, a full node compositor, which is the most capable free option by some distance. **GIMP** and **Krita** handle still composites. Nuke and After Effects are named in job adverts and neither is required to learn any of this.

The limitation

A matte is a single number per pixel, and reality is not. A strand of hair thinner than a pixel is genuinely a mixture of hair colour and background colour, and no matte can separate them — the information is gone, averaged together by the sensor. That is why hair composites over a new background often keep a faint memory of the old one, and why the fix is not a better keyer but a *despill* and an edge treatment that replaces the contaminated colour rather than trying to recover it.

The same applies to motion blur, glass, smoke and water. Compositing does not undo the mixing the camera did; it decides what to mix with instead.

BEFORE YOU MOVE ON

A keyed subject shows a faint ghost of the new background through the middle of a dark jacket. Viewing the matte at normal exposure it looks solid white. What should you do first?

A. Raise the keyer's tolerance so that darker greens are included in the key, which will push the interior of the jacket further from the screen colour and close the holes.

B. Apply a slight matte erosion followed by a blur, which is the standard fix for a matte that is not fully opaque where it should be.

C. Gain the matte up by a factor of five or ten and look again, because a region at 0.95 is indistinguishable from white at normal exposure.

D. Switch the composite from a straight over to a premultiplied over, since a foreground that is being multiplied by its own alpha twice will show the background through any region whose alpha is below one, including a solid interior.

29 · 10 min

The dark fringe and what causes it

THE ONE THING TO KEEP

Premultiplied RGB has already been multiplied by alpha, so compositing it with the straight formula multiplies the edges twice — a dark fringe means premultiplied treated as straight, a bright halo means the reverse, and colour correction belongs between an unpremultiply and a re-premultiply.

The dark fringe that will not go away

You place a rendered logo with transparency over a bright photograph. Around every edge is a thin dark outline that nobody put there. You blur it, you choke the matte, you try a different export, and it persists.

This is premultiplication, it is the most common bug in compositing, and once you know the arithmetic it takes five seconds to diagnose.

Two conventions for the same file

An RGBA image stores four numbers per pixel. There are two conventions for what the RGB values mean at partially transparent pixels.

Straight (also called unassociated, or unpremultiplied). RGB holds the pixel's true colour, regardless of transparency. A half-transparent pure red pixel stores `R=1, G=0, B=0, A=0.5`. The colour is complete; the alpha is a separate instruction.

Premultiplied (associated). RGB has already been multiplied by alpha. The same pixel stores `R=0.5, G=0, B=0, A=0.5`. Where alpha is zero, RGB is necessarily black, because anything times zero is zero.

Both are valid. Neither is better in general. The file format usually does not tell you which one you have, and that is the whole problem.

The fringe tells you which way round the mistake is

Straight RGB holds the pixel's true colour whatever its alpha; premultiplied RGB has already been multiplied by it. Composite a premultiplied file with the straight formula and every edge pixel is multiplied a second time. Go the other way and the foreground arrives at full strength over a background reduced only by one minus alpha. You never have to inspect the file — the colour of the fringe is the diagnosis.

Why the fringe appears

The over operation has a form for each convention:

```
straight:        result = FG × α  +  BG × (1 − α)
premultiplied:   result = FG      +  BG × (1 − α)
```

The premultiplied version omits the multiply because it has already been done. Now take a premultiplied file and composite it with the straight formula. Every edge pixel gets multiplied by alpha a *second* time. At an edge pixel with alpha 0.5, the foreground contribution is a quarter of what it should be rather than a half. Darker than it should be, only at the edges, exactly proportional to the transparency.

Dark fringe = a premultiplied image treated as straight.

Go the other way — take a straight image and hand it to a premultiplied pipeline — and the foreground is added at full strength on top of a background that has only been reduced by `1 − α`. Too much light at the edges.

Bright halo = a straight image treated as premultiplied.

That pair is the entire diagnosis. You do not need to inspect anything; the colour of the fringe tells you which way round the error is.

Where it bites in practice

3D renders are premultiplied. Almost universally. A render from Blender, Cinema 4D or Maya with a transparent film comes out premultiplied, and if your compositor imports it as straight you get the dark edge.

Photoshop's PNG exports are straight. So are most design-tool exports.

After Effects asks you. On import it shows an "Interpret Footage → Alpha" dialogue with Straight and Premultiplied options and a "Guess" button. The guess is right most of the time and wrong often enough to matter. If you see the fringe, this dialogue is where you fix it.

EXR is almost always premultiplied by convention, and the OpenEXR specification says so.

The rule that actually matters day to day

Unpremultiply before you colour correct. Re-premultiply afterwards.

Here is why. A premultiplied edge pixel's RGB is already darkened by alpha. If you now apply a gamma or a curve, you are applying it to a value that is part colour and part transparency, and the two get tangled — the correction changes the apparent edge softness, not just the colour. The result is edges that get harder or softer as you grade, which is bewildering until you know the cause.

Every node compositor has `Unpremultiply` and `Premultiply` nodes, and the standard shape of a grading branch is unpremult, grade, premult. Nuke's colour nodes even have an `(un)premult by` input to do it inline. In After Effects the same issue appears as "the edges got worse after I added the Curves effect".

Checking it yourself

Sample a pixel in a fully transparent region of your file.

- If RGB is `0,0,0` where alpha is `0`, it is almost certainly premultiplied.
- If RGB holds a real colour where alpha is `0`, it is straight.

```
ffprobe -v error -show_entries stream=pix_fmt -of csv=p=0
render.mov
```

will tell you the pixel format, and a viewer that shows the RGB values with alpha disabled will tell you the rest. In Blender's image editor, switching the display to "Color" rather than "Color & Alpha" shows the raw RGB, which makes this a two-second check.

The free path

Blender's compositor has `Alpha Convert` nodes in both directions and renders premultiplied by default. **Fusion**, inside the free DaVinci Resolve, has

`AlphaDivide` and `AlphaMultiply`, which are the same operations under their older names. **ffmpeg** has `premultiply` and `unpremultiply` filters. **GIMP** and **Krita** both work internally in a defined convention and will tell you which on export. Nothing here needs a licence.

The limitation

Premultiplied images lose information at the edges, and it is not recoverable. At alpha 0.1, the stored RGB is a tenth of the real colour, quantised into 8 bits — so you have roughly 25 distinct levels left to describe that pixel's colour. Unpremultiplying multiplies it back up by ten and multiplies the quantisation error with it, which is why heavily graded soft edges from 8-bit premultiplied sources go blotchy.

The fix is not a different convention; it is more bit depth. 16-bit float, which is what EXR uses, makes the problem disappear because there are enough levels left at alpha 0.1 to survive the division. This is one of the genuine, concrete reasons professional compositing happens in float rather than in 8-bit.

BEFORE YOU MOVE ON

A rendered element composites cleanly. After a Curves adjustment is added to it, the edges become harder and slightly haloed. Why?

A. The curve is being applied to premultiplied values, where each edge pixel's RGB is already scaled by its alpha, so the correction acts on a mixture of colour and transparency.

B. The Curves effect is operating in a different colour space from the composite, so the edge values are being transformed twice through the transfer function.

C. The alpha channel is being adjusted by the curve along with the colour channels, which steepens the transition from opaque to transparent.

D. The render was written with a straight alpha but the compositor imported it as premultiplied, so every colour operation is being applied to values that have had a division by alpha applied to them first, amplifying quantisation error at the edges.

30 · 9 min

Every blend mode is one line of arithmetic

THE ONE THING TO KEEP

Multiply can only darken and drops white, Screen can only lighten and drops black, Add is physically correct for light and clips in 8-bit, and Difference is a measuring instrument rather than a look — which is why stock light elements are filmed on black and fail over a bright sky.

Every blend mode is a one-line formula

The blend mode dropdown has thirty entries with names like Overlay, Soft Light, Linear Burn and Pin Light, and it invites you to try them all until something looks nice. That is a slow way to work and it does not transfer between projects.

There are really about six that matter, each is one line of arithmetic on values between 0 and 1, and knowing the line tells you exactly when the mode will work.

The six

Multiply. `result = a × b`

Anything times 1 stays the same; anything times 0 goes black. So white becomes invisible and black stays black. This is the mode for **shadows**, for **ink on paper**, for anything that removes light. A scanned pencil drawing on white paper multiplied over a photograph puts the pencil lines on the photograph and drops the paper. Multiply can only darken.

Screen. `result = 1 − (1 − a) × (1 − b)`

The inverse of multiply, applied to the inverted values. Black becomes invisible and white stays white. This is the mode for **light**: fire, sparks, lens flares, smoke shot against black, light leaks, any stock element filmed on a black

background. Screen can only lighten, and it never clips, because the result approaches 1 asymptotically.

Add (Linear Dodge). `result = a + b`

Physically the correct model for two light sources in the same place, which is why render engines use it. It clips: `0.7 + 0.6 = 1.3`, and in an 8-bit pipeline that becomes 1.0 and the detail is lost. In a float pipeline it does not clip and holds the value, which is one of the practical reasons to work in float. Use Add for genuine light emission; use Screen when you want a similar look that behaves politely at the top end.

Difference. `result = |a − b|`

Where the two images are identical, the result is black. This is almost useless as a look and invaluable as a tool: put two versions of a shot on top of each other in Difference and any area that is not pure black is a place they differ. It is how you verify an export matches its source, how you find the frame where a track slips, and how you check whether a "lossless" round-trip really was.

Overlay and **Soft Light.** Both apply Multiply where the base is dark and Screen where the base is light, with different roll-offs. They increase contrast while roughly preserving the midpoint. Useful for texture and grade layers, and both are *base-dependent*, which is why swapping the layer order changes the result and why they are unpredictable over a grade you are still adjusting.

What this tells you about stock elements

Most of the smoke, fire, dust, rain and flare footage sold as "stock elements" is filmed against black and intended for Screen. That is not a style choice; it is because Screen with a black background is mathematically identical to an over-composite with a perfect matte, for light-emitting subjects.

The same logic tells you when it will fail. Screen a fire element over a bright sky and the fire nearly disappears, because `1 − (1 − 0.9)(1 − 0.8)` is 0.98 — barely brighter than the sky already was. Light elements only read against darker backgrounds. If you need fire over a bright background you need a real matte, not a blend mode.

Blend modes are not a substitute for a matte

This is the useful boundary. A blend mode makes a decision per pixel based on *value*; a matte makes it based on a decision you control. Screen removes black. It does not remove "the background", and it will happily remove the dark parts of your subject too.

The test: if the thing you want to remove is genuinely black or genuinely white, a blend mode is free and exact. If it is a colour, or a value that also appears inside your subject, you need a matte.

The one that changes with colour space

Blend modes operate on encoded values, so the same Multiply gives a different result on linear data than on sRGB-encoded data. In a linear-light pipeline, Multiply by 0.5 halves the actual light and looks like a one-stop reduction. On sRGB-encoded values, multiplying by 0.5 is a much larger perceptual change, because the encoding is non-linear.

This is why a grade that looked right in one application looks wrong when the project is moved into a linear workflow, and why compositing applications are explicit about their working space in a way that design applications are not. Photoshop's default is not linear. Nuke's, Fusion's and Blender's are.

The free path

Every application named here has the full set. **Blender**'s Mix node exposes each mode by name and you can build any of them from Math nodes if you want to see the arithmetic directly. **Fusion** in the free DaVinci Resolve, **GIMP**, **Krita** and **Photopea** all implement the standard set. The formulas are also published in the W3C compositing specification, which is free to read and is the reason browsers, design tools and compositors agree on what "overlay" means.

The limitation

Blend modes are per-pixel and context-free. They cannot know that a pixel belongs to your subject rather than the background, they cannot respect an edge,

and they have no notion of depth. Everything they do well, they do because the imagery was prepared to suit them — filmed on black, or drawn on white.

There is also no standard for how they behave outside the 0–1 range, which is where high-dynamic-range work lives. Two applications can implement the same named mode and diverge on a pixel at 4.0, which is a real source of "it looked different in the other program" that no amount of matching settings will resolve.

BEFORE YOU MOVE ON

A fire element filmed on black looks convincing screened over a night street and almost vanishes when screened over a bright daytime sky. What does the formula tell you?

A. Screen normalises by the background's average luminance, so a bright background scales the foreground contribution down proportionally.

B. The fire element was filmed with a black point above zero, so over a bright background the residual black lifts the sky and the encoder discards the difference.

C. Screening a bright value over an already bright background can only move it a little closer to one, so the visible change is small.

D. Screen is a base-dependent mode like Overlay, applying Multiply where the background is light and Screen where it is dark, so over a bright sky the fire is being multiplied into it and the dark parts of the flame are being preserved rather than dropped.

31 · 10 min

A green screen is fixed on set, not in the edit

THE ONE THING TO KEEP

A keyer builds a matte out of colour information, so anything that degrades colour on set — uneven light, spill, subsampling — cannot be recovered afterwards.

The transparent export with a black background

Before green screens, alpha. An alpha channel is a fourth channel alongside red, green and blue, storing how opaque each pixel is. Almost every compositing problem beginners hit is an alpha problem in disguise.

Two facts save hours:

H.264 MP4 does not carry alpha. If you exported a transparent animation as MP4 and got a black background, that is why. Formats that do carry alpha: ProRes 4444, PNG sequences, WebM with VP9, and HEVC with alpha (which Safari plays and most other things do not).

Straight versus premultiplied. In premultiplied alpha, the colour values have already been multiplied against a background, usually black. If the software guesses the wrong interpretation, you get a dark or bright fringe on every semi-transparent edge. The fix is to tell the application the correct one — in After Effects, **File → Interpret Footage → Main → Alpha** — not to choke the matte until the fringe hides.

What a keyer is actually doing

It is not "removing green". It measures, for every pixel, how far that pixel's colour sits from a chosen colour in some colour space, and turns that distance into a matte — a grayscale image of opacity.

Once you hold that in your head, the failures explain themselves.

- **Uneven lighting on the screen.** The green varies across the frame, so one tolerance setting cannot cover it. On set, light the screen separately and evenly first, then light the subject. That order is the entire craft.
- **The subject too close to the screen.** Green light bounces off it onto hair, shoulders and the side of the face. That is spill, and spill suppression fixes it by desaturating those areas, which is why keyed people often look slightly grey. Two to three metres between subject and screen removes the problem at source.
- **Fine hair and motion blur.** These produce semi-transparent pixels that are partly green and partly subject. A keyer must guess. This is why a key looks perfect on a paused frame and falls apart when the person turns their head.
- **Chroma subsampling.** Nearly every phone and consumer camera records 4:2:0 — full resolution for brightness, a quarter resolution for colour. The matte is drawn from colour information that does not exist at pixel resolution. That is the mechanical reason a phone-shot green screen has stair-stepped edges that no amount of edge refinement fully removes. Shooting 4:2:2 with a capable camera or an external recorder, or simply at a much higher bitrate, is the real fix.

Which is the point of the lesson: the shoot decides whether the key will work. Post-production negotiates with what the shoot handed over.

Free tools that key properly

- **DaVinci Resolve** (free): the Color page qualifier and Fusion's Delta Keyer are both professional-grade. There is no meaningful keying feature you are missing by not buying Studio.
- **Blender** (free): the compositor has keying nodes and a good despill.
- **OBS** (free): a serviceable chroma key for live streaming.
- **Kdenlive**, **Shotcut** (free): basic keyers, adequate for a well-lit shot.
- **CapCut** on a phone: chroma key exists and works about as well as a phone-shot source deserves.
- **After Effects** with Keylight is the industry default and what job ads name.

When a key will not hold, the answer is rotoscoping — masking by hand, frame by frame or with a tracked shape. Slow, and sometimes the only way.

AI matting has genuinely changed this in the last few years. Resolve Studio's Magic Mask, Runway's background removal and several free web tools will cut a person out of an unprepared shot better than a bad key ever would. Their weaknesses are consistent: the edge wobbles slightly between frames, so a still looks better than the sequence, and fine hair still goes.

Tracking, in three levels

- **Point tracking.** Follow one high-contrast feature and attach something to it. Good for pinning a label to a moving sign. Needs a distinct, non-repeating feature that stays in frame.
- **Planar tracking.** Track a flat surface — a phone screen, a poster, a wall — and get its perspective as it moves. Mocha ships with After Effects; **Blender**'s motion tracker is the free route.
- **3D camera solve.** Recover the camera's actual path through space so a 3D object can sit in the shot. This needs parallax: nearer things must move differently from further things. A camera that only rotates on the spot gives the solver nothing to work with, and it will fail no matter how good the footage looks.

Rolling shutter, from phone and mirrorless sensors, skews fast pans and breaks solves. Most trackers have a rolling-shutter compensation setting. Find it before you spend an hour wondering why the solve drifts.

Ten minutes today

Shoot ten seconds against any green surface with your phone. Hold your hand up and wave it slowly. Key it in Resolve. Then look at the edge of your moving fingers rather than the edge of your still shoulder. That difference is the honest quality of your setup, and it tells you whether the next thing to fix is the lighting or the camera.

BEFORE YOU MOVE ON

A clip shot on a phone against an evenly lit green screen keys cleanly on a paused frame, but shows stair-stepped, ragged edges whenever the subject turns their head. What is the most likely cause?

A. The screen was not lit evenly enough, so one tolerance setting cannot cover the whole frame.
B. The phone records 4:2:0, storing colour at a quarter of the luminance resolution, so a moving, motion-blurred edge is being matted from colour data that does not exist at pixel resolution.
C. There is too much green spill on the subject, and stronger spill suppression will clean up the edges.
D. The keyer's tolerance is set too high, and narrowing it will recover the detail in the moving edge.

32 · 10 min

Spill is real light, and light wrap is an honest fake

THE ONE THING TO KEEP

Green bounce lands on the subject where alpha is one, so the matte cannot remove it — despill clamps green to the average of red and blue and therefore desaturates anything genuinely green, and light wrap replaces the missing interaction between subject and background by pushing a blurred copy of the background onto the edge band.

The subject is green and the key is not the reason

You have a clean matte. The edges are soft in the right places, the core is solid, and the composite still looks wrong: the actor's shoulder has a green rim, their

hair has a green tint through it, and the whole figure sits on the new background like a sticker.

Neither problem is a matte problem. The first is spill, the second is the absence of interaction between foreground and background. They are fixed after the key, and they are most of what separates a composite that reads as real from one that does not.

Spill is real light, and it is on the subject

A green screen is a large green surface with light bouncing off it. That light lands on your subject. It is not an artefact of keying; it is a physically correct green bounce, and it would be there if you photographed the scene on film with no digital process at all.

The matte cannot remove it, because the matte only decides *whether* a pixel shows, not *what colour* it is. The green rim is inside the subject, where alpha is 1.

The standard despill algorithm is one line, and knowing it tells you what it will do to your image:

```
if green > (red + blue) / 2:
    green = (red + blue) / 2
```

Wherever green exceeds the average of the other two channels, clamp it down to that average. Variants use `max(red, blue)` instead, which is gentler, or a weighted mix, which lets you dial the aggressiveness.

The consequence is immediate and worth anticipating: **despill desaturates**. Removing green from a pixel that was genuinely greenish — a khaki jacket, a green-tinted highlight in blonde hair — leaves it grey. This is the standard complaint after despilling, and there is no algorithm that distinguishes "green light from the screen" from "a green object", because in the image they are the same photons.

The professional workaround is to restore some colour afterwards: take the luminance of the despilled area and tint it with a colour sampled from the new

environment. Nuke, Fusion and Blender all let you build this; it is a hue-shift rather than a recovery, and it is honest about being one.

Blue screens, and when to choose one

Green is standard because sensors have twice as many green photosites in the Bayer pattern, so the green channel has the best signal-to-noise and the finest detail — which is what a keyer needs.

Blue is better when the subject is green, when there is a lot of skin in frame (skin has significant red and green and very little blue, so blue spill contaminates it less), or when the scene is lit low and warm and a green bounce would be obviously wrong. The despill arithmetic is identical with the channels swapped.

Light wrap: the cue nobody names

A subject in a real environment is lit by that environment. Bright areas behind a person spill light around their silhouette — this is partly physical light wrapping around the edge, partly lens flare inside the camera, partly the softness of any real edge. A composite with a perfectly clean edge has none of it, and the eye reads "cut out" without being able to say why.

Light wrap is the fix, built in four steps:

1. Take the **background** you are compositing onto.
2. **Blur** it substantially — 10 to 40 pixels depending on the frame size.
3. Restrict it to the **edge band** of the matte: multiply by the matte's edge, typically the matte minus an eroded copy of itself.
4. **Screen or add** that onto the foreground.

The result is that a bright window behind the subject now throws a faint bloom onto their shoulder, in the correct colour, at the correct place. It costs five nodes and it is the single highest-return step in a green-screen composite after despill.

Keep it subtle. Light wrap is heavily over-applied by people who have just discovered it; if you can point at it, it is too strong.

Edge treatment, in order

The sequence that solves most edges, applied only in the edge band:

- **Despill first.** Everything afterwards works better on clean colour.
- **Erode or dilate the matte by a fraction of a pixel.** Usually eroding by 0.3 to 1 pixel removes the last of the old background. Whole-pixel erosion eats detail.
- **Blur the matte very slightly** to reintroduce softness lost by eroding.
- **Light wrap.**
- **Match the grain**, covered in its own lesson, because a clean edge on a grainy plate is visible even when everything else is right.

What cannot be fixed here

If the screen was lit unevenly, or the subject stood too close to it, or the footage is heavily compressed with 4:2:0 chroma subsampling, the colour information at the edges is already gone. Despill and light wrap operate on what is in the file; they do not recover what the codec discarded.

This is the connection back to the shoot: chroma subsampling stores colour at a quarter of the luminance resolution, so a 4:2:0 file has colour edges that are literally two pixels wide where the luminance edge is one. Every edge tool in this lesson is then working on smeared data. It is the concrete reason 4:2:2 or better is specified for green-screen work, and the reason a phone's default recording is a poor source for it.

The free path

Blender's compositor has a `Despill` option inside its keying nodes and the node set to build light wrap by hand — blur, matte erode, mix. **Fusion** in the free DaVinci Resolve has dedicated despill controls and a `MatteControl` node with erode, dilate and blur in one place. **Natron** ships Keyer and despill nodes. All free.

The limitation

Despill is destructive and approximate. It cannot tell a green object from green light, it desaturates whatever it touches, and every implementation is a heuristic rather than a physical inversion. There is no correct answer — only a choice about which error is less visible in this shot.

Light wrap is an even franker fake: it is not a simulation of anything, just a blurred copy of the background pushed onto the edge because the result looks right. Both are craft rather than science, which is why two compositors will produce different and equally defensible results from the same plate.

BEFORE YOU MOVE ON

After despilling, a subject's khaki jacket has turned grey. What is the underlying reason, and what is the realistic remedy?

A. The despill was applied before the matte was finished, so it operated on background pixels as well; re-order the operations and the jacket is preserved.

B. The keyer has included part of the jacket in the key, so the jacket is partially transparent and mixing with a neutral background; tighten the key on that colour range.

C. In the image, green light from the screen and a green object are the same photons, so the clamp cannot distinguish them; restore the hue afterwards by tinting the despilled luminance.

D. The despill algorithm clamps green to the average of red and blue, which is mathematically only correct when red and blue are equal, so any subject lit with a colour cast will lose saturation in proportion to how far apart the red and blue channels are in the original plate.

33 · 10 min

Drawing the matte by hand

THE ONE THING TO KEEP

Roto is done by halving intervals rather than working forward frame by frame, with the subject split into simply-moving parts — and machine segmentation is best used to generate the garbage and core mattes in seconds while the hand work is spent only on the edge band it gets wrong.

When there is no screen, somebody draws it

A great deal of professional compositing has no green screen at all. The shot exists, it was filmed on location, and you need a matte around an actor who was never keyed. The answer is rotoscoping: drawing the shape by hand, and animating it so it follows the subject.

It is slow, it is the entry-level job in every visual effects house, and it is the technique that makes everything else possible when the shoot did not cooperate.

Working practice that halves the time

Beginners roto frame by frame, forward, from frame 1. That is the slowest possible method.

Work in binary. Set the shape on the first frame and the last frame of the range. Go to the middle and correct it. Then the middle of each half. Then the middle of each quarter. The software interpolates between your keyframes, so you are only fixing what interpolation got wrong, and the error is largest at the midpoint. Typically you need a keyframe every 5 to 15 frames rather than every frame.

Split the subject into parts. One shape for the forearm, one for the upper arm, one for the torso, one per finger if the shot demands it. A single shape

trying to follow an entire articulated body needs a keyframe on every frame; separate shapes each move simply and can be interpolated over long stretches. Parent them so the hand follows the forearm.

Match the point count to the shape, and keep it low. Every extra control point is another thing to animate. Most limbs need four to eight points. A shape with forty points is a shape you will not finish.

Do not track the outline; track the form. The outline of a rotating arm changes; the arm does not. Place points where features are, not evenly around the silhouette.

Motion blur is the real difficulty

A crisp spline against a crisply-filmed subject is manageable. A fast-moving limb is smeared across twenty pixels, and there is no line where the arm ends — there is a gradient from arm to background.

Three approaches, all partial:

- **Feather the shape** by an amount that varies along its length — heavier on the leading edge, lighter where the motion is small. Every roto tool supports per-point feather for exactly this reason.
- **Render the matte with motion blur**, letting the tool blur the shape according to how fast its points are moving between frames. This is accurate when the shape's motion matches the subject's, and wrong when it does not.
- **Roto the shape as if there were no blur**, then blur the resulting matte directionally to match. Crude, quick, and adequate for a matte that will sit under a heavy effect.

A shot with severe motion blur is a shot that should have been keyed or shot differently, and saying so early is part of the job.

Planar tracking: the technique that changed the discipline

For any part of a subject that is roughly flat — a forehead, a phone screen, a wall, a door, a clothing panel, a licence plate — you do not have to animate the

shape at all. A **planar tracker** analyses the movement of a whole region of pixels and solves for the four-corner perspective transform that maps frame A onto frame B. You draw the shape once and the tracker carries it through the shot, including perspective change.

This is what Mocha does, and it is why "Mocha" appears in job listings as a skill of its own. It works on surfaces that stay planar and fails on things that deform — a face turning through profile, cloth folding, anything organic that bends out of plane.

The free equivalent is **Blender's** motion tracking with a plane track: define a plane from four tracked markers and it solves the same corner pin, and you can parent a mask to it. It is genuinely capable, and it is the strongest free entry in this module.

When machine segmentation helps, and when it does not

Segmentation models will now produce a person matte from arbitrary footage, and the free ones — Segment Anything and its video successors, the `rembg` family, the person-segmentation features in DaVinci Resolve's free edition — are remarkable at the body.

Two failure modes matter, and both follow from how the models work.

Temporal instability. A per-frame model has no knowledge of the previous frame, so the matte's edge shifts slightly every frame even on a static subject. Played back, this reads as a crawling, boiling edge. Video-aware models reduce it and do not eliminate it.

Edges. These models are trained and scored on region-level accuracy, so being right about 99 per cent of the pixels — the interior — scores well while getting hair and motion blur wrong. Those are precisely the pixels a composite is judged on.

The productive use is therefore not "generate the final matte". It is: use the model to produce a garbage matte and a core matte in seconds, and spend your hand-roto time only on the edge band. That is a genuine several-fold speed-up and it is available free today.

Quality control

Check roto by playing the matte, not by scrubbing it. A shape that looks right on every individual frame can chatter when played, because the control points are jittering by a fraction of a pixel. Chatter is far more visible in motion than a shape that is consistently a pixel too small.

View the composite with the background replaced by a flat, saturated colour — magenta is traditional — and play it at full speed. Anything that flickers is a problem; anything that holds steady is done.

The free path

Blender has a full mask editor with bezier shapes, per-point feather, animation and plane tracking, free. **Fusion** in DaVinci Resolve's free edition has polygon and B-spline masks with tracking. **Natron** has a roto node modelled on Nuke's. **Segment Anything** and **rembg** are open source and run on modest hardware. Mocha and Silhouette are the paid industry names, worth knowing for a CV, not required for the skill.

The limitation

Rotoscoping is honest manual labour and no technique removes that. A complex shot is measured in days per second of footage, which is why it is outsourced in volume and why the economics of the discipline are what they are.

It is also a reconstruction, not a measurement. You are deciding where the edge is, and on a blurred or low-contrast edge that decision is a judgement. Two artists produce two different mattes, both defensible, and the shot only needs the one that survives playback.

BEFORE YOU MOVE ON

Why is setting keyframes at the start, the end, and then the midpoint of each remaining interval faster than animating a roto shape forward frame by frame?

A. Because interpolation between two keyframes is most wrong at the midpoint, so correcting midpoints repeatedly converges on the shape with the fewest keyframes.

B. Because the software can only interpolate between keyframes that are an even number of frames apart, so a binary subdivision guarantees valid interpolation intervals.

C. Because working backwards from the end frame lets the tracker use the sharper frames first, which improves the interpolation quality across the whole range.

D. Because a shape with keyframes at regular intervals produces smoother bezier interpolation between control points than one with irregular spacing, and binary subdivision is the only scheme that keeps the spacing regular at every level of refinement.

34 · 10 min

What a tracker can and cannot see

THE ONE THING TO KEEP

A tracker correlates a patch of pixels against a search region, so a feature on a straight edge is unsolvable along that edge — the aperture problem — and every good feature is one with contrast in two directions.

What a tracker is actually doing

A 2D point tracker does one thing per frame: it takes a small rectangle of pixels from the previous frame — the pattern — and searches a larger rectangle in the current frame for the position where those pixels match best. It reports

that position, usually to a fraction of a pixel by fitting a curve to the correlation scores around the best whole-pixel match.

That is the whole mechanism. Every tracking failure follows from it, and once you can name the mechanism you stop guessing at settings.

The aperture problem, which explains most failures

Put a tracker on a long straight edge — the top of a table, the join between two walls, a cable against the sky. The track slides along the edge and the result is useless.

The reason is geometric. Through a small window onto a straight edge, motion *along* the edge produces no change in the pixels at all. The pattern at position x looks identical to the pattern at position x+5. The tracker can measure the component of motion perpendicular to the edge and has no information whatsoever about the component parallel to it. This is the aperture problem, and it is not a limitation of the software.

The consequence is the single most useful rule in tracking: **a good feature has contrast in two directions.** A corner. A crossing. A bolt head. A high-contrast blemish. Not an edge, not a smooth gradient, not a repeating texture like brickwork — repeats give the search several equally good answers and the track jumps between them.

Blender's tracker and most others can show you a correlation score per frame. A feature that is genuinely trackable holds above 0.9; one that drifts along an edge falls off steadily and tells you before the result is visibly wrong.

Pattern size and search size

Two rectangles, two different jobs.

The **pattern** is what is being matched. Make it large enough to contain a distinctive feature and small enough that everything inside it moves together. If your pattern spans two objects at different depths, they move by different amounts and no single position matches both.

The **search area** is where the tracker looks. It must be large enough to contain the frame-to-frame movement and no larger — a bigger search area is slower and offers more chances to find a false match. If the camera whips, the movement between frames can be a hundred pixels and a default search area will simply lose the feature.

When a track fails on a fast move, the cause is nearly always the search area, not the pattern.

The four things that break a track

Motion blur. A smeared feature has no sharp structure to correlate against. Trackers commonly survive mild blur and fail on heavy blur, and the usual answer is to track a different, larger feature that stays distinguishable when smeared.

Occlusion. Something passes in front. Every tracker offers a way to keyframe the track manually through the occlusion and resume; on a short occlusion, interpolating between the position before and after is usually accurate enough.

Lighting change. The pattern is matched on pixel values, so a feature that moves through a shadow changes its values. Most trackers can normalise for brightness, which handles a gradual change and not a hard shadow edge.

Parallax inside the pattern. A feature near a depth discontinuity — a corner of a building against a distant background — changes its surroundings as the camera moves. Shrink the pattern until it contains only the near object.

Using a track

Once you have a path, there are three standard applications.

- **Stabilise.** Apply the inverse of the track to the footage. This crops, because the frame is being moved around inside itself, so a stabilised shot is always smaller than its source — typically 5 to 15 per cent. Decide this before framing the shot, not after.

- **Match-move.** Apply the track to an inserted element so it appears attached. One point gives position; two points give position, rotation and scale; four points give a full perspective corner pin.
- **Drive an effect.** A blur, a censor patch, a colour correction that follows a moving object.

Planar over point, when you can

For any surface that is flat and stays flat, a planar track is more robust than four point tracks, because it uses thousands of pixels rather than four small patches and can tolerate parts of the surface being occluded or badly lit. A phone screen, a poster, a wall, a face's forehead — all planar enough. This is the practical reason planar tracking took over so much of the work that used to be point tracking.

The free path

Blender's motion tracking is a full implementation — 2D tracks with adjustable pattern and search, correlation display, plane tracks, and a solver — and it is free with no watermark and no resolution cap. It is the single best free tool in this module. **Fusion** in DaVinci Resolve's free edition has a tracker and a planar tracker. **Kdenlive** and **Shotcut** both expose simple motion tracking for effects. Mocha is the paid standard and worth naming on a CV.

The limitation

A tracker reports where a pattern of pixels went. It does not know what an object is. It will happily track a reflection, a shadow, or a specular highlight that is sliding across a surface rather than attached to it — and a highlight is a superb-looking track that is physically wrong, because its position depends on the light and the camera rather than on the object.

The result looks convincing in the tracker's own display and slips visibly once you attach something to it. The defence is to check what you are tracking with your eyes before you press the button, and to test the result by attaching a crosshair and playing the shot at full speed.

BEFORE YOU MOVE ON

A track placed on the long horizontal join between two wall panels drifts sideways over the shot, although the correlation score stays high. What is happening?

A. The pattern box is larger than the search box, so the tracker is matching a subset of the pattern at several positions and choosing whichever scores highest by chance.

B. Motion blur along the direction of camera movement has smeared the join, so the tracker is locking onto the blur's leading edge, which moves faster than the feature itself.

C. Through a small window onto a straight edge, motion along the edge produces no change in the pixels, so the tracker has no information about that component at all.

D. The join is a low-frequency feature, and sub-pixel interpolation of the correlation surface is only accurate for high-frequency detail, so the reported position accumulates a rounding error each frame.

35 · 11 min

Solving for a camera, and when you cannot

THE ONE THING TO KEEP

A camera solve triangulates depth from parallax, so a purely rotating camera carries no depth information at all and can only be solved as a nodal pan — and reprojection error in pixels, not the look of the point cloud, is the number that says whether the solve will hold.

From flat tracks to a camera in space

A 2D track tells you where a feature moved on the screen. A **camera solve** — also called matchmoving or a 3D track — works out where the camera was in three dimensions on every frame, and where a cloud of tracked points sits in

the world. Once you have that, you can put a 3D object into the shot and it will stay put as the camera moves, with correct perspective.

The solver is doing structure from motion: it has many features, each seen from many camera positions, and it searches for the set of 3D point positions and camera poses that best explains all the 2D observations at once. The measure of "best" is **reprojection error** — project each solved 3D point back through each solved camera and measure how far it lands from where the feature actually was, in pixels.

That number is the one thing to look at. Under about 0.5 pixels on a 1080p plate is a solve you can build on. Between 1 and 2 is marginal and will slide. Above 3, something is wrong and no amount of adding more tracks will fix it.

The limitation that surprises everybody

A camera that only rotates cannot be solved for 3D.

If the camera is on a tripod and pans, every feature in the frame moves in a way that depends only on the rotation. A near object and a distant mountain shift by the same amount on the sensor. There is no parallax, so there is no depth information, so the solver has nothing to triangulate with. It is not a software limitation — the information does not exist in the footage.

This is why solvers offer a separate **nodal** or **tripod** mode: it solves rotation only, gives you no point cloud and no translation, and is entirely adequate for adding a distant element like a sky replacement or a far-off building. What it cannot do is let you place an object on the floor in front of the camera, because there is no floor in the solution.

The practical consequence on set: **if you want a 3D solve, move the camera sideways.** Even a metre of lateral movement over a shot turns an unsolvable pan into a solvable one. A handheld shot usually solves well for precisely this reason — the operator's sway supplies parallax for free.

What the solver needs from you

Enough tracks, spread through depth. Twenty to forty good tracks is typical. They must not all be on one plane — tracks that all sit on a flat wall give an ambiguous solution, because a plane seen from different angles has multiple consistent interpretations. Get near objects and far objects in the set.

Tracks that last. A solve is stronger when features persist across many frames, because each one is then constraining many camera positions. A shot where every track lives for eight frames is a weak solve even if there are two hundred of them.

The focal length, if you have it. Solving for the focal length as well as the geometry is possible and less accurate. If you know the lens was 35mm on a Super 35 sensor, tell the solver and it has one fewer unknown. Most modern cameras and phones record the focal length in metadata; `exiftool` will read it out of the file, free.

Lens distortion handled. A real lens bends straight lines, most visibly at wide angles. The solver assumes a pinhole camera. If you feed it distorted footage, the error shows up as a solve that fits the centre of frame and drifts at the edges. The standard workflow is: estimate the distortion, undistort the plate, solve and composite on the undistorted image, then redistort the finished comp so it matches the original lens. Blender, Fusion and Nuke all have this loop, and skipping it is a common cause of an insert that looks glued on near the frame edge.

Setting the scene up afterwards

A solve gives you a camera and a point cloud in arbitrary units, with an arbitrary origin and orientation. It does not know which way is up or how big anything is. Before you can place an object you have to constrain it:

- Pick three or more points you know lie on the ground and define them as the floor plane.
- Pick two points a known distance apart and set the scale — the gap between two paving slabs, a person's height, a door.
- Set an origin so your scene's coordinates are meaningful.

Skipping this is why a first solve produces a cube that drifts through the floor at forty-five degrees.

Checking the solve honestly

Do not judge it by looking at the error number alone. Put a simple 3D object in the scene — a grid on the floor, a cube on a surface, a sphere at a tracked point — and play the shot at full speed. A solve with acceptable numbers can still slide if the distortion was not handled or the scene orientation is slightly off, and playback shows it in a second.

Then check the individual track errors and delete the worst offenders. A single bad track with a large error can distort the whole solution, and removing three bad tracks is usually worth more than adding thirty good ones.

The free path

Blender has a complete matchmoving pipeline — 2D tracking, lens distortion estimation, a camera solver with per-track error readout, scene orientation tools, and then the 3D scene and compositor in the same application. It is used on real productions and costs nothing. This is genuinely a case where the free tool is a first-class option rather than a compromise. **COLMAP** and **Meshroom** are open-source structure-from-motion packages aimed at photogrammetry that can also produce camera solutions. PFTrack, 3DEqualizer and SynthEyes are the paid specialists.

The limitation

A solve describes a rigid world seen by a moving camera. Anything in the scene that moves independently is a contaminant, and the solver has no way to know which tracks are on the moving car and which are on the road. You remove them by hand, and on a busy shot that is most of the work.

Rolling shutter is the other reliable spoiler. Most CMOS cameras expose the frame line by line over several milliseconds, so during fast movement the top and bottom of the frame are from different moments and straight verticals lean. A solver assuming a global shutter fits an average that is wrong every-

where. Some tools compensate; the compensation is a model, not a measurement, and heavy rolling shutter remains a reason a shot cannot be solved to a usable error.

BEFORE YOU MOVE ON

A shot filmed from a locked-off tripod, panning left to right, will not solve for a 3D camera. Adding more tracks and running the solver again does not help. Why?

A. The tracks are all moving in the same direction at the same rate, so the solver cannot distinguish camera rotation from a uniform translation of the whole scene.

B. Under pure rotation, near and distant features shift identically on the sensor, so there is no parallax to triangulate depth from and the information is absent from the footage.

C. Tripod shots are usually shot at a longer focal length, and the solver cannot estimate a long focal length reliably without being given it in advance.

D. The solver needs tracks that persist across many frames to constrain each camera position, and a pan continuously moves features out of frame, so no track survives long enough to contribute to more than a handful of camera poses in the solution.

36 · 11 min

Black level, grain, focus, and the order to do them in

THE ONE THING TO KEEP

A composite usually fails on black level first and grain second, both of which are measurable rather than matters of taste — and grain and lens distortion go last in the chain because they are properties of the camera, so anything applied after them is physically happening inside the lens.

Four cues, and the viewer checks all of them

The matte is clean, the despill is done, the track holds. The shot still looks composited. It nearly always fails on one of four things, and in this order of frequency:

1. **Black level.** The insert's shadows go to 0; the plate's shadows sit at 8 or 12 out of 255.
2. **Grain.** The plate has noise; the insert is clean.
3. **Focus.** The insert is sharp everywhere; the plate's lens has depth of field.
4. **Colour and contrast.** The insert was lit by a different light than the scene.

None of these is subtle once you know to look, and all are measurable rather than matters of taste.

Black level, the cheapest fix in compositing

Real footage almost never reaches zero. Lens flare, atmospheric haze, sensor noise floor and the codec's own black offset mean the darkest pixel in a photographed frame typically sits somewhere between 4 and 20 in 8-bit terms. A rendered element or a graphic goes to a true 0.

Put a true black next to a lifted black and the eye reads the true black as a hole. This is the "sticker" effect, and it is frequently the whole problem.

Measure it: put a pixel probe on the darkest part of the plate and read the value. Then lift your insert's black point to match. In any colour tool this is the lift or offset control; in ffmpeg it is one `curves` or `colorlevels` call. Thirty seconds of work, and the single largest improvement available for the effort.

Do the same at the top. If the plate's highlights roll off and clip at 240 rather than 255, matching that matters too, though less.

Grain, and why adding it is not cheating

Every photographed image has noise — from the sensor's read noise, from photon shot noise, amplified by whatever gain was used. It is not a defect; it is a signature of the capture, and its absence is conspicuous.

Two properties have to match. **Amplitude** varies with brightness: shot noise scales with the square root of the signal, so a grain tool applying uniform noise across the range looks wrong, and the good ones let you set the response per luminance band. **Size and structure** follow the sensor: a large sensor at low ISO gives fine grain, a small one at high ISO gives coarse blotches with chroma noise in the blue channel. Grain that is too fine disappears after compression; grain that is too coarse reads as a filter.

The most accurate method is not to synthesise it. Find a flat, featureless section of the plate — a wall, a patch of sky — subtract a heavily blurred copy of itself from it, and you have the plate's real grain as a signed difference to add to your insert. It matches by construction rather than by eye.

The order a composite is matched in

1 Key or roto	build the matte
2 Despill	green clamped to the red–blue average
3 Colour match	black level first, then balance
4 Defocus	match the depth it is meant to sit at
5 Light wrap	the background's bloom on the edge band
6 Motion blur	if the insert moves and the plate is blurred
7 Lens distortion	a property of the camera, not the subject
8 Grain	last, and never before a blur

Grain and distortion go last because they are properties of the lens, and the lens is the outermost thing in the chain — anything applied after them is physically happening inside the glass. Most composites fail on black level first and grain second, and both are measurable rather than matters of taste.

Apply grain **last**, and never before a blur: blurring grain destroys it.

Focus, and the direction that matters

A real lens has one plane in focus and everything else progressively softer. If your insert is meant to sit two metres behind the subject, it should be softer than the subject by roughly the amount the plate's other background objects are.

Judge it by finding something in the plate at the same depth and comparing edge sharpness directly — zoom to 200 per cent and look at an edge in each. Guessing produces inserts that are systematically too sharp, because renders and graphics are perfectly sharp by default and our instinct is to preserve detail we worked for.

Note also that defocus is not a Gaussian blur. Out-of-focus highlights in a real lens become discs or polygons the shape of the aperture — bokeh — rather than fading smoothly. For a background with bright points in it, a Gaussian blur reads as wrong. Compositors have dedicated defocus nodes with an aperture shape for exactly this; Blender's Defocus node and Fusion's Defocus both do it.

Lens distortion and chromatic aberration

A wide lens bends straight lines and refracts red and blue slightly differently, producing coloured fringing that grows toward the frame edges. Your insert has neither. Near the centre nobody notices; near the edges, straight insert edges in a curved plate are obvious. Undistort the plate, composite, redistort — or apply a matching distortion to the insert alone. Chromatic aberration is one node: offset red and blue radially by a fraction of a pixel, scaled by distance from centre.

The order of operations

A defensible default, and departures from it should be deliberate:

1. Key or roto, build the matte.
2. Despill.
3. Colour match — black level first, then overall balance, then contrast.
4. Defocus to match depth.
5. Light wrap.
6. Motion blur, if the insert moves and the plate's motion is blurred.
7. Lens distortion.
8. Grain.

Grain and distortion are last because they are properties of the camera, and the camera is the outermost thing in the chain. Anything applied after them is, physically speaking, happening inside the lens.

Checking it

Three tests that catch what staring does not:

- **Play at full speed.** Half of all composite errors are temporal and invisible when paused.
- **Halve the size, or squint.** Removing detail leaves the broad values, where black-level and contrast mismatches live.

- **Flip it horizontally.** Mirroring resets the adaptation your eye has built up, and mismatches jump out. An old painting technique that works exactly as well here.

The free path

Blender's compositor has all of it — a Defocus node with aperture blades, grain by difference, colour balance with lift, gamma and gain, and lens distortion in both directions. **Fusion** in the free DaVinci Resolve has the same set under its own names, plus Resolve's scopes for measuring black level properly. **ffmpeg** can add grain and adjust levels for simple cases. Scopes matter more than the compositor here, and Resolve's are free and broadcast-grade.

The limitation

Matching is judged by eye and there is no metric that says a composite is done. Reprojection error tells you a track is good; nothing tells you a grade is right. The nearest thing to an objective test is the flip-and-squint pair above, and experienced compositors still miss things that a fresh viewer catches in three seconds.

There is also a hard floor set by the source. An 8-bit, heavily compressed plate has quantised shadows and blocked-up grain, and an insert graded to match will inherit those artefacts rather than the plate's real texture. You can match a degraded image, but you cannot match it well.

BEFORE YOU MOVE ON

An inserted element is graded, defocused and given matching grain, then blurred slightly at the end to soften it into the plate. The result looks soft and oddly clean. What went wrong?

A. The defocus and the blur are compounding, so the element is now softer than anything in the plate at that depth, and the cleanliness is a perceptual consequence of the excess softness.

B. The blur after the grain has averaged the grain away, leaving an element that is both soft and noise-free — grain belongs last in the chain.

C. The grain was applied uniformly across the luminance range rather than scaled by brightness, so it sits mainly in the midtones and is not visible in the shadows where the eye looks for it.

D. The blur was applied to the premultiplied element rather than after unpremultiplying it, so the grain in the partially transparent regions has been scaled down by alpha and is no longer visible against the plate.

CASE STUDY · MODULE 4

Forty minutes of lecture on a phone-shot green screen

The media cell of an arts and science college in Pune, over the winter break.

Three chemistry lecturers, forty minutes of usable takes, and a plan to key each of them over animated diagrams for a revision series. The shoot had been cheap on purpose: a green cloth pinned to the wall of a seminar room, two phones — one recording, one as backup — and the room's own tube lights.

The assistant keyed the first take in DaVinci Resolve's free edition. On a paused frame it looked clean. The moment Dr Kulkarni turned her head it fell apart: stair-stepped edges through her hair, a green rim along her shoulder that survived every setting, and a matte that crawled between frames even where nothing moved.

Three causes, all of them decided before the camera rolled. The phones recorded 4:2:0, so the colour information a keyer builds its matte from exists at half resolution horizontally and vertically — the colour edge is two pixels wide where the luminance edge is one, and no amount of edge refinement recovers what was never sampled. The cloth was lit by the room, so it was about two stops brighter on the left than the right and no single tolerance covered both. And the lecturers stood roughly sixty centimetres from the cloth, so green bounce landed on the side of every face and shoulder.

The break was half over when they had to decide.

Reshooting meant borrowing two mirrorless bodies from the photography department that could record 4:2:2 to an external recorder, hiring two soft lights for four days at about nine thousand rupees, and getting three lecturers back — two of whom had gone home to Nagpur and Belgaum, and one of whom had made it clear that he had done this as a favour and would not be doing it twice. The cost was a week the cell did not have, money it did not have, and goodwill it could not replace.

Working with what existed meant a free segmentation model to generate a garbage matte and a core matte in minutes per shot, hand-roto confined to the

edge band between them, despill, and a light wrap built from the diagram behind. Estimated at three weeks of the assistant's time for forty minutes of footage, and the hair would still boil.

On the second day the assistant proposed a third thing: stop compositing. Put the lecturer in a corner of the frame at a fixed size, give the diagram the rest of the screen, and key nothing at all. The green cloth then stops being a matte source and becomes a background to be graded to the series colour, which it is perfectly good at.

The head of the cell resisted it at first, and his reason was not vanity. The series had been pitched to the principal as "the lecturer inside the diagram", and a corner box looks like every recorded Zoom class the students had already sat through for two years. Against that, the assistant put a number on the alternative: at three weeks for forty minutes, the series would not exist before the exams it was made for.

They also checked the one thing that might have rescued the original plan. The backup phone had recorded the same takes from a slightly different angle, and if either device had been set to a higher bitrate the chroma might have held up better. It had not; both files were the same 4:2:0 at the same rate. There was nothing to go back to, which is the ordinary state of affairs when the shoot is discovered to be the problem.

STOP HERE

Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

They keyed nothing for thirty of the forty minutes and composited the other ten — the shots where a lecturer's hand entered the diagram to point at a bond, which is the only place the composite carried information the layout could not. The three-week estimate for those ten minutes came in at nine days, because the segmentation model handled the bodies in seconds and the

hand work stayed inside the edge band, apart from two shots where a loose sleeve crossed the frame. Nobody was called back to Pune. The cell wrote a one-page shoot note for the next series: light the screen separately and evenly first and the subject second, keep two to three metres between subject and cloth, and shoot 4:2:2 if a key is planned at all. It ends with the sentence the assistant has repeated ever since — post-production negotiates with what the shoot handed over.

The picture looked fine. Why did 4:2:0 specifically ruin the key?

A keyer does not remove green; it measures how far each pixel's colour sits from a chosen colour and turns that distance into a matte. In 4:2:0 the colour is sampled at a quarter of the luminance resolution, so the data the matte is drawn from is already blurred across two pixels before the keyer starts. That produces stair-stepped edges on hair and motion blur that edge refinement cannot fix, because the information was never recorded.

The matte was clean and the green rim on the shoulder stayed. Why can a matte not remove it?

The rim is real light: green bounced off a large green surface and landed on the subject, in a region where alpha is one. A matte decides whether a pixel shows, not what colour it is. Removing it is a despill operation, which clamps green down to about the average of red and blue — and which desaturates anything that was genuinely green, because in the image a green jacket and green bounce are the same photons.

Option three keyed nothing at all. What made that engineering rather than surrender?

The right first question is whether the background is genuinely arbitrary, and here it was not — for thirty minutes the composite carried no information the layout could not carry. Asking it cost ten seconds and removed most of the problem along with most of the cost. The shots that survived the question were exactly the ones where the composite did something a side-by-side layout cannot: put the lecturer's hand inside the diagram.

MODULE 4 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 Why do experienced compositors build a garbage matte and a core matte before touching the keyer?

A. Because a keyer cannot produce matte values between zero and one on its own

B. Because the keyer then handles only a narrow edge band, and can be pushed harder

C. Because garbage and core mattes are required by any 4:2:0 chroma-subsampled source

D. Because it halves the render time of the keying node

2 A rendered logo over a bright photograph has a thin dark outline on every edge. What is the cause?

A. A straight file being composited with the premultiplied formula

B. The matte has been eroded by more than a whole pixel

C. The render was written in 8-bit rather than in 16-bit floating point

D. A premultiplied file being composited with the straight formula

3 After despilling, an actor's khaki jacket has gone grey. Why?

A. The matte ate into the jacket; dilate it by a pixel and re-key

B. The light wrap was applied before the despill and has bleached it

C. The plate was interpreted as full range, so re-tagging it will bring the colour back

D. Despill clamps green above the red and blue average, so real green goes with it

4 A tracker placed on the top edge of a table slides along it and produces a useless track. What is the mechanism?

A. The search area is too small to contain the camera's movement between frames

B. Motion along a straight edge produces no change in the pattern

C. The pattern spans two objects at different depths

D. The tracker's brightness normalisation is switched off

5 A shot from a locked-off tripod pans across a room. Why will a 3D camera solve fail?

A. A rotating camera produces no parallax, so there is nothing to triangulate

B. Rolling shutter always defeats a solver on a panning shot

C. Tripod shots contain too few trackable features to constrain the solution at all

D. The solver needs the focal length, which a pan does not record

6 Why do grain and lens distortion go last in the compositing chain?

A. Because they are the slowest nodes and should not be recomputed

B. Because both are destructive and cannot be undone once applied

C. Because they are properties of the camera, which is outermost in the chain

D. Because grain must be added before any blur if it is to survive compression

MODULE 5

Codecs, colour and delivery

What a compressor throws away and why, the difference between a frame you can seek to and one you cannot, how to hit a quality target or a size target on purpose, and the four colour and timing decisions behind most files that look washed out, banded or juddery when they arrive.

> BY THE END OF THIS MODULE YOU CAN
>
> Export a piece of video against a stated constraint — a size budget, a platform's requirements, an archival master, an editable intermediate — by choosing the codec class, rate-control mode, bit depth and colour tags from what the file is for; and diagnose a delivered file that looks milky, crushed, banded, juddery or twenty times too large by naming which of those decisions produced it

37 · 10 min

What a codec throws away

THE ONE THING TO KEEP

Compression exploits specific limits of vision — colour resolution first, then fine detail within blocks — so the bitrate a clip needs is set by how unpredictable its content is rather than by its resolution, and noise is expensive precisely because it cannot be predicted.

Compression is a set of decisions about what you will not notice

A raw 1080p frame is 1920 × 1080 × 3 bytes, about 6MB. At 25 frames a second that is 150MB every second. A streamed 1080p video runs at about 0.6MB a second. The compressor is throwing away roughly 99.6 per cent of the data and the result still looks acceptable.

It manages that by exploiting specific, known limits of human vision. Knowing which ones tells you exactly when a codec will fail, and the failures are predictable rather than mysterious.

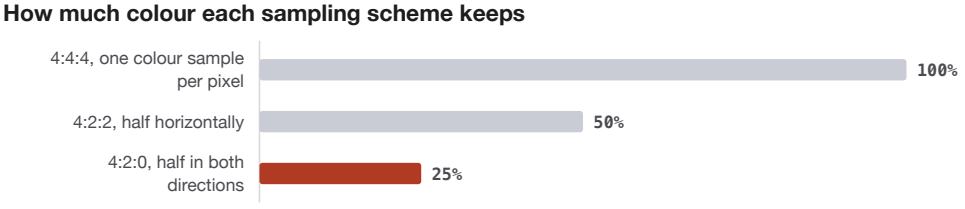
Colour resolution goes first

Your eye has around 6 million cone cells for colour and 120 million rods for brightness. Spatial acuity for colour is far below that for luminance — you can see a fine black-and-white grating that becomes a flat grey when the same pattern is made of two equally bright colours.

Codecs exploit this with **chroma subsampling**, written as three numbers.

- **4:4:4** — full colour resolution. One colour sample per pixel.
- **4:2:2** — colour sampled at half the horizontal resolution. Half the colour data.
- **4:2:0** — colour sampled at half horizontally *and* half vertically. A quarter of the colour data.

Almost everything you watch is 4:2:0. Every phone recording, every YouTube stream, every Blu-ray.



Almost everything you watch is 4:2:0, because the eye's spatial acuity for colour is far below its acuity for brightness. The consequences are specific rather than general: a single-pixel red line goes muddy, fine coloured text picks up a smeared fringe, and a green-screen edge is drawn from colour data already blurred by two pixels before the keyer starts.

The consequences are concrete. A single-pixel red line on a grey background has no colour samples of its own; its colour is averaged with its neighbours and it comes out muddy. Fine coloured text on a contrasting background gets a smeared fringe. And, from the compositing module: a green-screen edge in 4:2:0 has colour information at half resolution in both directions, so the keyer is working on data that is already blurred by two pixels before it starts.

This is the reason a 4:2:2 camera setting exists and matters for green screen, graphics and heavy grading, and does not matter for a talking head that will be watched once.

Blocks and frequencies

The image is cut into blocks — 8 × 8 in older codecs, variable up to 64 × 64 in HEVC and AV1 — and each block is transformed from pixels into frequency coefficients by a discrete cosine transform. The first coefficient is the block's average brightness; later ones describe progressively finer detail within it.

The compression step is **quantisation**: divide each coefficient by a number and round. The numbers are larger for high-frequency coefficients, because fine detail is less noticeable than broad tone. Rounding sends many high-frequency coefficients to zero, and long runs of zeros compress to almost nothing.

Everything visible about compression artefacts follows from this:

- **Blocking** — when quantisation is coarse, neighbouring blocks end up with different average values and the grid becomes visible. Codecs apply a deblocking filter to smooth it, which is why heavily compressed modern video looks smeared rather than blocky.
- **Ringing** and **mosquito noise** — ripples around sharp edges, because a sharp edge needs high-frequency coefficients that have been quantised away, and the reconstruction overshoots.
- **Banding** — smooth gradients lose the low-amplitude detail that made them smooth.
- **Detail loss in texture** — grass, water, confetti, rain, film grain. These are high-frequency, unpredictable, and enormously expensive to encode. The codec's response is to flatten them.

That last one is the practical planning point: **content, not resolution, drives the bitrate you need.** A static interview at 1080p needs a fraction of what a 1080p shot of rain on water needs, and no encoder setting changes that.

Why noise is expensive

A codec works by predicting and then coding the difference between prediction and reality. Noise is by definition unpredictable, so every noisy pixel costs bits. A grainy or high-ISO source can easily need twice the bitrate of a clean one for the same visible quality — and the bits are being spent on reproducing noise.

This is why a light denoise before encoding often improves apparent quality at a fixed bitrate: you are moving the bits from the noise to the picture. It is also why the grain you added in compositing partly disappears after upload, and why streaming platforms are interested in synthesising film grain at the decoder — AV1 has a grain synthesis tool for exactly this reason, transmitting grain parameters rather than grain.

The generational problem

Every encode quantises. Encode, decode, edit, encode again, and the second encoder is now trying to reproduce the first one's artefacts — including its block edges, which are high-frequency detail it did not have to encode the first time.

Three or four generations of a lossy codec is visibly worse than one. This is the entire reason intermediate codecs exist, covered in the next lesson, and the reason you never edit from a downloaded delivery file if a master exists.

The free path

ffmpeg is the reference implementation of most of this and lets you see it directly. To watch the subsampling loss on your own footage:

```
ffmpeg -i source.mov -vf format=yuv420p,format=yuv444p
chroma.mp4
```

That round-trip through 4:2:0 and back shows you exactly what the colour subsampling costs, with no encoding involved. **x264**, **x265**, **libvpx** and **libaom** are all open-source encoders and their documentation is unusually honest about what each setting does.

The limitation

None of this predicts perceived quality reliably. Codec decisions are tuned against average viewers on average content, and your content may not be average — anime, screen recordings, medical imagery and scientific data all break assumptions built for filmed footage. PSNR and SSIM, the standard automatic quality metrics, correlate only loosely with what people actually notice, which is why platforms increasingly use models like VMAF and why even those are contested.

The honest method is still to encode a representative thirty seconds at two or three settings and look at them, at full size, on the kind of screen your audience uses.

BEFORE YOU MOVE ON

Two 1080p clips are encoded at the same CRF. A static interview comes out at 2 Mbit/s; a handheld shot of rain on water comes out at 11 Mbit/s. Why does constant quality produce such different rates?

A. The handheld shot has more camera movement, and motion vectors are the largest component of an encoded frame at high bitrates.
B. The rain clip is likely to have been shot at a higher ISO, and the encoder allocates bitrate in proportion to the sensor gain recorded in the file's metadata.
C. Rain and moving water are high-frequency and unpredictable, so prediction fails and the residual detail that must be coded is large.
D. The interview is mostly skin tones, which fall in the part of the colour space where chroma subsampling discards the most information, so far fewer chroma coefficients survive quantisation.

38 · 10 min

Frames that stand alone, and frames that do not

THE ONE THING TO KEEP

A P- or B-frame is defined in terms of its neighbours, so a decoder cannot show frame 43 without decoding from the last keyframe forward — which is why long-GOP material scrubs badly, why all-intra intermediates exist, and why a proxy beats a hardware upgrade.

Why scrubbing a phone video feels like wading

Drag the playhead through footage straight off a camera and it lags, sometimes by half a second. Drag through the same footage converted to ProRes and it responds instantly, from a file five times larger. Nothing about your ma-

chine changed. The difference is whether each frame can be decoded on its own.

Three kinds of frame

I-frame (intra-coded, also called a keyframe). Compressed on its own, like a JPEG. Everything needed to reconstruct it is in the frame.

P-frame (predicted). Stored as a set of instructions relative to an earlier frame: take that block, move it 14 pixels left and 3 up, and add this small correction. For typical footage a P-frame is a small fraction of an I-frame's size.

B-frame (bidirectional). Predicted from frames both before *and* after it in display order. Even smaller. Because they reference a future frame, the frames have to be transmitted out of display order, which is why a video has a decode order and a presentation order that differ.

A **GOP** — group of pictures — is one I-frame and the dependent frames that follow it. A typical streaming encode uses a GOP of one to two seconds: at 25fps, an I-frame every 25 to 50 frames.

What follows for editing

To show frame 43 of a GOP starting at frame 1, the decoder must decode frames 1 through 43. There is no way around it; frame 43 is defined in terms of its predecessors. Scrub backwards and it is worse, because going back one frame means decoding forward from the keyframe again.

This is why editing "long-GOP" material is slow, and why the standard professional move is to transcode to an **all-intra** codec first. Every frame is independent, every seek is a single decode, and the price is file size.

The all-intra options, with the free ones named:

- **ProRes** (Apple) and **DNxHR** (Avid) — the two paid-ecosystem standards, both readable and writable by ffmpeg and by the free DaVinci Resolve. ProRes 422 runs around 150 Mbit/s at 1080p25 — roughly 20 GB an hour.

- **FFV1** — open source, mathematically lossless, and the format chosen by several national archives, including for preservation work at the Library of Congress. In a Matroska container with checksums per frame, it is the free archival answer.
- **Ut Video** and **HuffYUV** — free lossless intermediates, fast, larger than FFV1.
- **MJPEG** — every frame a JPEG. Old, inefficient, universally decodable, and still occasionally the pragmatic answer.

Keyframes and streaming

The GOP length is a trade-off with two visible consequences.

Seeking granularity. A player can only start at a keyframe. With a 10-second GOP, clicking the middle of a progress bar means the player decodes up to 10 seconds of video before it can show you anything — or shows you a frame up to 10 seconds away from where you clicked.

Adaptive streaming. HLS and DASH cut the video into segments and switch quality at segment boundaries, which must be keyframes. If your keyframe interval does not divide evenly into the segment length, the packager inserts extra keyframes or the segments drift.

The convention that avoids both problems: a keyframe every 2 seconds, with segments of 2, 4 or 6 seconds.

```
# 2-second keyframe interval at 25fps, no scene-cut keyframes
ffmpeg -i in.mov -c:v libx264 -g 50 -keyint_min 50 -sc_thresh-
old 0 out.mp4
```

`-sc_threshold 0` turns off automatic keyframes at scene cuts. For general viewing you want them on, because a cut is exactly where a fresh I-frame is cheap and useful. For adaptive streaming you want them off, so every keyframe is where the packager expects it.

Cutting without re-encoding

Because a GOP is a unit, you can trim a file at keyframe boundaries without touching the compressed data:

```
ffmpeg -ss 00:01:30 -i in.mp4 -t 00:00:45 -c copy out.mp4
```

`-c copy` means no decode and no encode: it is nearly instantaneous and there is no generational loss at all. The catch is that the cut lands on the nearest keyframe, so your out point can be up to a GOP away from where you asked. For a rough trim before uploading, this is the right tool and almost nobody uses it.

Where long-GOP wins

Nothing above says long-GOP is bad. For delivery it is dramatically more efficient — the whole reason streaming is possible. A 45-minute programme at 5 Mbit/s H.264 is about 1.7 GB; the same programme in ProRes 422 is about 50 GB.

The rule is the point in the chain. **Long-GOP for delivery, all-intra for work.** Acquisition sits in between and increasingly uses long-GOP for capacity, which is why the transcode step exists.

Proxies: the free middle path

You do not have to transcode to a huge intermediate to edit comfortably. A **proxy** is a small, low-resolution, all-intra copy that the editor uses for playback while the timeline still refers to the original for export.

```
ffmpeg -i big.mp4 -vf scale=-2:540 -c:v dnxhd -profile:v
dnxhr_lb \
  -c:a pcm_s16le proxy.mov
```

Every serious editor supports this: DaVinci Resolve, Kdenlive and Shotcut all have proxy workflows in their free versions. It is the answer to "my laptop

cannot edit 4K", and it is far more effective than any hardware upgrade at the same cost.

The free path

ffmpeg encodes and decodes every format named here, including ProRes and DNxHR. **DaVinci Resolve**'s free edition has a full proxy and optimised-media workflow. **Kdenlive** and **Shotcut** both generate proxies with one setting. **FFV1** and **Matroska** are both open standards with no licensing questions, which is precisely why archives chose them.

The limitation

All-intra editing costs storage at a rate that surprises people. An hour of ProRes 422 HQ at 1080p is roughly 30 GB; at 4K it is over 100 GB. A week of shooting fills a drive, and the backup of that drive fills another.

There is also no single intermediate everyone agrees on. ProRes is effectively a standard and is Apple's; DNxHR is Avid's; FFV1 is open and slower to encode and not accepted by many facilities. Choosing one is a workflow decision with political and practical edges, and anybody who tells you there is an obvious answer works somewhere that already made it.

BEFORE YOU MOVE ON

A team preparing video for adaptive streaming sets -sc_threshold 0 alongside a fixed two-second keyframe interval. Why turn off automatic keyframes at scene cuts, when a cut is exactly where a keyframe is cheapest?

A. Scene-cut keyframes are placed by a heuristic that misfires on fast-cut material, inserting so many I-frames that the average bitrate rises beyond the ladder's budget.

B. Adaptive streaming switches quality only at segment boundaries, which must be keyframes, so the keyframe positions have to be identical and predictable across every rendition.

C. Scene detection requires the encoder to look ahead, which is incompatible with the low-latency buffering that HLS and DASH players rely on.

D. A keyframe inserted at a scene cut is encoded without reference to the frames around it, so it cannot be used as the start of a B-frame pyramid, and the packager would have to re-encode the following frames to rebuild the reference structure at every segment boundary.

39 · 10 min

Quality target or size target

THE ONE THING TO KEEP

CRF asks for a quality and lets the size fall where it may, two-pass asks for a size and distributes the bits; the preset changes how hard the encoder searches rather than the quality it aims at, and a change of about six in CRF roughly halves or doubles the file.

Two different questions, two different controls

"How big should this file be?" and "how good should this look?" are different questions, and encoders have separate modes for them. Choosing the wrong

mode is the most common cause of a file that is either enormous or unnecessarily ugly.

Constant quality (CRF, or constant rate factor). You state a quality target and the encoder spends whatever bits that takes — more on the complicated shots, fewer on the simple ones. The output size is not known in advance.

Target bitrate (ABR, VBR, two-pass). You state a size or a rate and the encoder hits it, distributing the bits as well as it can. Quality varies with the content.

Constant bitrate (CBR). Every second uses the same number of bits, whatever is happening. Wasteful on easy content, poor on hard content, and required for broadcast transport streams and some live-streaming ingests because the downstream system has a fixed pipe.

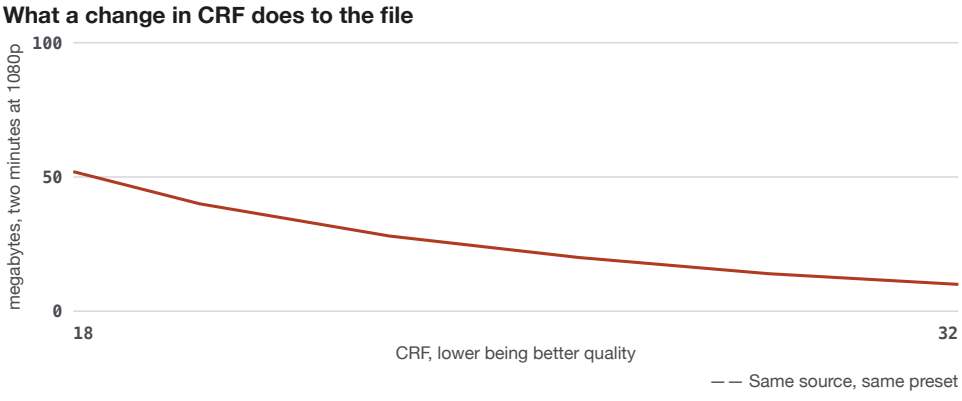
CRF, which is what you usually want

```
ffmpeg -i in.mov -c:v libx264 -crf 20 -preset slow -c:a aac -
b:a 192k out.mp4
```

The scale for x264 and x265 runs 0 to 51, lower being better. The numbers worth knowing:

- **CRF 18** — visually near-transparent for most content. Large files.
- **CRF 20–23** — the practical range for good delivery. 23 is x264's default.
- **CRF 26–28** — visibly soft on detail and motion; acceptable for small previews.

A change of about 6 in CRF roughly halves or doubles the bitrate. CRF 26 is about a quarter the size of CRF 20 on the same source. That relationship is the useful one to carry around, because it lets you estimate without trial encodes.



A change of about six in CRF roughly halves or doubles the size, which lets you estimate without trial encodes. The preset is a separate control: at a fixed CRF it changes how hard the encoder searches for a cheap way to reach that quality, so a slower preset gives a smaller file of the same quality rather than a better one. Numbers do not carry between encoders — x265 at 28 is roughly x264 at 23.

x265 and AV1 use the same scale with different meanings. x265's CRF 28 is approximately x264's CRF 23 in quality, and produces a smaller file. Do not carry a number between encoders.

The preset is not quality

```
-preset veryslow   # slowest, most efficient
-preset slow
-preset medium     # default
-preset veryfast
-preset ultrafast  # fastest, least efficient
```

With CRF fixed, the preset does not change the target quality — it changes how much **search effort** the encoder spends finding a cheap way to reach it. A slower preset at the same CRF produces a smaller file of the same quality. Going from `medium` to `slow` typically saves 5 to 10 per cent of the size for roughly double the encode time; `veryslow` saves a little more for a lot more time.

The practical rule: use the slowest preset you can tolerate for anything that will be downloaded many times, and `veryfast` for a preview you are going to look at once.

When you genuinely need a size

Two-pass, which is what "fit this into 8MB" means:

```
# target bitrate = (target size in bits) / duration, minus au-
dio
ffmpeg -y -i in.mov -c:v libx264 -b:v 1400k -pass 1 -an -f mp4
/dev/null
ffmpeg    -i in.mov -c:v libx264 -b:v 1400k -pass 2 -c:a aac -
b:a 128k out.mp4
```

The first pass analyses the whole file and writes a log of how complex each part is. The second pass uses it to distribute the bits. This is meaningfully better than a single-pass attempt at the same bitrate, because the encoder in pass one does not yet know that a difficult scene is coming.

The arithmetic: for an 8MB target over 30 seconds, that is 64 megabits over 30 seconds, about 2.1 Mbit/s total. Subtract 128 kbit/s for audio and a few per cent for container overhead, and you have roughly 1.9 Mbit/s for video.

The one flag people forget

```
-movflags +faststart
```

An MP4 contains a `moov` atom holding the index of the whole file. By default ffmpeg writes it at the end, because it does not know the final sizes until it has finished. A browser streaming that file has to download to the end before it can play a single frame.

`+faststart` runs a second pass over the finished file to move the index to the front. Without it, your web video does not start until it has fully downloaded, which people then diagnose as a slow server. It costs a file rewrite and it is not optional for anything served over HTTP.

Hardware encoders and what they cost

`h264_nvenc`, `h264_qsv`, `h264_videotoolbox` and `h264_vaapi` encode on the GPU or a dedicated block, often ten to twenty times faster than x264.

They are less efficient. At the same file size, a hardware H.264 encode is noticeably worse than `x264 -preset slow` — the gap is usually described as one to two CRF points, and it is larger on difficult content. The silicon implements a smaller search.

Use them for live streaming, for screen recording, for anything where encode time is the constraint. Use software for anything that will be distributed.

Choosing the codec

- **H.264** — universal. Every browser, every device, every player. Still the correct default for general delivery.
- **H.265 / HEVC** — roughly 30 to 50 per cent smaller at the same quality. Patent licensing is complicated and browser support is uneven, which is why it is common in apps and rare on the open web.
- **VP9** — comparable to HEVC, royalty-free, supported in Chrome, Firefox and Edge, and by Safari in WebM.
- **AV1** — another 20 to 30 per cent over VP9, royalty-free, and slow to encode. `libaom` is the reference and is very slow; `SVT-AV1` is far faster and is what most people should use now.

The free path

Everything above is **ffmpeg** and the open-source encoders it wraps: x264, x265, libvpx, SVT-AV1. These are not budget alternatives — x264 is the encoder against which commercial products are benchmarked, and it is what most of the industry ships. **HandBrake** is a free graphical front end for the same encoders if the command line is not for you.

The limitation

CRF is a quality target only in the sense that the encoder's internal model says so. That model is tuned on typical camera footage, and it misjudges content that is not: screen recordings with large flat areas and sharp text, animation with hard edges, and anything with heavy grain. For those, the CRF that looks right is often three or four points from the usual one, and there is no way to know except to look.

And every quality number here — CRF values, "30 per cent smaller", the PSNR and VMAF scores behind them — is an average over test material chosen by the people quoting it. On your footage the numbers will differ, sometimes substantially, which is why a thirty-second test encode remains the only honest method.

BEFORE YOU MOVE ON

A file encoded at CRF 20 with -preset medium is re-encoded at CRF 20 with -preset veryslow. What should you expect?

A. A noticeably better-looking file of about the same size, because the slower preset spends its extra time improving the picture at the same bitrate.

B. An identical file, because CRF fully determines the output and the preset only affects how long the encoder takes to produce it.

C. A smaller file of roughly the same quality, taking considerably longer to encode.

D. A larger file of better quality, because a slower preset enables more reference frames and a deeper B-frame pyramid, and each additional reference frame the encoder is permitted to use adds to the per-frame header cost across the whole stream.

40 · 10 min

Why your export looks washed out

THE ONE THING TO KEEP

Limited range puts black at 16 and white at 235 while full range uses the whole 0 to 255 — so a milky, low-contrast export is limited data read as full, a crushed and over-contrasty one is full data read as limited, and untagged files leave every player to guess differently.

Your export looks washed out and nothing is broken

The most common complaint in video delivery: the file looked correct in the editor, and in QuickTime or on the web the blacks are milky and the contrast is gone. Or the opposite — the shadows are crushed and the highlights are blown.

Nothing was damaged. The pixel values in the file are almost certainly correct. What is wrong is the *description* of what those values mean, and the player is interpreting them under a different convention from the one they were written with.

The two conventions

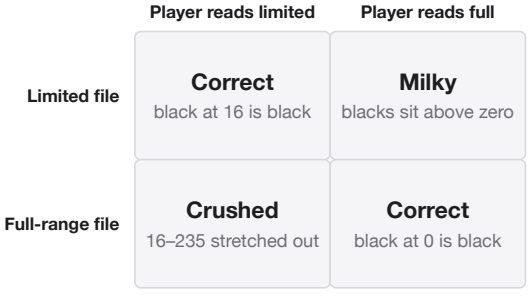
Full range (also called PC range, or `pc` in ffmpeg). Black is 0, white is 255. All 256 levels carry picture.

Limited range (video range, studio range, `tv` in ffmpeg). Black is 16, white is 235. The headroom below 16 and above 235 exists for historical reasons — analogue signal excursions, overshoot from filtering — and is preserved in digital video because broadcast equipment expects it.

Almost all delivered video is limited range. Almost all computer graphics, screenshots and rendered output are full range.

Now the arithmetic. A limited-range file interpreted as full range: the player takes 16 to be a real 16 rather than black, so blacks sit above zero — **washed out, milky, low contrast**. A full-range file interpreted as limited: the player expands 16–235 to 0–255, so everything below 16 clamps to black and everything above 235 clamps to white — **crushed shadows, blown highlights, over-contrasty**.

Milky or crushed, and which way round each one is



Limited range puts black at 16 and white at 235; full range uses all 256 levels. Nothing is damaged in either failure — the numbers in the file are right and the description of them is missing, so every player guesses and different players guess differently. Tag the four values on export and the whole category of problem disappears.

Those two descriptions are the diagnosis. Milky means limited read as full. Crushed means full read as limited.

Why it happens, and why it happens more than it should

The container can carry the answer. MP4 and MOV have a `colr` atom; Matroska has colour elements; H.264 and HEVC bitstreams have video usability information. All of them can say "limited range, BT.709 primaries, BT.709 transfer".

Many encoders simply do not write those tags. Faced with an untagged file, every player guesses, and different players guess differently — by resolution (assuming standard-definition means BT.601 and high-definition means BT.709), by codec, or by a fixed default. The same file then looks different in VLC, in Safari, in Chrome and in Premiere, and each of them is behaving reasonably.

Tag it explicitly

```
ffmpeg -i in.mov -c:v libx264 -crf 20 \
  -pix_fmt yuv420p \
  -colorspace bt709 -color_primaries bt709 -color_trc bt709 \
  -color_range tv \
  -movflags +faststart out.mp4
```

Four tags, and they cost nothing. `-colorspace` is the matrix used to convert between RGB and the luma-chroma representation; `-color_primaries` says which red, green and blue; `-color_trc` is the transfer characteristic, the curve; `-color_range` is the one this lesson is about.

Check what a file claims:

```
ffprobe -v error -select_streams v:0 \
  -show_entries stream=pix_fmt,color_range,color_space,col-
or_primaries,color_transfer \
  -of default=nw=1 out.mp4
```

If those come back `unknown`, you have found your bug before anybody reports it.

Converting rather than relabelling

Tagging says what the numbers mean. Sometimes you need to actually change them — a full-range source going into a limited-range delivery has to be scaled, not relabelled.

```
ffmpeg -i graphics.mov -vf scale=in_range=full:out_range=limit-
ed \
  -color_range tv -c:v libx264 -crf 18 out.mp4
```

Relabelling without scaling produces exactly the crushed look described above. This is the usual reason a motion graphic cut into filmed footage has harder blacks than everything around it.

The macOS QuickTime gamma story

For years, files exported from one application and viewed in QuickTime on macOS appeared with shifted gamma, while the same file viewed in VLC looked right. The cause was a mixture of missing tags and a system colour-management path that applied a display transform some applications expected and others did not.

Modern macOS is far more consistent and the problem is much reduced, but two lessons survive. First, **a video player is a colour-managed application and the display profile is part of the chain** — the same file genuinely can look different on two calibrated monitors and both be correct. Second, **judge an export in more than one player**, on more than one machine, before deciding it is right.

Where this bites hardest

- **Motion graphics and titles.** Authored full range in a design tool, dropped into a limited-range timeline. Blacks become crushed and pure white text clips.
- **Screen recordings.** Full range by nature, delivered to platforms that assume limited.
- **Green screen.** A range error changes every value in the frame, including the ones the keyer thresholds against, so a key that worked on the original fails on the re-tagged copy.
- **Anything round-tripped through a tool that strips tags**, which includes several popular uploaders and converters.

The free path

ffmpeg and **ffprobe** do all the tagging, converting and checking, free. **DaVinci Resolve**'s free edition has proper colour management with explicit input and output settings and broadcast-grade scopes — the parade and waveform will show you a milky black at 16 instead of 0 immediately, which is far more reliable than judging by eye on an uncalibrated laptop. **MediaInfo** is free and shows every colour tag in a file in a readable form.

The limitation

The tags are a claim, and nothing enforces them. A file can be tagged limited and contain full-range data, and no tool will object. Tagging tells a well-behaved player what to do; it does not make the data correct, and a mislabelled file is harder to diagnose than an untagged one because everything looks authoritative.

There is also no universal agreement on what to do with an untagged file, which means the only genuinely safe position is to tag everything you produce and to verify anything you receive. This is unglamorous and it removes an entire class of problem that otherwise reappears on every project.

BEFORE YOU MOVE ON

A motion graphic exported from a design tool is cut into filmed footage. In the finished piece, the graphic's blacks are noticeably harder and its white text clips, while the surrounding footage looks correct. What has happened?

A. The graphic was authored full range and has been relabelled as limited without being scaled, so 0 to 255 is being expanded as though it were 16 to 235.

B. The graphic is 8-bit while the footage is 10-bit, so the graphic's values are being shifted two bits left during the conversion and the top of its range overflows.

C. The graphic is tagged with BT.601 primaries while the timeline is BT.709, so its colours are being converted through the wrong matrix.

D. The design tool exported with a straight alpha channel that the editor is interpreting as premultiplied, so the graphic's values are being divided by their own alpha, which pushes opaque blacks below zero and opaque whites above the clipping point.

41 · 10 min

Counting the steps in a gradient

THE ONE THING TO KEEP

An 8-bit channel has 256 levels, so a fifty-level sky across a 1080-pixel frame has steps 21 pixels wide and bands visibly — and dither works by turning a structured quantisation edge into random error the eye integrates as smooth.

Count the steps in the gradient

Open a dark sky, a studio backdrop lit with a soft falloff, or a simple gradient behind a title, and you will often see distinct bands instead of a smooth ramp. The cause is arithmetic you can do in your head.

An 8-bit channel has 256 levels. Spread a gradient from black to white across a 1920-pixel-wide frame and each level covers about 7.5 pixels. Restrict it to a realistic range — a sky that runs from level 40 to level 90, fifty levels over 1080 pixels — and each step is 21 pixels wide.

The human visual system is very good at detecting an edge, including an edge of about 1 per cent in luminance between two large flat areas. A 21-pixel step is a visible line. This is **banding**, and it is not a compression artefact in origin; it is quantisation.

Ten bits, and why it fixes it

A 10-bit channel has 1024 levels. The same fifty-level range is now two hundred levels, each step about 5 pixels wide, below the threshold at which the eye separates them for most content. 12-bit gives 4096.

That is the entire argument for higher bit depth, and it is specifically about **smooth gradients and heavy grading**, not about "more colours" in any sense that matters for a detailed image. A photograph of a forest has so much variation that 8 bits per channel is ample. A sunset, a studio cyclorama, a fade

to black, or an image that has been pushed two stops in a grade, is where 8 bits runs out.

Grading is the multiplier here. Lift the shadows of an 8-bit image by two stops and you are stretching perhaps 30 levels across the range where 120 used to be. Whatever banding was latent becomes obvious, and there is nothing to recover — the intermediate values were never recorded.

Encoding at 10-bit even from 8-bit sources

This surprises people: encoding an 8-bit source to a 10-bit output can produce a *better-looking* file at the same bitrate.

The reason is internal precision. The encoder's transforms, motion compensation and rounding all happen at the working bit depth. At 8 bits, rounding error accumulates through those stages and shows up as extra banding. At 10 bits there is enough headroom that the rounding stays below the visible threshold.

```
ffmpeg -i in.mov -c:v libx265 -pix_fmt yuv420p10le -crf 22
out.mp4
```

For HEVC this is the Main 10 profile, and support is broad — every modern phone and television decodes it. For H.264, the equivalent is High 10, and support is much patchier: many hardware decoders refuse it. So the practical advice splits: **10-bit HEVC or AV1, yes; 10-bit H.264, only if you control the players.**

Dithering: making the error invisible

If you must stay at 8 bits, the fix is to add a small amount of noise before quantising.

The mechanism is worth understanding because it is counter-intuitive. Without dither, a value sitting between two levels always rounds the same way, so an entire region snaps to the same level and its neighbour snaps to the next — a hard edge. With dither, the value plus a little noise rounds up in

some pixels and down in others, in proportion to where it sat between them. The average across a region is now the correct value, and the edge has become a stochastic transition the eye integrates as smooth.

You have traded a visible structured error for an invisible random one. The same principle appears in audio, for exactly the same reason, when reducing from 24-bit to 16-bit.

```
ffmpeg -i in.mov -vf "gradfun=strength=1.2:radius=16" -c:v
libx264 -crf 20 out.mp4
```

`gradfun` is ffmpeg's debanding filter: it finds gradients, smooths them, and dithers the result. `deband` in other tool sets does the same. Use it lightly — too much strength smooths real detail.

The film-grain approach from the compositing module does the same job with a nicer texture: real or synthesised grain is dither, and it is why grainy footage does not band.

Where the depth is lost

Bit depth is a chain, and the lowest link sets the result.

- **Camera.** Many consumer cameras record 8-bit internally even when the sensor reads more. A camera that records 10-bit 4:2:2 gives you a genuinely different grading range.
- **The codec.** An 8-bit codec discards the rest no matter what fed it.
- **The intermediate.** Rendering a graded sequence through an 8-bit intermediate and grading again is where a surprising amount of banding is introduced.
- **The working space.** Compositing in 8 bits and applying several operations compounds rounding at each step. Float is the standard answer, and it is why EXR exists.
- **The display.** Most laptop panels are 8-bit, some 6-bit with temporal dithering. You may not be able to see your own 10-bit file's advantage on the screen you are grading on.

That last point matters: banding visible on your monitor may be your monitor, and banding invisible on your monitor may still be visible on a viewer's television. Check on more than one display before you chase it.

The free path

ffmpeg encodes 10-bit HEVC, VP9 and AV1, and has `gradfun` and grain filters. **DaVinci Resolve**'s free edition works in 32-bit float internally and exports 10-bit. **Blender** renders and composites in float and writes EXR. **ImageMagick** handles high-bit-depth stills. None of this is behind a paywall; the constraint is usually the camera and the display rather than the software.

The limitation

Bit depth is not free — 10-bit encoding is slower, files are larger, and hardware decode support varies by device and codec. And it cannot recover what was never captured: an 8-bit source graded aggressively will band regardless of what you encode it to, and adding bit depth at the end of the chain buys nothing.

It also does not save you from a badly-lit gradient. A backdrop with a genuinely steep falloff lit badly will band in 10 bits too. The ordering is: light it well, capture it deep, work in float, dither on the way out — and skipping any one of those is where the bands come from.

BEFORE YOU MOVE ON

Why can encoding an 8-bit source to a 10-bit HEVC output produce a better-looking file at the same bitrate, given that the source has no extra information to carry?

A. The 10-bit encode stores the same values with more headroom, so any subsequent grading of the delivered file has more levels to stretch across.
B. The encoder's transforms and motion compensation run at the working bit depth, so the extra headroom keeps accumulated rounding error below the visible threshold.
C. 10-bit encoding implies 4:2:2 chroma subsampling in the HEVC Main 10 profile, so colour resolution doubles alongside the bit depth.
D. A 10-bit encode applies dithering automatically when converting from the 8-bit source, and it is this dither — not the bit depth itself — that removes the banding, which means the same benefit is available by adding noise before an ordinary 8-bit encode.

42 · 11 min

Log footage and what a LUT is not

THE ONE THING TO KEEP

Log encoding allocates code values roughly evenly per stop so a wide range survives a limited file, and a LUT is a fixed lookup table with a defined input domain — which is why a conversion LUT is decoding rather than grading, and why applying a look before exposure is corrected destroys detail permanently.

Why the footage looks grey and flat

Footage shot in a log mode looks washed out, desaturated and wrong straight off the camera. That is the intended appearance, and it exists for a specific reason.

A scene can contain a range of brightness far wider than a display can show — a window and a shadowed interior in one frame might span fourteen stops. A Rec.709 display handles about six to eight. The sensor may capture twelve to fifteen. The question is how to store that range in the limited code values a file has.

Store it linearly and you waste levels. Human vision responds roughly logarithmically to light: the difference between 1 and 2 units of light is far more visible than between 100 and 101. A linear encoding spends half its levels on the brightest stop, which is the one where the eye is least sensitive.

Log encoding allocates code values roughly evenly per stop instead. Each doubling of light gets a similar number of levels. The result fits fourteen stops into a 10-bit file without visible steps in the shadows, and looks flat on screen because it is not meant to be viewed directly — it is a container.

Every manufacturer has one: S-Log, V-Log, C-Log, N-Log, Log-C, F-Log. They differ in curve shape and in where middle grey sits, which is why a LUT for one camera applied to another is wrong.

A LUT is a lookup table, nothing more

A **1D LUT** maps each channel independently: for this input value of red, output that value. Three lists of numbers. It can change contrast and per-channel balance and cannot change hue relationships, because red's output does not depend on green.

A **3D LUT** maps a colour to a colour. It is a cube — commonly 33 × 33 × 33 — of sample points in RGB space, with the output colour stored at each. Values between the samples are interpolated, usually tetrahedrally. A 33-cube is 35,937 entries. Because output depends on all three inputs, a 3D LUT can rotate hues, apply film-like colour crosstalk, and do everything a 1D LUT cannot.

The `.cube` format is plain text. Open one and you will see a header and a long list of RGB triples. There is nothing mysterious inside.

The distinction people get wrong

There are two quite different uses, and calling both "a LUT" causes most of the confusion.

A conversion LUT transforms from one defined encoding to another — S-Log3/S-Gamut3 to Rec.709, for instance. It is a technical operation with a correct answer, supplied by the camera manufacturer. Applying it is not grading; it is decoding.

A look LUT is a creative grade someone baked into a table: a teal-and-orange treatment, a film emulation, a bleach bypass. It has no correct answer and it is not tied to your footage's exposure.

The fatal ordering mistake is to apply a look LUT to log footage that has not been exposure-balanced first. The LUT was built expecting a particular input; feed it footage that is a stop under and every value lands in the wrong part of the table. The result is crushed shadows or clipped highlights that cannot be recovered afterwards, because the LUT has already discarded them.

Correct order: conversion or colour-space transform → exposure and white balance → look → final trim. Grading before the transform, or looking before exposure, produces results nobody can fix later.

Why a LUT clips and a transform does not

A LUT has a defined input domain, usually 0 to 1. Values outside it — specular highlights in a float pipeline, log values above the table's top entry — are clamped or extrapolated badly. So a LUT applied to high-dynamic-range data throws away everything above its ceiling.

This is why colour-managed pipelines increasingly use parametric transforms rather than tables. **OpenColorIO** — the open-source colour management system used by Blender, Natron, Nuke and, in recent versions, Resolve — describes the chain as a set of named colour spaces and transforms rather than as baked tables. It is free, it is the standard in visual effects, and Blender ships with a usable configuration out of the box.

Practical notes that save an afternoon

- **Never bake a look into your master.** Grade non-destructively and export the master in a wide space; apply looks on the way to each delivery.
- **A monitoring LUT is not an applied LUT.** Every editor and camera lets you view through a LUT while recording the log. That is the correct arrangement: you see a normal image and keep the range.
- **Check the LUT's expected input.** A `.cube` built for log input applied to already-converted Rec.709 footage produces a badly over-contrasted image, and it is a common mix-up because both files are just video.
- **Exposure discipline matters more with log.** Log's shadow range is thin; underexposing log and lifting in the grade produces noise very quickly. Most manufacturers recommend exposing to the right of their nominal middle grey, sometimes by a stop or more, and their documentation says by how much.

The free path

DaVinci Resolve's free edition is the strongest free colour tool in existence, includes LUT support and colour management, and is what a large part of the industry actually uses. **Blender** ships with OpenColorIO and the Filmic and AgX view transforms, which are well-regarded, free, and demonstrate the parametric approach directly. **ffmpeg** can apply a `.cube` with `-vf lut3d=file.cube` for batch work. **OpenColorIO** itself is open source. Camera manufacturers publish their conversion LUTs free.

The limitation

A LUT is a fixed table and it knows nothing about your shot. It cannot compensate for a colour cast, a different white balance, or a subject at a different exposure. Two shots from the same camera in the same scene will not match just because they went through the same LUT — matching shots is a separate job done by eye and by scope.

And the popular "film emulation" LUTs are approximations of one property of a film stock — its colour response — with none of the grain structure, halation

or highlight roll-off that make film look like film. They are a starting point some people find useful and a claim to treat carefully, because no lookup table can reproduce a photochemical process.

BEFORE YOU MOVE ON

A colourist applies a film-look LUT to log footage that is a stop underexposed, then tries to bring the exposure up afterwards. The shadows stay crushed. Why can the damage not be undone?

A. The LUT has baked a contrast curve into the metadata, which overrides any subsequent exposure node in the colour pipeline.

B. Log footage cannot be exposure-corrected after any transform, because the log curve is non-invertible once a 3D interpolation has been applied to it.

C. The LUT mapped several distinct input values onto the same output, so the distinctions that carried the shadow detail no longer exist to be lifted.

D. Applying a look LUT converts the footage from its camera colour space into Rec.709, and Rec.709 has a narrower gamut than the camera gamut, so the shadow information has been moved outside the representable colour volume and clipped at the gamut boundary rather than at the black point.

43 · 10 min

Frame rates, shutter and judder

THE ONE THING TO KEEP

Motion reads as continuous because each frame carries blur covering roughly half the interval to the next — the 180-degree rule — and judder comes from showing frames for unequal lengths of time, as 3:2 pulldown does when 24 is forced onto a 60Hz display.

The numbers, and where they come from

24, 25, 30, 50, 60. None of them is arbitrary and the history explains every annoyance that follows.

24 was the sound-film standard from the late 1920s — the slowest rate that gave acceptable optical soundtrack quality, chosen to save film stock. It remains the cinema rate and the reason "cinematic" and "24fps" have become entangled.

25 is European television, one frame per two cycles of 50Hz mains.

30 is American television, matched to 60Hz mains. When colour was added to NTSC in 1953, the frame rate was nudged to **29.97** to stop the colour subcarrier interfering with the audio carrier. That fractional rate is why American video timing is awkward to this day.

50 and 60 are the doubled rates, used for sport, for broadcast, and for anything where motion clarity matters more than the look of 24.

Drop-frame timecode is not dropped frames

At 29.97fps, one hour of video contains 107,892 frames, while an hour of timecode counting at 30fps expects 108,000. Over an hour, timecode drifts about 3.6 seconds ahead of the clock.

Drop-frame timecode fixes this by skipping certain *labels* — the numbers `:00` and `:01` at the start of each minute except every tenth minute. No frames are removed. It is a renumbering scheme so that timecode matches wall-clock time, which matters when a programme has to be exactly 28 minutes 30 seconds.

People lose hours to this. Nothing is missing; the counter skips.

Shutter angle, and what 180 degrees means

A film camera's rotating shutter is a disc with a slice cut out. A 180-degree opening means the film is exposed for half of each frame's duration: at 24fps, 1/48 second.

This is the **180-degree rule**, and it is a look rather than a law. Each frame carries motion blur covering half the interval between frames, so consecutive frames overlap in what they record and the eye reads continuous movement.

- **Shorter exposure** — 90 degrees, 1/96 at 24fps — gives crisp, blur-free frames that strobe visibly in motion. The opening sequence of *Saving Private Ryan* is the famous example.
- **Longer exposure** — 270 or 360 degrees — gives heavy blur and a dreamy, smeared quality.

On a digital camera it is a shutter speed, and the arithmetic is: **shutter speed = 1 / (2 × frame rate)** for the standard look. 1/50 at 25fps, 1/60 at 30fps, 1/120 at 60fps.

The practical consequence outdoors: matching 1/50 in bright sunlight needs a very small aperture or a neutral-density filter, which is why ND filters are standard kit rather than an accessory. Going to 1/500 to control exposure instead gives a strobing look you did not choose.

Judder, and why 24fps looks odd on a 60Hz screen

To show 24 frames a second on a 60Hz display, something has to be repeated. 60 does not divide by 24, so the standard scheme — **3:2 pulldown** — shows one frame for 3 refreshes and the next for 2, alternating.

The result is that consecutive frames are displayed for different lengths of time. A smooth pan then advances in uneven increments, which reads as a stutter: **judder**. It is most visible on slow horizontal camera movement, which is why cinematographers have a rule of thumb about how slowly a pan may cross the frame at 24fps before it falls apart.

Modern displays running at 120Hz can show 24fps cleanly — 120 is divisible by 24, five refreshes each — which is one real reason a film looks different on a high-refresh screen.

Changing frame rate, three ways

Conform. Play the frames at a different rate. 25fps material played at 24 is 4 per cent longer, with correspondingly lower-pitched audio unless it is corrected. This is the traditional film-to-PAL method, in reverse.

Duplicate or drop frames. Cheap, and it produces exactly the judder described above.

Interpolate. Generate new frames between the real ones by estimating motion. Modern optical-flow retiming is impressive and fails in characteristic ways: around fast-moving thin objects, at occlusion boundaries where something appears from behind something else, and on repetitive textures where the motion estimate is ambiguous. The artefacts look like melting or tearing.

For slow motion, the honest answer is to shoot at the higher rate. 120fps played at 24 is true five-times slow motion with real frames. Interpolating 24 up to 120 invents four frames out of every five, and on anything with complex motion it shows.

Mixed rates on one timeline

Dropping 60fps material into a 24fps project makes the editor decide what to do, and different editors default differently — some conform to slow motion, some drop frames. This is the source of a great deal of "why is this clip stuttering" confusion.

Set the project rate first, deliberately, and convert anything that does not match on the way in rather than letting the timeline improvise.

Free tools

ffmpeg handles all of it:

```
# conform: relabel 25fps as 24fps, no new frames, 4% longer
ffmpeg -i in.mov -r 24 -filter:v "setpts=25/24*PTS" out.mov

# interpolate 24 to 60 with motion estimation
ffmpeg -i in.mov -vf "minterpolate=fps=60:mi_mode=mci" out.mov
```

DaVinci Resolve's free edition has optical-flow retiming that is genuinely good. **Blender**'s video sequencer and **Kdenlive** both do frame-rate conversion. **RIFE** and similar open-source interpolation models often beat classical optical flow on difficult motion and are free to run locally.

The limitation

There is no correct frame rate, only conventions and what an audience is used to. The reaction to *The Hobbit* at 48fps — that it looked like television or a stage play rather than a film — is the clearest evidence that frame rate carries learned meaning rather than an objective quality ranking. Higher is not better; it is different, and the difference is cultural as much as perceptual.

Interpolation also has a hard limit that no model removes: information genuinely absent from the source cannot be recovered, only invented plausibly. When the invention is wrong it is wrong in a way that draws attention, which is why broadcast and cinema still prefer to shoot at the rate they intend to show.

BEFORE YOU MOVE ON

A director wants true slow motion in a 24fps project. One editor proposes shooting at 120fps and conforming; another proposes shooting at 24fps and interpolating up. What is the substantive difference?

A. Interpolation preserves the 180-degree shutter relationship while conforming from 120fps does not, so the interpolated version will look more natural in motion.

B. Conforming from 120fps changes the audio pitch and requires a separate sound pass, whereas interpolation leaves the audio untouched, which is the main practical distinction between the two approaches.

C. Interpolation has to be done at the source resolution while conforming can be done at any resolution, so the 120fps route allows a higher-quality master.

D. The 120fps route has five genuinely photographed frames per source frame, while interpolation invents four of every five and fails at occlusion boundaries and on repetitive texture.

44 · 10 min

Your hero video is why the page feels slow

THE ONE THING TO KEEP

File size follows how much changes between frames, not how many pixels are in one — so bitrate and content, not resolution, are the levers.

The hero video is why the page feels slow

A 40 MB autoplaying video at the top of a landing page is a common and expensive mistake. On a 3G connection in a place where data costs money, it is not slow — it is a decision to lose that visitor.

Most people trying to shrink a video only touch resolution. Resolution is the least useful of the three dials.

The three dials

Resolution — the pixel dimensions. Dropping 1080p to 720p helps, and on a phone-sized display nobody sees it.

Bitrate — how many bits per second the encoder is allowed to spend. This is the real size lever. Fifteen seconds of 1080p H.264 at 8 Mbps is about 15 MB. The same clip at 3 Mbps is under 6 MB and, watched on a phone, is frequently indistinguishable.

Codec — the compression scheme itself.

Why some clips compress and others do not

Video compression stores one complete frame and then, for the frames that follow, only what changed. This is the whole trick, and it explains almost everything about file sizes.

A talking head against a plain wall changes very little between frames, so there is almost nothing to store, and it compresses to nearly nothing. Rain, confetti, water, foliage in wind, fast pans and film grain change every pixel every frame. There are no cheap differences left, and the encoder has to spend real bits. Two clips at identical settings can differ by six times in size, and neither one is broken.

Two practical consequences. Adding grain or noise for a filmic look multiplies your file size. And a slow gradient — a clear sky, a soft background — will band into visible steps at low bitrates, because gradients are exactly what a quantiser throws away first.

The complete frames are called keyframes, and the interval between them is worth setting to about two seconds for anything streamed, so a viewer who scrubs does not wait. Shorter intervals mean bigger files.

Getting the file smaller

Which codec

- **H.264** plays on everything, including old phones and old browsers. Default choice.
- **H.265 / HEVC** is roughly 30–50 per cent smaller for the same quality, but it is patent-encumbered and browser support is inconsistent.
- **VP9** is well supported in browsers, royalty-free, and slow to encode.
- **AV1** is the smallest and the slowest to encode; decoding is now widespread but not universal on older Android devices.

For a website, serve H.264 and, if you can be bothered to provide two sources, VP9 or AV1 as the first option with H.264 as the fallback.

A command that does the job

HandBrake is free, has a graphical interface, and will get you most of the way with the "Web Optimized" presets. **ffmpeg** is free and does everything. Resolve's free Deliver page has sensible YouTube and web presets.

```
ffmpeg -i in.mov -vf scale=1280:-2 -c:v libx264 -crf 24 -preset
slow \
  -pix_fmt yuv420p -movflags +faststart -an out.mp4
```

- `-crf 24` sets quality rather than a fixed bitrate; lower is better and bigger, and a change of about 6 roughly halves or doubles the size. Try 20 for something detailed, 28 for a plain talking head.
- `-preset slow` spends more time encoding to get a smaller file. It costs you minutes and saves megabytes.
- `-pix_fmt yuv420p` is required for playback on a lot of hardware. Files that play on your Mac and fail everywhere else are usually missing it.
- `-movflags +faststart` moves the file's index to the beginning so the video starts playing before it has finished downloading. On a slow connection this is the difference between a video that starts and one that appears broken.

- `-an` drops the audio track. If the video is silent, that is free bytes.
- `scale=1280:-2` sets the width and lets the height follow, rounded to an even number, which the encoder requires.

Stop using GIF

A GIF has a 256-colour palette and no modern inter-frame compression. A three-second GIF that comes to 5 MB is the same three seconds an MP4 delivers in 200 KB, with better colour and no dithering.

Replace it with a muted, looping video:

```
<video autoplay muted loop playsinline poster="first-
frame.jpg"></video>
```

`playsinline` is the attribute people forget. Without it, iOS takes the video fullscreen instead of playing it in place.

For transparency on the web, you need WebM with VP9 alpha for Chrome and Firefox and HEVC with alpha for Safari, served as two sources — or, if the content is vector rather than photographic, a Lottie file instead, which is usually the better answer.

Everything above is about delivery. The cutting, pacing and grading that come before it are video editing's subject, not this one's.

Two habits

Set `preload="none"` and a poster image on anything below the fold, so a visitor on mobile data does not download a video they never scroll to.

And judge quality on a phone at arm's length, in daylight, not on your monitor zoomed to 200 per cent. Almost every video on the internet is over-encoded for a viewing condition nobody experiences.

BEFORE YOU MOVE ON

Two fifteen-second 1080p clips are exported with identical settings from the same project. The talking head comes out at 4 MB; a shot of falling rain comes out at 26 MB. What explains it?

A. The rain clip contains far more distinct colours, and colour information is what dominates video file size.

B. The encoder raised the effective resolution of the rain clip to preserve fine detail.

C. Compression stores one full frame and then only the differences; in rain almost every pixel changes every frame, so there are few cheap differences left and the encoder has to spend real bits.

D. The talking head has been over-compressed and has quietly lost quality that the rain clip retained.

45 · 10 min

The tool under most of the others

THE ONE THING TO KEEP

Options before -i apply to reading and options after apply to writing, -c copy trims and joins with no decode and no loss, and -movflags +faststart moves the MP4 index to the front so a web video can start before it has finished downloading.

One tool under most of the others

ffmpeg is the free software that decodes, encodes, filters and converts essentially every audio and video format. It is also what a great many commercial applications use internally, which means learning it is not learning an alternative to the professional tools — it is learning the layer most of them sit on.

The command line puts people off. It is worth twenty minutes, because a handful of commands cover most of what an editor is slow at.

The shape of a command

```
ffmpeg  [input options]  -i input  [output options]  output
```

Options before `-i` apply to reading the input; options after apply to writing the output. Getting that the wrong way round is the most common beginner error, and `-ss` is where it shows: before `-i` it seeks quickly, after `-i` it decodes and discards.

Trim without re-encoding

```
ffmpeg -ss 00:01:30 -i in.mp4 -t 00:00:45 -c copy out.mp4
```

No decode, no encode, no quality loss, done in under a second on a long file. The cut lands on the nearest keyframe, so the start may be up to a GOP early. For a precise cut you have to re-encode:

```
ffmpeg -ss 00:01:30 -i in.mp4 -t 00:00:45 -c:v libx264 -crf 18
-c:a aac out.mp4
```

Scale, with the trap included

```
ffmpeg -i in.mp4 -vf "scale=1280:-2" -c:v libx264 -crf 21 -c:a
copy out.mp4
```

`-2` means "work out the height to preserve the aspect ratio, and make it divisible by 2". `-1` preserves aspect but can produce an odd number, which H.264's 4:2:0 chroma cannot represent — that is the source of the "width not divisible by 2" error everybody meets once.

For a high-quality downscale, add `flags=lanczos`. For upscaling, nothing in ffmpeg invents detail; a good scaler is just less soft than a bad one.

Join clips

Same codec and parameters, no re-encode:

```
printf "file 'a.mp4'\nfile 'b.mp4'\nfile 'c.mp4'\n" > list.txt
ffmpeg -f concat -safe 0 -i list.txt -c copy joined.mp4
```

Different sources have to be re-encoded, via the `concat` filter, and it is worth normalising them to the same resolution and frame rate first rather than letting the filter improvise.

Extract and rebuild frames

```
ffmpeg -i in.mp4 -vf fps=12 frames/f_%04d.png
ffmpeg -framerate 12 -i frames/f_%04d.png -c:v libx264 -crf 18
-pix_fmt yuv420p out.mp4
```

`-pix_fmt yuv420p` on the way out is not optional if the file has to play in browsers and on televisions; without it, ffmpeg may write 4:4:4 from PNG input and many hardware decoders will refuse it.

Audio jobs

```
# extract audio untouched
ffmpeg -i in.mp4 -vn -c:a copy audio.m4a

# replace the audio track
ffmpeg -i video.mp4 -i new.wav -map 0:v -map 1:a -c:v copy -c:a
aac -shortest out.mp4

# normalise loudness to -14 LUFS, two-pass
ffmpeg -i in.wav -af
loudnorm=I=-14:TP=-1.5:LRA=11:print_format=summary -f null -
ffmpeg -i in.wav -af
loudnorm=I=-14:TP=-1.5:LRA=11:measured_I=...:measured_TP=...
out.wav
```

`loudnorm` in a single pass does a dynamic normalisation that can pump. Two-pass — measure, then apply the measured values — does a linear gain change and is what you want. The mixing module covers what those numbers mean.

Inspecting a file

```
ffprobe -v error -show_streams -show_format in.mp4
ffprobe -v error -select_streams v:0 -show_entries \
  stream=codec_name,width,height,r_frame_rate,pix_fmt,col-
or_range,bit_rate \
  -of default=nw=1 in.mp4
```

Most delivery arguments end the moment somebody runs this. **MediaInfo** presents the same information in a graphical window and is also free.

Three flags worth memorising

- `-movflags +faststart` — moves the MP4 index to the front so the file streams. Non-negotiable for web.
- `-pix_fmt yuv420p` — maximum compatibility.

- `-c copy` — no re-encode, no loss, near-instant. Try it first for anything that does not change the pictures.

Batching

A loop over a folder is where ffmpeg earns its keep against a graphical editor:

```
for f in *.mov; do
  ffmpeg -i "$f" -vf "scale=-2:720" -c:v libx264 -crf 22 \
    -c:a aac -b:a 128k -movflags +faststart "web/${f%.mov}.mp4"
done
```

Forty files while you make tea. The same job in an editor is forty exports and forty chances to get one setting wrong.

The free path, and the front ends

ffmpeg itself is free under LGPL or GPL depending on build. **HandBrake** wraps the same encoders with a graphical interface and sensible presets. **Shutter Encoder** is a free front end aimed at editors with named jobs rather than flags. **MediaInfo** reads files. **Losslesscut** trims with `-c copy` behind a timeline interface. None of these is a lesser version of a paid tool; they are the tool.

The limitation

ffmpeg does exactly what you say, which includes silently doing something unhelpful. Omit `-c:a` and it re-encodes your audio; omit `-pix_fmt` and it may write something unplayable; get the order of `-ss` wrong and you wait ten minutes for a seek. It has no preview, no undo, and its error messages assume you know the vocabulary.

It also will not make creative decisions. It cannot tell you that the CRF is too high for this footage or that the crop lost somebody's head. It is a precise instrument for jobs you can already specify, and the judgement stays with you.

BEFORE YOU MOVE ON

A team reports that their web video does not begin playing until it is fully downloaded, although the server supports range requests and the file is encoded at a modest bitrate. What is the most likely cause?

A. The keyframe interval is too long, so the player has to buffer an entire GOP before it can present the first frame.

B. The moov atom holding the file's index was written at the end, so the player cannot begin decoding until it has the whole file.

C. The pixel format is 4:4:4 rather than yuv420p, so the browser is falling back to a software decoder that buffers the stream before playback.

D. The file was encoded with two-pass rate control, which writes a variable-bitrate stream whose per-segment sizes are only knowable from the completed log, so a player performing range requests cannot determine which byte range corresponds to the opening seconds until it has read to the end.

CASE STUDY · MODULE 5

Two hundred lessons, rejected for milky blacks

A four-person e-learning outfit in Bhopal delivering a Class 10 science series to a state education portal.

Two hundred videos, eight to twelve minutes each, graded in Resolve and exported as H.264. The upload took a fortnight on a ten-megabit line. The portal's reviewer rejected the whole batch in one line: washed out, low contrast, the blacks are grey.

On every machine in the Bhopal office, and in VLC, the files looked correct.

The cause took an hour to find and cost nothing to state. The export had written no colour tags at all — no `colr` atom, no range flag, nothing in the bitstream's video usability information. The files were limited range, with black at sixteen and white at two hundred and thirty-five. The reviewer's player, faced with an untagged file, assumed full range and treated sixteen as a real sixteen, so every black sat above zero. Milky means limited data read as full. Nothing in the files was damaged; the numbers were right and the description of them was missing.

Three ways forward, none free.

The first was to re-tag. `ffmpeg -c copy` with the four colour flags writes the description into the container with no decode and no re-encode, about two minutes a file, and no generational loss at all. The processing was trivial. The re-upload was not: another fortnight on the same line, three weeks before term started.

The second was what the producer wanted. Take the reviewer's screenshot as the truth, add contrast, crush the blacks until it matched his player, re-export everything. Two days of machine time and the same fortnight of uploading. It would pass the review. It would also make every file wrong in every correctly-behaving player, including the portal's own Android app, which honours the tags — the same footage would then arrive crushed in the shadows and clipped in the highlights for the students it was made for.

The third was to argue: send a MediaInfo readout, ask the reviewer to open the file in a second player. Free, and it puts a four-person supplier in the position of telling a state portal that its review is wrong.

There was a fourth thing they considered and rejected, which is worth recording because it is the one most teams reach for. They could have re-exported with the tags written, rather than copying the existing streams. It would have been the same upload and roughly the same calendar time, and it felt tidier — one clean pass, one set of settings, everything consistent. It would also have put two hundred already-compressed lessons through a second generation of lossy encoding, spending real quality on a problem that was not in the pixels at all. Every encode quantises, and the second encoder would have been working to reproduce the first one's block edges as if they were detail.

The argument inside the office was not about the physics, which nobody disputed. It was about whose fault the rejection looked like. Sending a corrected file with an explanation reads as a supplier fixing its own error; sending a MediaInfo readout reads as a supplier correcting a client. They wrote the note four times before sending it, and the version that went out did neither — it showed the tags missing and present, said what each player would do with each, and asked him to try a second player before they re-uploaded the remaining hundred and ninety-nine.

They did the first and the third together, in that order and on one file.

STOP HERE

Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

They re-tagged a single lesson, uploaded it, and sent it with a two-line note and a MediaInfo screenshot showing the tags absent before and present after. The reviewer opened it in a second player, saw the same picture in both, and accepted the batch on condition the rest were tagged identically. The re-up-

load still took eleven days, which is the part nobody could compress. The outfit added four flags to its export script — bt709 for colourspace, primaries and transfer, and `tv` for range — and a check step that runs ffprobe on every finished file and refuses any that reports `unknown`. It costs nothing per file and has caught two since, both from a converter that strips tags. The producer, who had wanted to re-grade, now puts it in the plainest terms available: the values were correct and the label was missing, and changing the values to suit one player's guess would have been the only way to actually break them.

Milky or crushed — which way round was this failure, and how do you know without opening the file?

Milky means a limited-range file read as full range: the player takes sixteen for a real sixteen, so blacks sit above zero and the contrast collapses. Crushed and overcontrasty is the opposite error, a full-range file read as limited, where everything below sixteen clamps to black and above two hundred and thirty-five to white. The description of the symptom is the diagnosis.

Re-grading would have passed the review. Why was it the worse fix?

It changes the pixel values to compensate for one player's guess about a missing label. Every correctly-behaving player — including the portal's own app — would then show crushed shadows and clipped highlights, and the fault would be much harder to find because the file would be authoritative and wrong rather than merely untagged. It converts a labelling error into a data error.

Why does re-tagging with `-c copy` matter across two hundred files?

There is no decode and no encode, so it is near-instant and there is no generational loss. Every lossy encode quantises, and a second encode spends bits reproducing the first one's artefacts. Re-encoding two hundred already-compressed lessons to change metadata would cost real quality to fix a problem that is not in the pixels.

MODULE 5 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 Two 1080p clips encoded at identical settings come out six times different in size. What best explains it?

A. One was encoded with a faster preset than the other
B. One is 4:2:0 and the other is 4:4:4
C. One is rain on water and the other is a talking head against a wall
D. One has a keyframe every two seconds and the other one every ten seconds

2 Why does scrubbing camera footage lag while the same footage in ProRes responds instantly?

A. A P-frame is defined by its neighbours, so frame 43 decodes from the keyframe
B. ProRes files are smaller and therefore load faster from disk
C. ProRes stores a decoded preview thumbnail for every frame inside the file header
D. Long-GOP files use variable frame rate, which editors handle poorly

3 At a fixed CRF, what does changing the preset from medium to slow actually change?

A. The quality target, which rises as the preset slows
B. The bitrate ceiling, which the encoder is then allowed to exceed
C. The number of B-frames, which is what actually makes the picture look cleaner
D. How hard the encoder searches, so the file gets smaller at the same quality

4 An export looks crushed and over-contrasty in a player. Which error is that?

A. A limited-range file being read as full range
B. A full-range file being read as limited range
C. A 10-bit file being decoded by an 8-bit hardware decoder
D. A file tagged BT.601 being displayed on a BT.709 monitor

5 A sky that runs from level 40 to level 90 across 1080 pixels bands visibly. Why?

A. The encoder's deblocking filter has smoothed the gradient into visible steps
B. Fifty levels across 1080 pixels makes each step about 21 pixels wide
C. The file was tagged BT.709 when the camera recorded BT.2020
D. An 8-bit file cannot represent more than 256 colours in total

6 A web video only begins playing once it has fully downloaded. What is missing?

A. The yuv420p pixel format, without which browsers refuse to decode
B. A shorter keyframe interval, so the player can start sooner
C. The preload attribute set to auto on the video element
D. The faststart flag, which moves the MP4 index to the front

MODULE 6

Capturing sound

Recording a voice in a room you cannot rebuild: what the two microphone constructions can and cannot do, why distance beats equipment, where the level should sit and why the two ends of that range are not symmetrical, what the noise floor costs you three steps later, and how to get two devices into sync and into phase.

> BY THE END OF THIS MODULE YOU CAN
>
> Record a usable voice in an ordinary room — choosing pattern and distance from what the room is doing rather than from a specification sheet, setting gain so peaks land with headroom you can justify, capturing room tone as a matter of routine, and bringing a second microphone or a second device in without comb filtering or drift — and say for any given capture fault whether it can be repaired afterwards or not

46 · 10 min

Pressure, gradient, and everything that follows

THE ONE THING TO KEEP

A sealed capsule measures pressure and is omnidirectional with no proximity effect, while an open one measures the front-to-back difference and gains both directionality and a close-up bass lift — and a shotgun's interference tube only works where the tube length is comparable to the wavelength.

Two ways to detect sound, and the difference decides everything

A microphone converts air pressure variation into voltage. There are two arrangements, and almost every practical characteristic follows from which one you have.

Pressure microphones have a sealed capsule: the diaphragm is exposed to the air on one side only. It responds to absolute pressure at that point in space. Pressure is a scalar with no direction, so a pressure microphone is **omnidirectional** — it hears equally from all sides, at least until the capsule itself becomes large compared with the wavelength.

Pressure-gradient microphones are open on both sides. The diaphragm responds to the *difference* in pressure between front and back, which depends on the direction the sound came from. Sound arriving from the side reaches both faces at once, cancels, and is rejected. This gives **figure-of-eight**; mixing gradient with a little pressure response gives **cardioid** and **supercardioid**.

Everything that follows — proximity effect, off-axis colouration, wind sensitivity, the rear lobe — is a consequence of that choice.

The patterns, and what to use them for

- **Omnidirectional.** Flat response, no proximity effect, least wind and handling noise, and it hears the room. Excellent when the room is good or when you can be close. This is why lavalier microphones are usually omni: they are 20cm from the mouth, where the direct sound dominates anyway.
- **Cardioid.** Rejects the rear. The general-purpose choice for a voice in an imperfect room.
- **Supercardioid** and **hypercardioid.** Narrower front, but they have a **rear lobe** — a zone of renewed sensitivity directly behind. Point one at a speaker in a room and you are also recording whatever is behind you. People place monitors and noise sources by cardioid logic and then wonder where the bleed came from.
- **Figure-of-eight.** Full sensitivity front and back, deep null at 90 degrees. The null is the useful part: it is much deeper than a cardioid's rear rejection, so a figure-of-eight aimed so its null points at an air-conditioning unit is a powerful tool.
- **Shotgun.** An interference tube in front of a supercardioid capsule. Sound arriving off-axis enters through slots along the tube at different phases and partly cancels. This only works where the tube length is comparable to the wavelength, so a 30cm tube gives useful directivity above roughly 1kHz and almost none below 500Hz. That is the honest description of a shotgun: directional at high frequencies, broadly cardioid at low ones.

The consequence indoors is important. A shotgun rejects off-axis *direct* sound but reflections arrive from all directions and at all frequencies, so the tube colours them unevenly. A shotgun in a reflective room often sounds worse than a plain cardioid — hollow and boxy — which is why location sound crews carry both and use the shotgun outdoors.

Proximity effect, which you can use deliberately

Move a pressure-gradient microphone close to a source and the bass rises. Within about 5cm, the boost below 100Hz can be 6 to 10dB.

The mechanism: at a distance, the front-to-back pressure difference comes mostly from the wave's travel time between the two faces, which is frequency-dependent in a way the capsule design compensates for. Close up, a second effect dominates — the sound's amplitude itself falls off sharply across those few centimetres because of the inverse square law, and that difference is frequency-independent. The result is a low-frequency lift that grows as you approach.

Two things follow. **Omnidirectional microphones have no proximity effect at all**, because they have no gradient — which is why a lavalier does not get boomy when it rides up the chest. And the radio announcer's voice is a cardioid at 5cm with the bass boost fully engaged; it is a technique, not a microphone.

Sensitivity, and why a dynamic needs more gain

A dynamic microphone moves a coil through a magnet: rugged, no power needed, and comparatively insensitive — a Shure SM58 puts out about −54.5 dBV for 1 pascal of sound pressure. A condenser uses a charged capsule and a built-in amplifier, needs 48V phantom power, and typically produces around −37 dBV/Pa, roughly 20dB more.

Practically: a dynamic microphone on a quiet speaker asks for 55 to 65dB of preamp gain, and at that setting the preamp's own noise becomes audible on cheap interfaces. This is the real reason in-line gain boosters exist and why quiet-voiced podcasters struggle with a budget interface and a dynamic microphone. A condenser at 35 to 45dB of gain asks much less of the preamp.

What no microphone can do

It cannot separate you from your room. The capsule sums everything arriving at that point in space — your voice, the reflection off the desk 3ms later, the fridge, the road. Directionality changes the weighting by some decibels; it does not isolate.

This is the connection to the next lesson: the ratio of direct sound to room sound is set almost entirely by **distance**, and a more expensive microphone does not change it.

The free path

Nothing in this lesson requires buying anything to learn. **Audacity** and **Ardour**, both free, record and let you compare patterns if you have a multi-pattern microphone or two different ones. **REW** (Room EQ Wizard) is free and will measure a microphone's response against another one. A phone's own microphone is an omni pressure capsule, and comparing a phone recording at 10cm with one at 1m demonstrates the distance argument for nothing.

The limitation

Published polar patterns are measured in an anechoic chamber with a sine sweep and shown at a few frequencies. Real patterns vary substantially with frequency: most cardioids are nearly omnidirectional below 200Hz because the capsule is small compared with the wavelength, and become narrow and uneven at high frequencies.

So "cardioid" describes the midrange and misleads about both ends. The practical version: you cannot reject low-frequency noise by pointing a microphone away from it. Rumble, traffic and air handling arrive regardless of aim, and the only remedies are distance, isolation and a high-pass filter.

BEFORE YOU MOVE ON

A lavalier microphone clipped 20cm from a speaker's mouth does not become boomy when the speaker leans forward, while a cardioid on a boom at the same distance does. Why?

A. The lavalier's smaller capsule has a higher resonant frequency, so its response falls away below 150Hz and the proximity boost lands where there is no sensitivity to boost.

B. The lavalier is a pressure microphone with no front-to-back gradient, and proximity effect is a property of gradient operation.

C. The lavalier sits on the chest rather than in front of the mouth, so it is outside the near field where proximity effect occurs.

D. Lavaliers are manufactured with a fixed high-pass filter in the capsule housing specifically to counteract proximity effect at typical chest-mounting distances, which is why they sound thin when used at a distance.

47 · 10 min

People forgive a soft picture and leave over bad sound

THE ONE THING TO KEEP

Distance and soft furnishings set the ratio of voice to room; the microphone's price is not on that list.

People forgive a soft picture and leave over bad sound

Watch any two videos on a phone speaker. The one shot on an old camera with clean audio holds you. The one shot beautifully with the microphone across the room does not, and you will not be able to say exactly why you closed it.

Grain and softness read as authenticity. Bad sound reads as effort — the listener is doing continuous work to reconstruct words from a smeared signal, and that work is tiring in a way that has nothing to do with taste. They leave before they consciously decide to.

If you have limited time, spend it on the sound. This is the least glamorous advice in the course and the most reliably true.

The room beats the microphone

A microphone picks up two things: the sound travelling straight from your mouth, and the same sound arriving late after bouncing off walls, floor, ceiling, desk and window.

Direct sound weakens quickly with distance — roughly 6 dB for every doubling. Reflected sound barely weakens at all, because it arrives from everywhere at once. So the ratio of clean voice to room is set by two things: how close you are, and how reflective the room is. The microphone's price is not on that list.

This is why a four-hundred-dollar microphone at one metre in a tiled kitchen sounds worse than a phone at twenty centimetres under a duvet. It is the single most useful fact in beginner audio, and it costs nothing to act on.

Distance and angle

- About a fist's width from your mouth — 15 to 20 cm.
- Slightly off to the side, maybe 30 degrees, so the blast of air from p and b sounds passes the capsule rather than hitting it. A sock over a wire coat hanger works as a pop filter if you want one.
- Directional microphones get bassier the closer you get. That is the proximity effect. Some people like it; you can roll it off later.

The cheapest real improvement is soft mass

Not acoustic foam. Soft, thick, heavy things.

A duvet, a blanket, a full wardrobe of hanging clothes, a bed, curtains, a sofa, a rug. Put them on the surfaces facing you and facing the microphone — the

wall behind the microphone and the wall behind you matter most. Bare parallel walls and a hard floor are what you are fighting.

Recording inside a wardrobe with clothes in it is not a joke. It is the standard budget vocal booth, and it beats most treated rooms for a single voice.

Here is the reason it matters so much: **reverb cannot be fixed afterwards.** Noise reduction works by learning the pattern of a steady, unchanging sound and subtracting it. Reverb is not a separate sound to subtract — it is your own voice arriving a few milliseconds late, with the same frequencies. De-reverb tools have improved a great deal and still leave a hollow, gated quality on anything heavy. Solve it by moving closer and adding soft mass, not by processing.

Levels, phones and room tone

Levels

Record so the loudest peaks land around −12 to −6 dBFS. Not near zero.

Digital clipping is not the warm saturation of analogue tape. It is a flat square edge where the waveform should have been, and nothing recovers it. A recording that is too quiet can be raised. A clipped one cannot be repaired. If your recorder offers 32-bit float, this problem largely disappears — but assume it does not until you have checked.

Recording on a phone

Most readers of this will do exactly that, and it works.

- Use a voice recorder app that saves at a decent bitrate rather than recording video and taking the audio from it.
- The microphones are usually on the bottom edge. Do not put your hand over them.
- Airplane mode, so a radio burst does not appear in the file as a buzz.
- Fridge off, fan off, window shut. Then record thirty seconds and listen on headphones before you commit to a take.
- Put a folded towel under the phone so desk knocks do not travel into it.

- In a large or echoey room, a wired earbud microphone held at chest height often beats the phone's own microphone, purely because it is closer to your mouth.

Cheap lavalier microphones are genuinely good value. Their failure is cloth rustle: tape the cable in a small loop as strain relief, and keep it clear of collars and jacket zips.

Record room tone, every time

Thirty seconds of the room with nobody speaking, at the same levels, in the same position. It sounds like nothing. It is not nothing — the next two lessons both depend on it, and the moment you need it, going back to record it is impossible because the room has changed.

Ten minutes today

Record the same sentence twice: once at arm's length in your normal room, once at twenty centimetres with a coat draped over your head and the microphone. Listen to both on the worst speaker you own. You will not need the rest of this argument.

BEFORE YOU MOVE ON

Someone replaces their phone with a $400 microphone but records in the same tiled kitchen, standing a metre back. The result sounds worse than their old phone recordings made 20 cm from their mouth. Why?

A. At a metre the direct sound has dropped substantially while the reflections have not, so the ratio of clean voice to room has worsened. Distance and room reflectivity set that ratio, not microphone quality.

B. The new microphone has a wider pickup pattern than the phone's and is collecting sound from the whole room.

C. The recording level is too low for the new microphone, and raising the gain will bring the voice forward.

D. It is fixable in post — a noise reduction pass will subtract the room and leave the voice.

48 · 11 min

Direct sound, reverberant sound, and the distance between them

THE ONE THING TO KEEP

Direct sound falls 6dB per doubling of distance while the reverberant field stays roughly constant, so the ratio that makes a recording sound professional is set by distance — and a porous absorber only works for frequencies whose quarter-wavelength is comparable to its thickness.

The one number that decides how a recording sounds

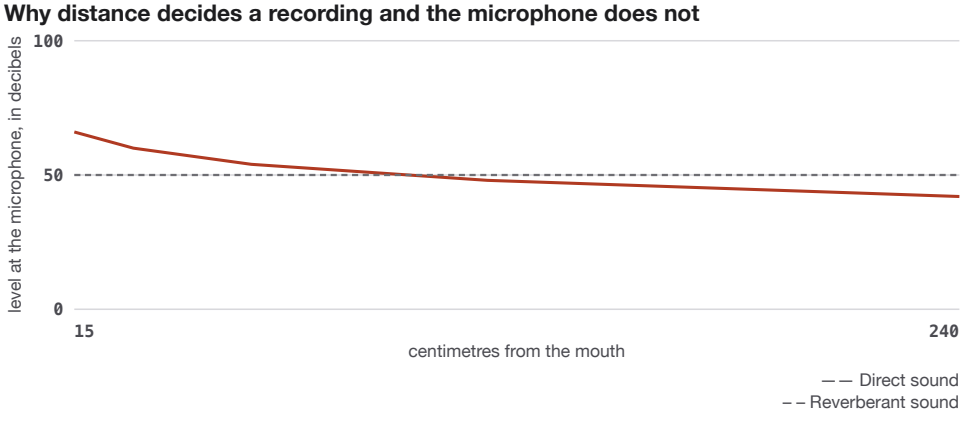
Every recording is a mix of two things: **direct sound**, which travelled straight from the source to the microphone, and **reverberant sound**, which bounced off surfaces first. The ratio between them is what people hear as "roomy" or "close", "amateur" or "professional".

The physics is simple and unforgiving.

Direct sound obeys the inverse square law: **every doubling of distance costs 6dB**. At 15cm from your mouth, then 30cm, then 60cm, then 1.2m, the direct sound falls by 6dB each step.

Reverberant sound does not behave that way. Reflections have bounced around and filled the room fairly evenly, so the reverberant level is roughly constant everywhere beyond a short distance from the source.

So as you move back, direct falls and reverberant stays. The ratio collapses. The distance at which they are equal is the **critical distance**, and in an untreated domestic room it is often only 0.5 to 1.5m. Beyond it you are recording mostly room.



Direct sound falls 6 dB for every doubling of distance. Reflections have filled the room fairly evenly, so the reverberant level barely changes. They cross at the critical distance — often well under a metre in an untreated domestic room — and past it you are recording mostly room. Moving from 60 cm to 15 cm buys 12 dB of that ratio, which is more than any microphone upgrade will sell you.

This is why the single most effective thing you can do costs nothing: **get closer**. Moving from 60cm to 15cm improves the direct-to-reverberant ratio by 12dB, which is a larger change than any microphone upgrade will give you.

Absorption and isolation are different problems

These get conflated constantly and the confusion wastes money.

Absorption (acoustic treatment) reduces reflections *within* a room. Soft, porous materials convert sound energy to heat as air moves through them. It makes a room sound less boxy. It does essentially nothing to stop sound getting in or out.

Isolation (soundproofing) stops sound passing between spaces. It requires mass, airtightness and decoupling — a heavy wall, sealed gaps, separated structures. Foam on the wall provides none of these.

A duvet over your head will make your voice sound dramatically better and will not reduce the neighbour's drill by a decibel. If your problem is a noise coming in, treatment is the wrong tool and there is rarely a cheap answer.

Why thin foam does not fix a boomy room

A porous absorber works by air movement, and air movement is greatest a quarter-wavelength from a hard surface. So an absorber is effective for frequencies whose quarter-wavelength is comparable to its thickness plus any air gap behind it.

Run the numbers. Sound travels about 343 m/s. At 100Hz the wavelength is 3.4m and a quarter of that is 86cm. At 1kHz the wavelength is 34cm and a quarter is 8.5cm.

So 5cm of foam absorbs well above roughly 1kHz, weakly around 250 to 500Hz, and almost nothing at 100Hz. The tiles on the wall are removing exactly the frequencies that were not the problem, while the boom — which is low-frequency room modes — remains untouched. This is why bass traps are thick, why they go in corners where pressure is highest, and why a serious low-frequency fix occupies real volume.

Thick porous absorbers with an air gap behind them work far better per pound than thin foam glued flat. A 10cm mineral-wool panel mounted 10cm off the wall behaves, roughly, like a 20cm absorber.

Flutter echo, and the cheapest test in audio

Stand in the middle of the room and clap once, sharply. Listen to what comes back.

- A short, even decay: good.
- A metallic ringing or a rapid "brrrp": **flutter echo**, sound bouncing between two parallel hard surfaces. Break it by putting something soft or angled on one of the two facing walls. One wall is enough.
- A boomy tail: low-frequency modes, which need thickness.

This takes two seconds, needs no equipment, and tells you more about a room than a specification sheet.

What actually works in a spare room

In rough order of effect per effort:

1. **Get closer to the microphone.** Free, and the largest single improvement.
2. **Do not stand in the middle of the room**, and do not face a bare wall closer than about a metre. Record into the room's long dimension.
3. **Put something soft behind the microphone**, in the direction it is facing past you — a duvet on a clothes rail, an open wardrobe of hanging clothes, a heavy curtain. You are absorbing the reflection that would otherwise return straight into the capsule.
4. **Break the two parallel pairs** — one soft or angled surface on one wall of each pair.
5. **Soft floor.** A rug removes the desk-floor-ceiling path.
6. **Thick panels in corners**, if you are building anything.

The wardrobe trick is genuinely effective and frequently mocked. Hanging clothes are a deep, porous absorber with an air gap, which is the correct construction; they work better than the foam tiles that cost more.

Measuring instead of guessing

REW (Room EQ Wizard) is free, runs on all three desktop platforms, and will measure your room's impulse response with a sweep played through a speaker. It gives you an RT60 figure — the time for sound to decay by 60dB — per frequency band, which turns "it sounds boxy" into "there is a 0.6 second tail at 160Hz".

For speech, an RT60 below about 0.4 seconds is comfortable and below 0.3 is good. Domestic rooms with hard floors commonly measure 0.6 to 1.0 seconds, and that is what makes a recording sound like a kitchen.

The free path

REW for measurement. **Audacity** or **Ardour** to record test clips and compare. Materials: duvets, curtains, a rug, an open wardrobe, and mineral wool

or rockwool in fabric-covered frames if you build panels — the material costs a fraction of branded acoustic products and performs measurably better than thin foam.

The limitation

Treatment has a floor. You can make a small room sound neutral; you cannot make it sound like a large one, because a small room's low-frequency modes are widely spaced and no amount of absorption removes the geometry. A 3-metre dimension supports a mode at around 57Hz and its multiples, and that is a property of the space.

And none of this addresses noise coming in. If a road, a flight path or a neighbour is the problem, acoustic treatment will not help, the honest options are expensive or involve moving, and the practical answer is usually to record at a different time of day.

BEFORE YOU MOVE ON

Someone covers a small room's walls with 5cm foam tiles and reports that the boominess is unchanged, although the room sounds less bright. What does the physics predict?

A. Foam absorbs by surface friction rather than internal air movement, so its effect depends on coverage area rather than thickness and the tiles were spaced too far apart.

B. The tiles have reduced the reverberation time so much at high frequencies that the low-frequency energy is now relatively more audible, and the absolute low-frequency level is unchanged.

C. 5cm of foam is effective above roughly 1kHz, and a 100Hz quarter-wavelength is 86cm, so nothing in the room is absorbing the frequencies causing the boom.

D. The tiles were mounted flat against the wall rather than on a batten with an air gap behind them, and a porous absorber mounted with any air gap at all performs identically across the whole spectrum regardless of its own thickness.

49 · 10 min

Where the level should sit

THE ONE THING TO KEEP

Clipping replaces the waveform's peak with a flat line carrying harmonics that were never in the source and cannot be recovered, while recording quietly only moves you closer to a noise floor that 24-bit has room to spare above — so the two ends of the meter are not symmetrical and headroom is cheap.

Where the level should sit, and why

Set your recording so **peaks land between −12 and −6 dBFS**, with the average of speech somewhere around −18. That is the whole recommendation, and the reasons for both ends of it are worth knowing because they are not symmetrical.

The top end is a hard wall. 0 dBFS is the largest number the format can represent. Exceed it and the waveform's peak is replaced by a flat line at the maximum value. That flat top is not a slightly-too-loud version of the sound; it is a different waveform, containing high-order harmonics that were never in the original. No processing recovers the shape, because the information is not attenuated — it is absent. Clipping is permanent.

The bottom end is soft. Recording 12dB quieter than you could have does not destroy anything. It uses fewer of the available code values, and in 24-bit there are so many that it does not matter, for reasons covered in the next lesson. The cost of recording too quiet is that you are closer to the noise floor, and the noise floor of a decent preamp is a long way down.

The asymmetry is the point: **too loud is fatal, too quiet is an inconvenience.** Leave headroom.

Where to apply gain matters

Signal chain: microphone → preamp → converter → file → software.

Gain applied at the **preamp** amplifies the microphone signal before the converter, so the signal arrives at the converter well above the converter's own noise. Gain applied **in software afterwards** multiplies the signal *and* everything the preamp and converter added to it.

So the rule is: get the level right at the preamp. A recording made at −40 dBFS and normalised to −12 afterwards has 28dB more of the preamp's hiss than one recorded at −12 in the first place. This is the single most common cause of a hissy recording made with adequate equipment.

The opposite error exists too: cranking the preamp to the point where it distorts, then turning it down digitally. The preamp's distortion is already in the signal.

What a meter is telling you

Two kinds of number, and people confuse them constantly.

Peak meters show the highest sample value. This is what you set to avoid clipping.

RMS or loudness meters show an average over time, which corresponds much more closely to how loud something sounds. Speech typically has 12 to 18dB between its average and its peaks — the crest factor — which is why speech peaking at −6 sounds much quieter than music peaking at −6.

And a sample-peak meter can miss a peak. The analogue waveform reconstructed between samples can overshoot the highest sample, by a decibel or more. **True peak** meters model that reconstruction. It matters at the delivery stage, where a file peaking at −0.1 dBFS by sample measurement can clip a lossy encoder's decoder, and it is why the loudness lesson later specifies a true-peak ceiling of −1 dB or lower.

32-bit float, and the caveat nobody mentions

Recorders advertising 32-bit float claim you cannot clip. The claim is half true and the half that is false is the important one.

In a 32-bit float file, values above 1.0 are representable, so a signal that would have clipped in a fixed-point file is stored intact and can be pulled down afterwards. That part is real.

But the **analogue front end** — the microphone's own maximum sound pressure level, the preamp's supply voltage, the converter's input stage — still has a physical ceiling. Recorders achieve this with dual converters at different gains, combined in the file, which extends the range enormously and does not make it infinite. Overload the capsule or the input stage and the distortion is captured faithfully into a float file that cannot be repaired.

32-bit float is an excellent insurance policy against a sudden shout. It is not a reason to stop setting gain.

Practical routine

1. Ask the speaker to talk at their loudest realistic level — the laugh, the emphatic bit — not their test-voice.
2. Set the preamp so that peaks land around −9 dBFS.
3. Check the quiet passages are comfortably above the noise floor; if they are near −50, either get closer or accept a quieter recording and a noisier lift.
4. Record ten seconds of silence for room tone before anybody speaks. It is free, it takes ten seconds, and every edit later will want it.
5. Watch the meter through the first minute. People get louder once they relax.

Limiters on the way in

A limiter at the input stops a peak from clipping by reducing gain fast when the level approaches a ceiling. Many field recorders include one.

For unrepeatable recordings — an interview, a live event — turn it on with a ceiling around −6 dBFS. The occasional couple of decibels of gain reduction on a laugh is far better than a clipped take. For controlled studio work, leave it off and set gain properly; a limiter is an insurance policy with a small premium, not a replacement for a level.

The free path

Audacity shows peak and RMS on its meters and has a clipping indicator you should turn on. **Ardour** and **Reaper** — Reaper is paid but has an unlimited evaluation period and a very low licence — offer full metering. **ffmpeg** can tell you what a finished file actually did:

```
ffmpeg -i take.wav -af
"astats=metadata=1:measure_overall=Peak_level+RMS_level" -f
null -
ffmpeg -i take.wav -af ebur128=peak=true -f null -
```

The second prints integrated loudness and true peak, both free, both exactly what a delivery specification asks for.

The limitation

Metering describes a file, not a performance. A recording at a perfect −9 dBFS peak of a voice 80cm from a microphone in a hard room is a well-levelled bad recording, and no meter says so. Level is the easiest thing to get right and the least important of the things that matter.

There is also no universal standard for what a meter shows. Different applications use different ballistics and different reference points, so two meters can legitimately disagree by several decibels on the same file. The numbers in this lesson are dBFS peak, which is unambiguous; anything described as "VU" or "dB" without a suffix is not.

BEFORE YOU MOVE ON

A recordist raises the preamp until occasional peaks distort, then reduces the track's level in software by 6dB and reports that it sounds fine. What is wrong with this reasoning?

A. Reducing level in software applies to the whole track, so quiet passages are now 6dB closer to the file's noise floor than they need to be.
B. The software reduction is applied after the fader's dither stage, so the 6dB cut reintroduces quantisation error in the reduced signal that the original recording did not have.
C. The distortion happened in the preamp before the converter, so it is part of the recorded signal and turning the whole thing down turns the distortion down with it.
D. A 6dB reduction in software is not equivalent to 6dB less preamp gain, because preamp gain is applied logarithmically to the microphone's voltage while digital attenuation scales sample values linearly, so the two operate on different curves.

50 · 10 min

Two numbers doing two different jobs

THE ONE THING TO KEEP

Sample rate sets the frequency ceiling and anything above it folds back as an inharmonic alias, while bit depth sets dynamic range at about 6dB per bit — which is why 24-bit makes headroom cheap and why dither belongs only at the final reduction.

Two numbers that do different jobs

A digital audio file is described by a sample rate and a bit depth, and they control entirely separate things. Sample rate sets the highest frequency that can

be represented. Bit depth sets the dynamic range between the loudest and quietest representable signal. Neither improves the other.

Sample rate and the folding problem

The sampling theorem says a sample rate of f can represent frequencies up to $f/2$, the Nyquist frequency. 44.1kHz gives a ceiling of 22.05kHz; 48kHz gives 24kHz. Both are comfortably above the usual upper limit of adult hearing, which is why 44.1 was chosen for the compact disc in the first place.

What happens to frequencies *above* the ceiling is the part worth understanding. They do not disappear. They **fold back** into the audible range, mirrored around the Nyquist frequency. A 25kHz tone sampled at 44.1kHz appears in the file as 44.1 − 25 = 19.1kHz — an audible tone that was never in the room, unrelated harmonically to anything else. This is **aliasing**, and because the folded frequencies bear no musical relationship to the source, it sounds harsh and metallic rather than merely bright.

Converters prevent it with a steep anti-aliasing filter before the sampler. That filter is the real reason people argue about sample rates: a brick-wall filter at 22kHz has consequences in the top octave, and running at 96kHz lets the filter be gentler and moves its effects well out of hearing.

48kHz, and why video uses it

48kHz is the standard for video and film. It is not audibly better than 44.1; it was chosen for professional video systems because it gives a whole number of samples per frame at the common broadcast rates, which makes editing and synchronisation arithmetic exact.

Use 48kHz for anything that will meet a picture. Converting 44.1 to 48 is a resampling operation — it is fine, but it is an extra process and a chance for a small error, and there is no benefit to starting at 44.1 for video work.

When higher rates genuinely help

Not for fidelity of the recording, for most purposes. There are three real cases:

- **Extreme pitch-shifting or time-stretching.** Slowing a 96kHz recording to a quarter speed brings content that was at 40kHz down to 10kHz, where you can hear it. Sound designers record at 96 or 192kHz for exactly this.
- **Non-linear processing.** Distortion, saturation and some compressors generate harmonics above Nyquist which alias back down. Good plugins oversample internally to avoid it; running the whole session higher is a blunter version of the same fix.
- **Very low latency monitoring**, because a buffer of a fixed number of samples is shorter in time at a higher rate.

Against those: double the storage, double the processing, and for a spoken-word recording, no audible gain. 48kHz is the correct default.

Bit depth is dynamic range

Each bit adds about 6.02dB of range. 16-bit gives roughly 96dB; 24-bit gives roughly 144dB.

144dB is larger than any analogue front end can deliver. The best preamps and converters have a noise floor around 120dB below full scale; most interfaces are nearer 100 to 110. So in a 24-bit recording, the bottom two to four bits are recording the analogue circuitry's noise, not silence.

This is exactly why the headroom advice in the previous lesson is cheap. Recording at −18 dBFS average rather than −6 costs you 12dB of the available range. In 16-bit that would be a real sacrifice; in 24-bit, you still have more range than your preamp can use. **Record in 24-bit and leave headroom** is the whole practical conclusion.

Dither, and why it matters going down

When you reduce bit depth — a 24-bit master to a 16-bit file — the extra bits have to be discarded. Rounding produces **quantisation distortion**, and unlike noise it is correlated with the signal, so it appears as a granular, crunchy texture on quiet fades rather than as steady hiss.

Dither is a tiny amount of random noise added before the truncation. It decorrelates the error: instead of a fade stepping down through discrete levels, it dissolves into noise. You have traded a signal-dependent distortion for a constant, benign noise around −90 dBFS, which is inaudible in almost any real listening situation.

The same mechanism as the video banding lesson, for the same reason.

Apply dither **once**, at the final reduction to 16-bit, and never twice. Staying in 24-bit or 32-bit float throughout means it never arises until the final export.

What to use, in practice

- **Recording:** 48kHz, 24-bit. Always, for anything meeting picture.
- **Working:** keep 24-bit files; work in 32-bit float internally, which every editor does by default.
- **Delivery to video:** 48kHz, and let the encoder handle the rest. AAC at 192 to 256 kbit/s stereo is transparent for speech and generous for music.
- **Delivery as WAV:** 48kHz 24-bit for masters, 16-bit with dither if something demands it.
- **Sound design source material:** 96kHz if you intend to stretch it.

The free path

Audacity records at any rate and depth, works internally in 32-bit float, and includes dither options in its export settings. **Ardour** is a full open-source recording environment. **ffmpeg** resamples with the high-quality `soxr` resampler:

```
ffmpeg -i in.wav -af aresample=resampler=soxr:precision=28 -ar
48000 out.wav
ffmpeg -i in.wav -af aresample=osf=s16:dither_method=triangular
-ar 44100 out16.wav
```

The limitation

Very little of this is audible on a spoken-word recording made in a normal room. The room, the distance and the microphone's placement account for nearly all of the perceived quality; the difference between 48 and 96kHz accounts for approximately none of it.

Claims about high sample rates sounding better are also genuinely contested. Controlled listening tests have repeatedly failed to show a reliable difference on playback of material within the audible range, while the practical benefits for processing are real and measurable. The two claims get conflated in marketing, and the honest position is to use higher rates where there is a process that benefits and not to expect a difference otherwise.

BEFORE YOU MOVE ON

Why is aliasing perceived as harsh and metallic rather than simply as unwanted brightness?

A. Aliased components are reproduced at a higher amplitude than the original content because the folding process sums energy from several frequencies onto one.

B. A frequency above the ceiling reappears mirrored around it, so it lands at a frequency bearing no harmonic relationship to the source.

C. The anti-aliasing filter introduces phase distortion in the top octave, and it is that phase shift rather than the folded content that is audible.

D. Aliased content is not band-limited on playback, so the reconstruction filter at the output stage cannot remove it and it interacts with the playback system's own distortion products to create intermodulation across the whole audible range.

51 · 11 min

What the noise floor costs you later

THE ONE THING TO KEEP

Room noise is diffuse and roughly constant while the voice obeys the inverse square law, so every halving of distance buys 6dB of signal-to-noise for nothing — and spectral subtraction leaves a residue of isolated tonal bursts because the noise it subtracts is an average of something random.

The cost of noise is paid later, not now

A recording with a quiet background hiss sounds acceptable on its own. The problem appears three steps downstream: you compress it and the hiss comes up with everything else, you cut between two takes and the background jumps, you add music and dialogue and the mix turns grey, and every attempt to fix it makes something else worse.

Noise is the constraint that limits what you can do to a recording afterwards. It is worth measuring rather than sensing.

Where it comes from, in the usual order of size

1. **The room.** Ventilation, a computer fan, a fridge, traffic, a light fitting's transformer. Almost always the dominant source in a domestic recording, and almost always underestimated because you stop hearing it within a minute of entering.
2. **The preamp.** Its own thermal and circuit noise, amplified with the signal. The figure to look for is equivalent input noise, typically −125 to −129 dBu for a good preamp; a cheap one at high gain will be audibly worse.
3. **The microphone.** A condenser's self-noise, published as an A-weighted figure. Under 15 dB-A is quiet; over 20 dB-A is audible on a soft voice.
4. **The converter.** Usually the smallest contributor in any modern interface.

The order matters because it tells you where to spend effort. Turning off the ventilation is free and typically wins 10dB. A better microphone might win 3.

Measure it in thirty seconds

Record ten seconds of the room with nobody speaking, then:

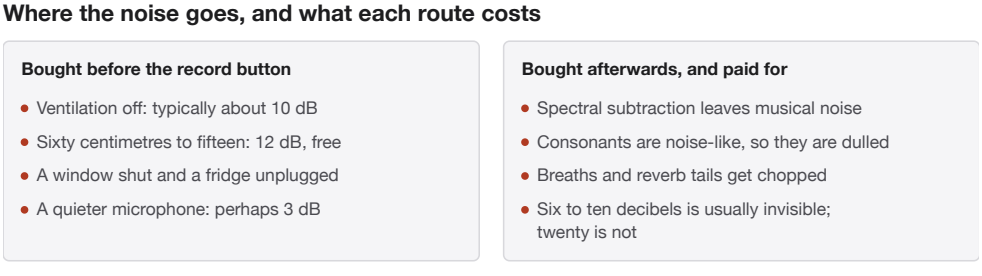
```
ffmpeg -i roomtone.wav -af "astats=measure_overall=RMS_level" -
f null -
```

Then measure the speech. The difference is your signal-to-noise ratio.

- **60dB or more** — very good, you can do anything to it.
- **45 to 60dB** — normal for a decent home setup; usable with care.
- **30 to 45dB** — the noise will be audible after compression and will rise in every quiet passage.
- **Under 30dB** — you are going to hear it, and denoising will cost you quality.

The inverse square law is a noise tool

Room noise is largely diffuse — it arrives from everywhere and its level does not change much as you move. Your voice obeys the inverse square law.



Room noise is diffuse and roughly constant wherever you stand; your voice obeys the inverse square law. So every halving of distance is 6 dB of signal-to-noise for nothing, and it is the only improvement on this page that costs neither money nor quality.

So halving the distance from your mouth to the microphone raises the voice by 6dB and leaves the noise where it was. **Every halving of distance buys**

6dB of signal-to-noise ratio, free. Going from 60cm to 15cm is 12dB — more than the difference between a budget microphone and a good one.

This is the same mechanism as the direct-to-reverberant argument, and it points the same way: distance is the most powerful control you have and it costs nothing.

Room tone, and why ten seconds saves an hour

Record the room in silence for at least ten seconds at the same gain, in the same position, before or after every setup.

It has two uses. First, **edits**. Cut a sentence out of an interview and the background stops for a frame; drop room tone underneath and the join disappears. Every dialogue editor does this and it is invisible when done and glaring when not.

Second, **a noise profile**. Every denoiser works better when given a clean sample of exactly the noise it is being asked to remove.

Ten seconds. Do it before anyone speaks, while everyone is still settling.

Why denoising leaves a strange texture

Classical noise reduction works by **spectral subtraction**. It measures the noise's average level in each of several hundred frequency bands, then reduces each band of the signal by that amount.

The noise is random, so in any given moment a band's actual noise level is above or below its average. Subtracting the average leaves a residual that is sometimes zero and sometimes a short isolated burst of energy in one band. Played back, those residuals are heard as brief tonal chirps appearing and vanishing — **musical noise**, and once you can name it you will hear it in every over-processed podcast.

Push the reduction harder and two further things happen. Consonants — s, t, k, f — are broadband and noise-like, so the algorithm treats them as noise and dulls them. And breaths and room reverb tails get chopped, which makes the voice sound as if it is being switched on and off.

Machine-learning denoisers, including the free **RNNoise** and the models built into several editors, are considerably better because they have learned what speech looks like rather than subtracting an average. They fail differently: they can remove non-speech content you wanted, smooth the voice into a slightly synthetic texture, and behave unpredictably on accents or languages under-represented in training.

The practical discipline is the same for both: **use the least reduction that solves the problem.** 6 to 10dB is usually enough and often inaudible. 20dB is nearly always audible and usually worse than the noise was.

Hum is a different problem

A steady tone at 50 or 60Hz with harmonics is mains hum, caused by a ground loop or an unbalanced cable near a power supply. It is not broadband noise and a denoiser is the wrong tool.

Fix it at the source: a balanced cable, a different socket, moving a power brick away from the microphone cable, or a ground-lift on a DI box. Failing that, a narrow notch filter at the fundamental and its harmonics removes it with far less collateral damage than broadband reduction.

The free path

Audacity has a noise-profile-based reduction — the classic spectral subtraction, with the residual characteristics described above. **RNNoise** is open source and available as an LADSPA plugin and inside several editors. **ffmpeg** has `afftdn` for spectral denoising and `anlmdn` for a non-local-means approach that is gentler on speech. **DaVinci Resolve**'s free edition includes its own voice isolation. **iZotope RX** is the paid industry standard and is named in job adverts; nothing here requires it.

The limitation

Denoising cannot add back what the noise masked. If a word sits at the same level as the hiss, removing the hiss removes the word with it. The information

about what was said is genuinely not separable from the noise at that point — they are the same samples.

And the most advanced tools are the hardest to evaluate, because a machine-learning denoiser produces confident, clean output whether or not the result resembles what was said. On a quiet or accented passage it can smooth a word into something else entirely, and the result sounds plausible. Anything that will be quoted, transcribed or relied on should be checked against the original rather than trusted because it came out clean.

BEFORE YOU MOVE ON

After heavy noise reduction, an interview sounds clean but is dotted with brief tonal chirps and the consonants have gone dull. What produced each symptom?

A. The noise profile was taken from a section that contained speech, so the algorithm is subtracting speech formants as well as noise.

B. Subtracting an averaged noise spectrum from a random signal leaves isolated residual peaks, and broadband consonants resemble noise closely enough to be attenuated with it.

C. The reduction was applied twice, and cascading two passes of spectral subtraction produces intermodulation between the two residuals.

D. The denoiser is operating on too short an analysis window, so its frequency resolution is insufficient to separate narrow noise components from speech harmonics, and lengthening the window would resolve both the chirps and the dulled consonants without any loss of reduction depth.

52 · 10 min

Two microphones and a comb filter

THE ONE THING TO KEEP

A path difference of d metres puts the first cancellation at 343/(2d) hertz and repeats at odd multiples, so a 34cm difference notches the speech band at 504Hz upward — and a polarity flip, being frequency-independent, cannot fix a difference caused by time.

Two microphones, one source, and a sound that gets worse

Add a second microphone to a voice expecting more of it, and it frequently sounds thinner, hollow, or strangely filtered. Nothing is broken. You have built a comb filter.

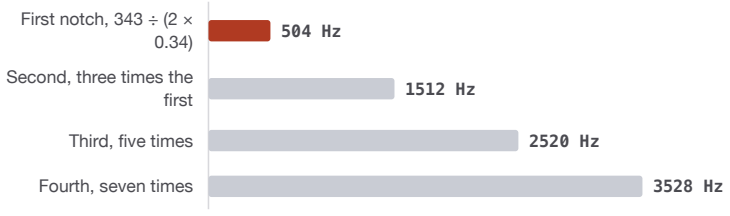
Sound travels at about 343 metres per second, so it takes time to reach each microphone. If one is 34cm further from the mouth than the other, its signal arrives about 1 millisecond later. When the two are mixed, some frequencies add and some cancel.

The frequencies that cancel are those for which the delay is half a wavelength, or an odd multiple of it:

```
first cancellation  f = c / (2 × path difference)
                      = 343 / (2 × 0.34)  ≈  504 Hz
```

Then again at three times that, five times, and so on — 504Hz, 1512Hz, 2520Hz. Plot the response and it looks like the teeth of a comb, which is where the name comes from. Those notches sit squarely in the range that carries speech intelligibility.

Where a 34-centimetre path difference cancels



Sound covers about 34 cm per millisecond, so a microphone a third of a metre further from the mouth hears everything about a millisecond late. Mixed together, the two signals cancel at these frequencies and add between them — a comb, with its teeth straight through the band that carries intelligibility. A polarity flip applies equally at every frequency, so it cannot undo a difference caused by time.

A single microphone has no path difference and no comb. That is why one well-placed microphone usually beats two badly-placed ones.

The 3:1 rule

The standard working rule for multiple microphones on separate sources: **the distance between two microphones should be at least three times the distance from each microphone to its own source.**

The reason is arithmetic rather than tradition. Three times the distance means the leaked sound arrives about 9.5dB quieter than the intended sound in that microphone. At that level, the comb filtering it causes when the channels are mixed is shallow enough to be inaudible.

Two people each 20cm from their own microphone should have their microphones at least 60cm apart. In practice this means: closer to your own microphone lets the microphones be closer together, which is convenient rather than restrictive.

Polarity is not phase, and people say phase

The button labelled **Ø** or "phase invert" flips the polarity: every positive sample becomes negative. It applies equally at all frequencies.

Phase differences caused by distance are frequency-dependent, because a fixed time delay is a different fraction of a wavelength at each frequency.

So flipping polarity does not fix a timing difference. It is the right tool in exactly one common case: two microphones on opposite sides of the same source — a snare drum top and bottom, a guitar amp front and back — where the diaphragms move in opposite directions on the same pressure wave. There, one is genuinely inverted and the flip is correct.

For a timing difference, the fix is to **move a microphone** or to **nudge one track in time** until the waveforms align. Every editor lets you slide a clip by samples; zoom in far enough to see the individual cycles and line up a transient.

Check it in mono, always

Comb filtering can hide in stereo. Panned apart, the two signals reach your ears separately and the cancellations are much less obvious. Sum to mono and they collapse into the same signal and the notches appear in full.

This matters because mono playback is not an edge case. A phone's single speaker, a laptop, a smart speaker, a Bluetooth speaker, many television soundbars in their default mode, and most social video on a phone — all mono. If your voice disappears in mono, it disappears for a large share of your audience.

Press the mono button on every mix, before delivery. It takes one click and catches an entire class of problem.

Stereo techniques that are designed not to comb

- **Coincident pairs** (XY, Blumlein, mid-side) put both capsules at effectively the same point, so there is no path difference and no comb filtering. They are perfectly mono-compatible and give a narrower stereo image.
- **Spaced pairs** (A/B) deliberately use time differences to create width. They are wide and they are not mono-safe.
- **Mid-side** deserves a mention because it is free width with guaranteed mono compatibility: one forward-facing microphone and one figure-of-eight sideways, decoded as mid+side and mid−side. Summing to mono cancels the side component exactly and leaves the mid microphone alone.

For a voice with a picture, you almost never want stereo at all. **Dialogue should be mono, centred.** A stereo voice moves when the camera cuts and smears in mono playback.

The lavalier-and-boom case

The common real situation: a lapel microphone and a boom on the same speaker, deliberately, for redundancy and for different textures.

They will comb if mixed at equal level. The standard approaches: use one as the primary and the other only where the first fails, cutting between them; or align them in time first — slide the boom track earlier by the path difference, which you can measure by looking at the waveforms — and then mix.

A useful measurement: sound covers roughly 34cm per millisecond. If the boom is a metre away and the lav is 20cm away, the boom is about 2.3ms late. Slide it, then mix.

The free path

Audacity, **Ardour** and **Reaper** all let you zoom to the sample and nudge a clip. **ffmpeg** can delay a channel precisely:

```
ffmpeg -i boom.wav -af "adelay=0|0,atrim=start=0.0023"
aligned.wav
```

Most editors also include an automatic alignment function that cross-correlates two tracks and lines them up; Resolve, Reaper and Audacity's alignment tools all do it free. For checking, any editor's mono-sum or a phase correlation meter — Ardour and Resolve both have one free — shows you whether two channels are fighting.

The limitation

Comb filtering from room reflections cannot be fixed this way. Every hard surface creates a delayed copy of your voice arriving at the microphone, so a desk

40cm below the microphone is producing a comb at about 430Hz whether you like it or not. You cannot time-align a reflection.

The only remedies are the ones from the room lesson: get closer so the direct sound dominates, absorb the surface, or angle it. This is the concrete reason a microphone on a bare desk sounds hollow, and the reason a cloth under the microphone measurably helps.

BEFORE YOU MOVE ON

A voice recorded with a lavalier at 20cm and a boom at 1m sounds hollow when the two are mixed equally. An engineer presses the polarity invert on the boom channel. Why does this not fix it?

A. Polarity inversion shifts every frequency by the same amount, while the problem is a fixed time delay that corresponds to a different fraction of a wavelength at each frequency.

B. Polarity inversion only applies to the signal after the fader, so it does not reach the summing point where the cancellation occurs.

C. The two microphones have different polar patterns, and polarity inversion is only valid between microphones of the same pattern.

D. Inverting the boom moves the notches rather than removing them: the frequencies that previously cancelled now reinforce and the ones that reinforced now cancel, so the comb is shifted by half a tooth and the total amount of cancellation across the spectrum is unchanged.

53 · 9 min

Latency, and why a speaker starts stumbling

THE ONE THING TO KEEP

Buffer latency is buffer size divided by sample rate and is paid both ways, so a 512-sample buffer at 48kHz is over 20ms round trip — past the point where hearing your own voice late disrupts speech — and direct monitoring in hardware sidesteps the whole chain.

Why a speaker starts stumbling over their words

Put headphones on somebody, feed their own voice back to them through a computer, and if the delay is more than a few milliseconds they begin to hesitate, slow down and repeat syllables. They will usually blame themselves.

This is **delayed auditory feedback**, and it is a well-documented effect rather than a matter of preference. Hearing your own voice late disrupts the feedback loop speech production relies on. The disruption is strongest around 150 to 200ms — where it is severe enough that it has been used as an experimental technique — and it is measurable well below that. For anyone speaking or performing on a headphone feed, the useful threshold is roughly **10ms round trip**; above about 20ms most people notice something is wrong even if they cannot name it.

Where the milliseconds come from

Buffer size is the dominant term. Audio is processed in blocks, and the computer needs a block's worth of samples before it can do anything.

```
buffer latency = buffer size / sample rate
```

- 128 samples at 48kHz = 2.7ms
- 256 samples at 48kHz = 5.3ms

- 512 samples at 48kHz = 10.7ms
- 1024 samples at 48kHz = 21.3ms

That figure is incurred on the way in *and* on the way out, so round-trip is roughly double. Add converter latency — the anti-aliasing and reconstruction filters, typically 1 to 2ms combined — and any plugin that looks ahead.

Note the useful consequence: at 96kHz, the same 256-sample buffer is 2.7ms instead of 5.3. This is the genuine low-latency argument for higher sample rates from the previous lesson.

The trade you are making

A smaller buffer gives lower latency and leaves the CPU less time to finish each block. Miss the deadline and you get a **buffer underrun** — a click, a pop, or a dropout. There is no way around the trade; it is the same CPU either way.

The working practice everyone converges on:

- **Small buffer (128 or 256) while recording**, with as few plugins as possible.
- **Large buffer (1024 or more) while mixing**, where latency does not matter and you want the processing power.

Most recording software has a one-click switch between the two.

Direct monitoring, which removes the problem entirely

Most audio interfaces can send the input signal straight to the headphone output in hardware, without it passing through the computer. Latency is then microseconds — effectively zero.

This is the correct arrangement for recording a voice. The speaker hears themselves instantly through the hardware path; the computer records in parallel and its latency is irrelevant because nobody is listening to its output.

The limitation is that you hear the raw signal, without any processing. For voiceover that is fine and usually preferable. For a singer who wants reverb to

perform against, you need either an interface with built-in effects on the monitor path or a low enough buffer to monitor through the computer.

Bluetooth is not a monitoring option

Bluetooth audio latency is typically 100 to 300ms depending on the codec, and the protocol is not designed to be low. Even the low-latency codecs land in the 40 to 80ms range.

This makes Bluetooth headphones unusable for recording, for monitoring, and for editing to picture — lip sync will look wrong and you will "fix" it into being wrong. Wired headphones for anything you are making; Bluetooth for listening to things other people made.

Some playback software compensates by delaying the video to match, which works for playback and not for recording.

Latency inside the session

Plugins introduce their own delay. A linear-phase EQ, a look-ahead limiter or a convolution reverb may need dozens of milliseconds. Every modern host applies **plugin delay compensation**, delaying every other track to match the slowest, so the mix stays aligned.

Two things to know. It only works for plugins that report their latency honestly, and a few do not — the symptom is one track sounding slightly loose against the rest. And compensation cannot help live monitoring: if you are listening through a look-ahead limiter, that delay is real and unavoidable, which is another argument for direct monitoring while recording.

The monitoring path matters more than the numbers

Two practical points that are worth more than any buffer setting.

Closed-back headphones for recording. Open-back headphones leak, and the leak is picked up by the microphone. It will not be loud, but it produces a faint echo of the programme under the voice that is impossible to remove.

Someone must listen on headphones during the recording. Not watch the meters — listen. A meter shows level and says nothing about a cable crackle, a rustling sleeve, a phone buzzing on a desk, or a microphone that has drifted off axis. On a shoot this is a person's job for a reason; on a one-person setup, wearing the headphones is the substitute.

The free path

Audacity, **Ardour** and **Reaper** all expose buffer size directly. On Linux, **JACK** and **PipeWire** handle low-latency routing well and let you set the period size explicitly. On macOS, Core Audio is low-latency by default. On Windows, **ASIO4ALL** is free and is often the difference between a usable and an unusable interface, since the default Windows driver path adds substantial latency.

Measure what you actually have rather than trusting a display: play a click out, loop the output back into an input with a cable, record it, and measure the gap in samples. Most hosts have a built-in routine that does exactly this.

The limitation

Latency cannot be reduced to zero through a computer. Block processing is how digital audio works, and the converters have filters with real group delay. Direct monitoring sidesteps it rather than solving it.

And reported latency is often wrong. Interfaces report a figure to the host that may omit the converter delay, and a host's displayed round-trip number can be several milliseconds optimistic. If a musician says it feels late while the software claims 6ms, believe the musician — the loopback measurement is the only number that is not a claim.

BEFORE YOU MOVE ON

A voiceover artist monitoring through the computer keeps hesitating. The engineer halves the buffer from 512 to 256 samples and the session starts producing clicks. What is the better fix?

A. Raise the sample rate to 96kHz, which halves the latency at any given buffer size and gives the CPU the same time per block.
B. Switch to direct hardware monitoring, so the artist hears the input before it reaches the computer and the buffer can be set for stability instead.
C. Enable plugin delay compensation, which aligns the monitored signal with the recorded one and removes the perceived delay.
D. Move the monitoring chain to a lighter set of plugins and raise the buffer back to 512, accepting the 21ms round trip on the grounds that it is well below the 150 to 200ms range where delayed auditory feedback has its strongest disruptive effect on speech production.

54 · 10 min

Two devices, one event

THE ONE THING TO KEEP

Offset is a constant to be slid and drift is a rate error to be stretched, and the commonest reported drift is not a clock problem at all but a 1000/1001 frame-rate mismatch worth 3.6 seconds an hour — while timecode fixes the offset and does nothing about the drift.

Two recorders, one event

Recording sound into the camera is convenient and usually worse: camera preamps are mediocre, the camera is in the wrong place for a microphone, and the microphone has to be wherever the camera is. So most filmed work records sound separately — **double system** — and the two recordings have to be brought back together.

The joining has two distinct problems, and people conflate them. **Offset** is a constant difference: the recorder started three seconds before the camera. **Drift** is a difference that grows: the two devices are not running at exactly the same rate.

An offset is fixed once. Drift has to be fixed by changing the length of one of the recordings.

The clap, and why it still works

A sharp transient visible on the picture and clearly marked in the waveform gives you a single unambiguous alignment point. A clapperboard does it formally; two hands do it perfectly well.

Two details that make it work:

- **In frame and in focus.** You need to see the exact frame where the hands meet.
- **Close to the microphone, and mark the end too** on a long take. Two claps, one at each end, let you measure drift directly: if the head claps align and the tail claps are 12 frames apart after an eight-minute take, you have measured your drift.

Where drift comes from

Every digital device has a clock crystal, and no crystal is exact. A recorder nominally at 48kHz might run at 48,002Hz. That is 42 parts per million — about 0.15 seconds over an hour.

That much is usually tolerable for speech. The failures that matter are larger and have specific causes:

A frame-rate mismatch of 0.1 per cent. A camera at 29.97fps with audio timed for 30, or 23.976 against 24. The ratio is 1000/1001, which is 3.6 seconds per hour — grossly out of sync within minutes. This is by far the most common "my audio drifts" report, and it is not a clock problem at all; it is a project setting.

Variable frame rate recording. Phones and screen-recording software frequently record with a variable frame rate, storing frames as they arrive rather than at fixed intervals. An editor that assumes a constant rate will drift progressively. The fix is to conform to a constant rate on the way in:

```
ffmpeg -i phone.mp4 -vsync cfr -r 30 -c:v libx264 -crf 18 -c:a
aac fixed.mp4
```

Genuine clock drift between two independent devices, which is the small residue after the other two are excluded.

Timecode, and what it does and does not solve

Professional recorders and cameras can record **timecode**: a clock value embedded per frame or striped as an audio signal. If both devices are set to the same time-of-day timecode, the editor can align every clip automatically with no claps at all. On a multi-camera shoot with hours of material, this saves days.

The important distinction: **timecode gives you the offset; it does not stop drift.** Two devices sharing a timecode value still have independent clocks. Devices that stay locked either jam-sync periodically from a master, or are genlocked to a shared clock signal. A recorder jammed at the start of the day and left alone will drift by the afternoon, which is why the convention is to re-jam every few hours.

Timecode-generating boxes are relatively expensive and are the usual dividing line between a hobby setup and a professional one.

Free automatic sync

You do not need timecode for a small shoot. Waveform-matching is free and works well when the camera also recorded scratch audio from its own microphone — which it should, precisely for this.

The technique cross-correlates the camera's low-quality track with the recorder's good one, finds the offset with the highest correlation, and slides

the clips together.

- **DaVinci Resolve**'s free edition has audio waveform sync built in, including multicam.
- **Kdenlive** and **Shotcut** both offer clip alignment.
- **Blender**'s video sequencer has a waveform-based sync.
- On the command line, `audio-offset-finder` and similar open-source tools print the offset in seconds for scripting across a folder.

Two requirements: the camera must have recorded *some* audio, and the two must overlap in content. Sync fails when the camera microphone heard something entirely different, as with a distant subject and a radio microphone in a noisy room.

Fixing drift after the fact

Once you have measured it — head and tail claps, or the first and last obvious transient — correct the rate rather than cutting the audio up.

```
# stretch audio by 0.1% to match a 29.97 vs 30 mismatch
ffmpeg -i audio.wav -af "atempo=0.999" -ar 48000 fixed.wav

# or resample, which also shifts pitch very slightly
ffmpeg -i audio.wav -af "asetrate=48000*0.999,aresample=48000"
fixed.wav
```

`atempo` changes duration without pitch, and at 0.1 per cent the artefacts are undetectable. `asetrate` is a straight speed change with a pitch shift of about 1.7 cents, also undetectable at this magnitude and preferred by some because it is a pure rate change with no time-stretching algorithm involved.

Practices that save the day

- **Always record scratch audio on the camera**, even when you have a separate recorder. It is your sync reference and your backup.
- **Clap at the head and the tail** of anything over two minutes.

- **Announce the take out loud** — scene, take, and anything unusual. Audio slates cost nothing and are searchable by ear when file names have gone wrong.
- **Check sync at the end of the take, not the start.** Aligning the first clap proves nothing about minute nine.
- **Set the project frame rate first**, and conform anything that does not match on ingest.

The limitation

Nothing here recovers a recording that stopped. Separate devices fail separately: a card fills, a battery dies, somebody knocks record on a recorder that was already running and turns it off. The camera's scratch track is the only thing that saves the take, which is the real reason to insist on it.

And automatic sync is a correlation, not an understanding. On repetitive content — a metronome, a looped announcement, a rhythmic machine — it can lock onto the wrong cycle with total confidence and be exactly one bar out. Spot-check by ear on a consonant, at the end of the clip, before trusting a batch of automatic syncs.

BEFORE YOU MOVE ON

Audio synced perfectly at the head of an eight-minute take is about half a second late by the end. The recorder and camera were both set to 48kHz and the project is 23.976fps. What should you check first?

A. The recorder's clock crystal, which at a typical tolerance of a few dozen parts per million would produce exactly this magnitude of error over eight minutes.

B. The camera's audio sample rate, since a camera recording scratch audio at 44.1kHz while the project assumes 48kHz would produce a progressive offset as the timeline stretches the file.

C. Whether the audio was timed for 24fps and the picture is 23.976, a ratio of 1000/1001 worth about half a second in eight minutes.

D. Whether the camera recorded with a variable frame rate, which is the usual cause of progressive drift on consumer devices and which would need conforming to a constant rate before the clips can be aligned at all.

CASE STUDY · MODULE 6

A twelve-thousand-rupee grant and a tiled kitchen

A government higher secondary school in Sirsa, recording revision audio for students who cannot stream video at home.

Manpreet and Iqbal record fifteen-minute revision talks. They go out on memory cards to students without data and on WhatsApp to those who have it. The first twenty were recorded on a phone laid on the staff-room table, about a metre and a half away. Students said they sounded far away. One boy said he could not follow them at all on his father's phone speaker in the shop.

Then a district innovation grant arrived: twelve thousand rupees, to be spent on equipment, itemised in the register. A vendor quoted a USB condenser microphone and a desk stand. The headmaster liked the idea of a microphone. It is a thing, it can be shown to a visiting officer, and it is unambiguously equipment.

Manpreet measured before buying anything, which took about twenty minutes. She recorded ten seconds of room tone in the staff room and ten seconds of speech, and compared them with ffmpeg's `astats`: the voice sat twenty-eight decibels above the room. She moved the phone to twenty centimetres and measured again: forty. Then she recorded the same paragraph three ways — staff room at a metre and a half, staff room at twenty centimetres, and the store room at twenty centimetres with two spare curtains hung on the facing wall and a folded blanket under the phone. She played all three on the cheapest phone in the building.

The store-room version was not slightly better. It was a different recording.

That produced an awkward decision. The two changes that had done the work — getting close, and putting soft mass on the surfaces facing the microphone — cost nothing and could not be entered in a register. The purchase that could be entered would not change the direct-to-reverberant ratio at all, because that ratio is set by distance and by the room, and the microphone's price is not on the list.

Manpreet asked the office whether curtains and mineral-wool panels could be booked as acoustic material. The office said no: soft furnishing is not equipment.

So they split it. Three and a half thousand went on a wired lavalier and a small handheld recorder, so that a teacher could record in the store room without carrying a laptop. Twelve hundred went on a second memory card and a backup card reader. The remaining seven thousand bought the condenser the headmaster wanted — and it lives in the store room, used at twenty centimetres, behind curtains the science department donated because they were being replaced anyway.

Manpreet was clear with herself that this was not a pure outcome. Some of the grant went on the least effective item in the room, and she agreed to that knowingly, because a grant returned unspent is a grant the district does not offer again and because the headmaster's support was worth more over a year than two thousand rupees of equipment. What she refused to do was let the purchase stand in for the change: the microphone was bought, and the recording position moved anyway.

The second thing she insisted on was cheaper still. Every talk now begins with thirty seconds of the room, recorded at the same gain, in the same position, before anyone speaks. It sounds like nothing. It is what lets a sentence be cut out of the middle of a talk without the background stopping for a frame at the join, and it is impossible to go back for once the chairs have moved.

> **STOP HERE**
>
> Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

The store-room setup measured forty-seven decibels of signal to noise against twenty-eight in the staff room, and the difference was obvious to everyone in the room within one sentence on a phone speaker. Used in the same place at

the same distance, the condenser measured only about two decibels better than the lavalier on Manpreet's voice — but it was noticeably cleaner on Iqbal's, whose voice is soft: the lavalier's own self-noise is over twenty decibels A-weighted and audible under him, and the condenser's is under fifteen. The curtains cost nothing and did more than either. Manpreet's note to the district, since circulated to eleven schools, says the microphone was the smallest of the three changes and the only one the grant could pay for, and asks that future grants permit room materials. The teachers now record thirty seconds of room tone before every talk and lay it under their edits, which is why the joins stopped being audible.

Why did two donated curtains change the recording more than a seven-thousand-rupee microphone?

What a listener hears as amateur or professional is the ratio of direct sound to reverberant sound. Direct sound falls about six decibels for every doubling of distance; the reverberant field is roughly constant throughout a room. So the ratio is set by how close you are and how absorbent the surfaces facing you are. A microphone's price appears nowhere in that, and no microphone can separate a voice from its room — the capsule sums everything arriving at that point in space.

What did moving from a metre and a half to twenty centimetres buy, arithmetically?

That is close to three doublings of distance — roughly a metre and a half to seventy-five centimetres to thirty-eight to nineteen — so about eighteen decibels more direct sound, against a room noise level that did not move at all. The measured signal-to-noise went from twenty-eight decibels to forty, and it cost nothing but standing closer.

The condenser measured only about two decibels better, yet it was worth using for Iqbal. Why?

Self-noise. A condenser's published A-weighted self-noise under fifteen decibels is quiet; over twenty is audible on a soft voice, and the cheap lavalier was over twenty. On Manpreet's louder delivery the difference is buried; on Iqbal's it sits under every pause. The number that mattered was not sensitivity or price but where the microphone's own noise floor sat relative to the quietest thing he said.

MODULE 6 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 Why does a shotgun microphone often sound worse than a cardioid in a small reflective room?

A. Shotguns have no proximity effect, so the voice loses body

B. Its rear lobe picks up whatever is directly behind the operator

C. Its tube only works above about 1 kHz, so it colours reflections unevenly

D. Interference tubes require phantom power that small rooms rarely supply properly

2 Why is recording 12 dB too quietly an inconvenience while clipping is fatal?

A. Clipping replaces the peak with a flat top and harmonics that were not there

B. Clipping permanently raises the noise floor of the whole recording

C. A quiet recording can be normalised afterwards, which also removes any hiss with it

D. A limiter on the way in repairs a clipped peak after the fact

3 A 25 kHz tone is sampled at 44.1 kHz with no anti-alias filter. What appears in the file?

A. A 19.1 kHz tone, folded back around the Nyquist frequency

B. Nothing — frequencies above Nyquist are simply not recorded

C. A 22.05 kHz tone, clamped at the Nyquist frequency

D. A 25 kHz tone that only plays back at higher sample rates

4 Moving a microphone from 60 cm to 15 cm improves signal to noise by how much, and why?

A. 6 dB, because the room noise falls off under exactly the same inverse square law

B. 3 dB, because halving distance doubles power rather than amplitude

C. 12 dB, because the voice rises two doublings and the room noise does not

D. Nothing measurable, because only absorption changes a room's noise floor

5 Two microphones on one speaker differ by 34 cm in path length and the mix sounds hollow. Does the polarity button fix it?

A. Yes — a polarity flip inverts one signal and cancels the comb

B. No — polarity is frequency-independent and the fault is a time difference

C. Yes, provided the two microphones are of the same pattern

D. No — the fault is the three-to-one rule, which applies only to separate sources

6 Separate audio drifts 3.6 seconds ahead of picture over an hour. What is the most likely cause?

A. A crystal tolerance of about 40 parts per million in the recorder

B. The clapperboard was marked at the head of the take only

C. Timecode was jam-synced at the start and not re-jammed later

D. A 0.1 per cent frame-rate mismatch, such as 29.97 against 30

MODULE 7

Shaping and mixing sound

From a raw voice to a delivered mix: the three EQ moves that cover most speech, why compression makes sibilance worse, what pre-delay is really telling a listener, how a music bed and a voice stop competing, and the loudness and true-peak numbers a platform will measure you against.

> BY THE END OF THIS MODULE YOU CAN
>
> Take a raw voice recording to a delivered mix — filtering and cutting before boosting, ordering the chain so each stage is not re-creating work for the next, placing the voice in a space without losing intelligibility, sitting music under it by arrangement rather than by ducking alone, and hitting a named integrated loudness and true-peak ceiling with a gain change rather than a limiter — and predict what the mix will do on a phone speaker before hearing it there

55 · 10 min

Three EQ moves that cover most voices

THE ONE THING TO KEEP

High-pass below the speaker's fundamental, cut two to four decibels of mud between 200 and 400Hz, add a broad presence lift around 2 to 6kHz — and cut narrow while boosting wide, because problems are resonances and pleasantness is a shape.

Cut first, boost rarely, and know which frequencies you are touching

Most voice EQ is three moves. They are the same three moves nearly every time, and the reason they work is that they address three specific and predictable problems rather than a taste.

1. High-pass filter. Remove what is below the voice.

Adult speech fundamentals sit roughly between 85 and 180Hz for typical male voices and 165 to 255Hz for typical female voices. Below the fundamental there is nothing but rumble: traffic, air handling, footsteps, desk knocks, handling noise, and the low-frequency half of plosives.

Set a high-pass at **80Hz for a low voice, 100 to 120Hz for a higher one**, with a slope of 12 or 18dB per octave. Sweep it upward while listening until the voice starts to thin, then back off. This is the single most useful filter in speech work and it is doing nothing to the voice at all — it is removing energy that was never part of it and that costs you headroom in every later stage.

2. Cut the mud. There is almost always a build-up between 200 and 400Hz, from proximity effect, from a desk or a boundary reflection, or from the room. A **2 to 4dB cut with a moderately wide Q** is usually right. Too much and the voice becomes thin and telephone-like.

The three moves that cover most voices

Cut narrow and boost wide. A problem is usually a resonance, which is narrow; what sounds pleasant is a shape across a region, so a narrow boost sounds like the resonance it has just created. None of this restores energy a reflection cancelled — a notch is not attenuation, and boosting into one amplifies the noise in a band where the signal is gone.

3. A small presence lift. Consonant clarity lives between about 2 and 6kHz. A gentle **1 to 3dB broad boost** in that region improves intelligibility noticeably. Above 10kHz, a shelf adds "air" — worth a decibel or two on a good recording and worth nothing on a noisy one, because up there you are mostly lifting hiss.

That is the whole default. Most voices need nothing else.

Finding a resonance rather than guessing

When something specific is wrong — a honk, a boxiness, a ring — there is a reliable method.

Take a narrow band, boost it by 10 to 12dB, and sweep slowly across the spectrum while the voice plays. At the offending frequency, the problem becomes unbearable. Note the frequency, then **cut** there instead, by 3 to 6dB, with a moderate Q.

This works because a resonance is narrow and amplifying it makes it unmistakable. It is also easy to overuse: the sweep makes everything sound bad at high boost, and there is always *a* worst frequency. If nothing jumps out as obviously wrong, there is nothing to fix.

Cut narrow, boost wide

A useful asymmetry. Problems tend to be narrow — a resonance at one frequency, a ring from one room dimension — so cuts can be surgical. What sounds pleasant tends to be broad, because tonal character is a shape across a region rather than a spike. A narrow boost sounds like a resonance, because it is one.

Practical: cuts at Q of 2 to 6; boosts at Q of 0.7 to 1.4.

Minimum phase, linear phase, and what each costs

An ordinary EQ is **minimum phase**: changing the amplitude at a frequency necessarily shifts the phase around it. This is not a defect but a mathematical consequence of a causal filter, and it is what analogue EQ does. On a single voice it is inaudible and unimportant.

A **linear phase** EQ keeps phase constant by processing with a symmetric filter, which requires looking ahead — so it introduces latency, and it produces **pre-ringing**: a faint smear *before* a transient, which has no physical analogue and can sound unnatural on percussive material.

Which to use: minimum phase for almost everything, and particularly for anything you are monitoring live. Linear phase is worth it when you are EQing two microphones on the same source that will be summed, because a phase shift on one and not the other changes how they combine. On a single voice track, reach for the ordinary one.

The point everybody skips

Fix it with placement before you fix it with EQ. A boxy recording caused by a desk reflection has a notch in it, and an EQ boost aimed at a notch is amplifying the noise in a band where the signal was cancelled. Moving the microphone 20cm or putting a cloth on the desk removes the cause.

EQ is very good at removing excess energy and very bad at restoring energy that cancelled. Cuts are cheap; boosts have a cost.

A worked starting point

For a close-miked voice in a treated-enough room:

```
high-pass       90 Hz,  18 dB/oct
bell           280 Hz,  -3 dB,  Q 1.2
bell          3.5 kHz,  +2 dB,  Q 1.0
high shelf     11 kHz,  +1.5 dB
```

Then listen and change one thing. The mistake is to build a six-band chain because the plugin has six bands.

The free path

Audacity has a graphic and a parametric EQ and a spectrogram view. **Ardour** ships with a capable parametric EQ and analyser. The **Calf** and **LSP** plugin suites are free, cross-platform, and professional in scope. **ReaEQ**, from Reaper, is free to use within Reaper's evaluation. **ffmpeg** applies EQ from the command line for batch work:

```
ffmpeg -i in.wav -af
"highpass=f=90,equalizer=f=280:t=q:w=1.2:g=-3,equalizer=f=3500:
t=q:w=1:g=2" out.wav
```

A spectrum analyser costs nothing and is worth more than any EQ: **Voxengo SPAN** is free, and Ardour and Audacity both include one.

The limitation

EQ changes the balance of what was recorded and cannot separate two things occupying the same band. If the fridge hum and the voice's fundamental are both at 120Hz, a cut at 120Hz removes both. Every EQ decision on a noisy recording is a trade rather than a repair.

It also cannot fix a room. A comb filter from a reflection has notches every few hundred hertz, and EQ can only apply a smooth curve to a jagged problem. You can reduce the most prominent tooth and you cannot fill in the gaps, because the cancelled energy is not attenuated — it is gone.

BEFORE YOU MOVE ON

A recording sounds hollow because of a reflection from a desk 40cm below the microphone. An engineer tries to fix it with EQ boosts at the affected frequencies and the result is noisier without sounding fuller. Why?

A. The boosts are too narrow, and a narrow boost always reads as a resonance rather than as added body; wider boosts at the same frequencies would restore the missing energy.

B. The comb filter's notches are cancellations, so at those frequencies the signal is absent and boosting raises only whatever noise remains there.

C. A minimum-phase EQ shifts phase around each boost, and those shifts are reinforcing the original comb filter rather than opposing it.

D. The reflection arrives 2.3ms after the direct sound, which is within the window where the ear fuses the two into a single event, so the notches are perceptual rather than present in the signal and an EQ cannot act on them at all.

56 · 11 min

Compression does not make anything louder

THE ONE THING TO KEEP

A compressor only turns loud parts down; the make-up gain afterwards is what raises everything, including whatever noise sits in the gaps.

Compression does not make anything louder

A compressor turns down anything above a threshold. That is the entirety of what it does. It has no mechanism for making sound louder.

What makes it louder is the make-up gain you add afterwards. Once the peaks have been pulled down, there is headroom, so you can raise the whole signal — and the quiet parts come up with it. Hence the only description of compres-

sion worth memorising: **turning the loud bits down so the quiet bits can come up.**

Understanding that changes how you set it, and it explains the failure mode at the end of this lesson.

Do it in this order

Clean the performance, then fix the tone, then control the dynamics, then set the loudness. In that order.

Out of order, the compressor reacts to noise you are about to remove, and your EQ shapes a signal you are about to change. People who cannot get a voice to sit right are usually sequencing wrong rather than doing any one step wrong.

Editing the performance

Cut the false starts, the long pauses and the ums. Do not cut every breath. A voice with no breaths sounds inhuman, and breaths mark where sentences begin and end — remove them all and listeners lose the structure of what you said. Reduce a distracting breath by 6–10 dB instead of deleting it.

Two mechanical points that account for most audible edits:

- Place a cut at a zero crossing, or put a 5–10 ms crossfade over it. Cutting through the middle of a waveform leaves a step, and a step is a click. Most editors handle this; Audacity does not always.
- Cut immediately after a hard consonant — a t, k or p — where the seam hides. Cutting in the middle of a vowel is audible even when the timing is right.

Then lay your room tone under the whole track. Silence between edited phrases sounds like the audio has dropped out. Continuous room tone is what makes heavily edited speech sound like one take.

EQ in three moves

1. **High-pass filter.** Around 80 Hz for a deep voice, 100–120 Hz for a higher one. Below that there is rumble, air conditioning, traffic and desk

thumps, and no speech at all. Sweep it upward until the voice starts to thin, then come back a little.

2. **Cut, do not boost, between 200 and 500 Hz** if the voice sounds boxy, boomy or muddy. Two to four decibels with a fairly wide bandwidth. This is where a small untreated room lives.
3. **A gentle lift of 2–3 dB between 3 and 6 kHz** for clarity. If that makes the s sounds harsh, use a de-esser around 5–8 kHz rather than a static cut, which dulls the whole voice to fix a few moments.

The general rule: cuts are forgiving, boosts are not. If you find yourself boosting more than about 4 dB anywhere, the answer is at the recording end.

Setting a compressor

A workable starting point for speech:

- Ratio 3:1.
- Threshold set so the loudest phrases show 3–6 dB of gain reduction on the meter — and the quiet phrases show none.
- Attack around 10 ms. Fast enough to catch the peaks, slow enough to let the consonant transients through. A very fast attack dulls speech and is a common cause of a voice that sounds oddly soft.
- Release 100–200 ms, or auto.
- Make-up gain until the compressed version matches the uncompressed version in perceived level, so you are judging the sound and not just the volume.

The failure mode. If the background hiss rises and falls between sentences, you are over-compressing. During speech the compressor is pulling the level down; in the gap there is nothing above the threshold, so nothing is being pulled down, and the full make-up gain lands on the room noise. That swelling and receding is the sound of too much gain reduction. Back it off, or gate gently, or fix the noise floor at the source.

De-noising, honestly

It works by learning a profile of a steady sound and subtracting it in the frequency domain. So it handles hum, fans and hiss well, and anything that varies badly.

Push it hard and you get the characteristic artefacts: an underwater quality, and small warbling tones sometimes called birdies, which appear where the subtraction has removed most but not all of a frequency band. Six to ten decibels of reduction is usually inaudible. Twenty is usually obvious.

Take the least you can live with. A slightly hissy voice is far easier to listen to for ten minutes than a clean one that warbles.

Loudness, and why louder is pointless

Streaming platforms normalise. Push your file louder than roughly −14 LUFS and YouTube or Spotify will simply turn it back down, and all you have achieved is a squashed, airless dynamic range that arrives at the listener at the same volume as everyone else's.

Aim for around −14 LUFS integrated for video platforms, around −16 LUFS for podcasts, with true peak no higher than −1 dBTP. Free ways to measure: Resolve's Fairlight page has a loudness meter, ffmpeg's `loudnorm` filter measures and applies, and Youlean's free loudness meter plugin works in most hosts.

The free toolkit, and ten seconds today

- **Audacity** — free and open source, does every step above, including a usable noise reduction.
- **DaVinci Resolve, Fairlight page** — free, a proper mixer with compression, EQ, metering and a ducker.
- **Reaper** — an evaluation that never expires and a personal licence around $60, as of 2025.
- **Ocenaudio**, **Ardour**, **Cakewalk** on Windows — all free or near enough. On a phone, **CapCut**'s audio enhancement is fine for speech.

Ten seconds today

Take one minute of your voice, apply a high-pass filter at 100 Hz, and A/B it. That one control is usually the largest single improvement available to you.

BEFORE YOU MOVE ON

After compressing a voiceover fairly hard, an editor notices the background hiss swelling up between sentences and receding when the person speaks. What is happening?

A. The noise reduction was applied after the compressor and is mistracking the changing signal.

B. The release time is too long, so the compressor is holding the gain down into the following phrase.

C. The high-pass filter is set too low and is letting low-frequency rumble through in the quiet passages.

D. During speech the compressor pulls the level down, but in the gaps nothing exceeds the threshold, so the full make-up gain lands on the room noise. There is too much gain reduction.

57 · 10 min

Wind, hiss, and why both are placement problems

THE ONE THING TO KEEP

A plosive is a jet of air rather than a loud sound, so angling the microphone 15 to 30 degrees off axis prevents what no filter fully repairs — and a de-esser is a compressor with a filtered sidechain, which belongs after compression because compression is what raised the sibilance.

Two problems that sound like microphone faults and are not

Sibilance — the harsh hiss on s, sh, z and t — and **plosives** — the thud on p and b — are the two artefacts most often blamed on equipment. Both are physical, both are predictable, and both are much easier to prevent than to repair.

A plosive is wind, not sound

Say "peter piper" with your hand in front of your mouth. You feel a puff of air. That blast hits the diaphragm and displaces it far beyond what the sound itself would, producing a large, very low-frequency excursion — often below 80Hz — that can overload the preamp even when the voice is at a moderate level.

Three prevention measures, in order of effectiveness:

1. **Speak past the microphone, not into it.** Angle it 15 to 30 degrees off the mouth's axis, or place it slightly above and aimed down at the mouth. The air jet is directional and the sound is not, so you lose almost nothing and avoid the blast entirely. This is free and it is what professionals do.
2. **A pop filter, 5 to 10cm from the capsule.** The distance matters: hard against the microphone it does very little, because the jet re-forms. Its job is to break up and slow the airflow across a gap.

3. **A high-pass filter.** The lesson before covers this. It reduces the aftermath rather than preventing it.

Repairing one afterwards: do not apply a global filter for the sake of one word. Automate a steep high-pass — 150 to 200Hz, 24dB per octave — over just the 100 to 200 milliseconds of the plosive, or use a clip-gain dip on that syllable. Both are invisible and neither touches the rest of the track.

Sibilance, and what a de-esser really is

Sibilant consonants are broadband noise concentrated between roughly 5 and 9kHz — typically 5 to 7kHz for lower voices, 7 to 9kHz for higher ones. Several things make them worse:

- **Proximity.** Close miking raises them relative to the body of the voice.
- **Compression.** A compressor turns the loud parts down, and sibilants are not usually the loudest parts, so everything else comes down around them and they stand proud. Compression makes existing sibilance worse, reliably.
- **A presence boost.** The 2 to 6kHz lift from the EQ lesson lands next door.
- **The speaker.** Some people are simply sibilant, and dental work changes it.

A **de-esser is a compressor with a filtered sidechain**. The detector hears only the sibilant band, so it triggers only on sibilants; the gain reduction is then applied either to the whole signal (wideband) or only to that band (split-band).

- **Split-band** is transparent for moderate work and is the sensible default: it turns down 6kHz and leaves the voice alone.
- **Wideband** ducks the entire voice on each "s", which sounds more natural on heavy reduction because it preserves the relationship between the sibilant and the rest — but it makes the voice dip audibly if pushed.

Setting one without ruining the voice

1. Put the de-esser **after** compression, not before. Compression is what exaggerates sibilance, so de-essing before it hands the compressor a signal it then re-unbalances.
2. Use the listen or solo-sidechain function to hear what the detector hears. Tune the frequency until you hear the s sounds clearly and the vowels barely.
3. Set the threshold so gain reduction is **3 to 6dB on the worst syllables and zero the rest of the time**. If the meter is moving constantly, the threshold is too low and you are dulling the whole voice.
4. Listen for the lisp. Over-de-essing turns "s" into "th", and it is instantly recognisable once you have heard it.

Two de-essers doing 4dB each, at slightly different frequencies, frequently sound better than one doing 8dB — the same principle as gentle serial compression.

Automation beats processing

For a handful of harsh syllables in a five-minute read, the cleanest fix is not a plugin. Find the syllable, reduce its clip gain by 3 to 5dB, move on. It takes a minute, it is perfectly transparent, and it does nothing at all to the other 99 per cent of the track.

Dialogue editors do a great deal of this by hand, and the reason is simple: a static process applies everywhere, and problems are local.

Breaths, and a word about removing them

Breath noise is the third item people want gone. Resist deleting them entirely — speech without breaths sounds unnervingly inhuman, and listeners cannot say why. Reduce the loud ones by 6 to 10dB and leave them present.

If you do cut one out completely, replace it with room tone rather than silence, for the reason covered in the capture module.

The free path

- **Audacity** has a De-Clicker and De-Esser in recent versions, and clip-level gain envelopes for manual work.
- **The LSP plugin suite** includes a proper split-band de-esser, free and cross-platform.
- **Calf Deesser** is free, LADSPA/LV2.
- **ffmpeg** can do a crude sidechain-filtered compressor, and `adeclip` / `adeclick` repair damage.
- **DaVinci Resolve**'s free Fairlight page has de-essing built in.
- iZotope RX's de-click and de-plosive tools are the paid standard; the free options handle a spoken-word track comfortably.

The limitation

A de-esser cannot distinguish a sibilant from any other loud high-frequency content. Cymbals, key jingling, paper rustle and a bright laugh all trigger it, and on a track with music or effects mixed in it will duck them too. This is why de-essing belongs on the dialogue track before the mix, not on the master.

And there is a floor set by the recording. Sibilance that has clipped — very common, because the s is short and can peak far above the vowels — is distortion, not excess level. Turning it down turns down a distorted sound. The only cure is at the microphone, which is why the placement advice at the top of this lesson matters more than any plugin below it.

BEFORE YOU MOVE ON

Why does adding compression to a voice track usually make sibilance more prominent, when a compressor only reduces gain?

A. The compressor's attack time is too slow to catch the short sibilant transient, so the s passes through at full level while the attack envelope is still developing.

B. Compression adds harmonic distortion in the high frequencies through its gain-reduction circuit, and sibilance is the band where those harmonics land.

C. Sibilants are rarely the loudest part of a voice, so the compressor turns everything else down around them and the make-up gain raises the whole thing back.

D. The compressor's detector is weighted toward low frequencies by its own sidechain design, so it responds strongly to vowels and barely at all to sibilants, which means the sibilant band is the only part of the spectrum passing through the processor without any gain reduction applied to it whatsoever.

58 · 10 min

Reverb is information about where

THE ONE THING TO KEEP

Pre-delay corresponds to the distance of the first reflecting surface at about 34cm per millisecond, and increasing it separates the voice from its own tail — which is why a muddy dialogue reverb is fixed with pre-delay before it is fixed with level.

Reverb is information about where

A completely dry voice sounds like it exists nowhere. That is occasionally what you want — a narrator outside the story, an interior monologue — and almost always wrong for a voice that is supposed to be in a place. Reverb is not deco-

ration; it is the cue that tells a listener the size of the room and the distance to the speaker.

The physical event it models is straightforward. You speak; the direct sound arrives first. A handful of distinct reflections arrive next, from the nearest surfaces, at times set by their distances. Then the reflections multiply until they become a dense, decaying wash.

Every reverb control maps to part of that.

Pre-delay: the control that matters most

Pre-delay is the gap between the direct sound and the start of the reverb. It corresponds to the extra distance the first reflection travelled, and at 343 m/s, 1 millisecond is about 34cm.

A 20ms pre-delay implies a reflecting surface roughly 3.4 metres away — the sound went 3.4m out and 3.4m back, arriving about 20ms after the direct. So pre-delay tells the listener how big the room is and how far from the walls the speaker is standing.

It also has a practical use that has nothing to do with realism: a pre-delay of 20 to 40ms separates the voice from its own tail in time, so the words stay intelligible even with a generous reverb. Zero pre-delay smears the reverb into the consonants and the voice goes mushy. **If a reverb is making the dialogue hard to follow, increase the pre-delay before reducing the level.**

Decay, early reflections, and the high-cut

Decay time (RT60) is how long the tail takes to fall by 60dB. For a voice in a believable interior, 0.4 to 0.8 seconds. A cathedral is 3 to 8 seconds and will swallow speech.

Early reflections are the first discrete echoes, and they carry room size more strongly than the tail does. A reverb with strong, well-spaced early reflections and a short tail reads as a small real room; a reverb that is all tail reads as an effect.

High-frequency damping is not optional. Air absorbs high frequencies over distance, and soft furnishings absorb them at every bounce, so a real room's reverb tail is duller than its direct sound. A reverb return with full treble sounds artificial and metallic. **Roll the reverb return off above 5 to 8kHz**, and high-pass it below 200 to 300Hz too, which keeps the low end of the mix clear.

Send, not insert

Put reverb on a **send**: the dry voice goes to the output, a copy goes to a reverb bus, and you mix the wet return underneath. Three reasons.

- The dry signal stays untouched, so intelligibility is preserved and the balance is adjustable at any time.
- Several sources can share one reverb, which makes them sound as if they are in the same room — which is the point.
- You can process the return independently: EQ it, compress it, duck it under the dry voice.

An insert reverb with a wet/dry mix does roughly the same thing for one track and gives up the shared-space benefit.

Matching a room you already have

The commonest real task is not "add atmosphere" but "make this studio pickup match the on-location dialogue around it". The line recorded in a booth has to sit inside a scene recorded in a hallway.

The approach that works:

1. Find a moment of the location recording with room tone and a reflection you can hear.
2. Use a **convolution reverb** loaded with an impulse response of a similar space. You can capture one yourself: record a balloon pop or a starter pistol in the room, and the recording *is* the impulse response. Every convolution plugin will load a plain WAV.

3. Match the pre-delay and decay by ear against the location line, alternating between them.
4. Match the tone with EQ before reaching for more reverb — a mismatch is usually brightness, not space.

Capturing an impulse response with a balloon costs nothing and is far more accurate than picking a preset called "Small Room".

Amounts

For dialogue in a visible room, the reverb should be barely noticeable when you solo the voice and clearly missed when you mute it. That test — mute the return and see whether the voice suddenly floats — is the reliable one. If you can point at the reverb, it is too loud for dialogue.

Music and creative work have no such constraint. Dialogue does, because intelligibility is the job.

The free path

- **Convolution:** the **IR.lv2** plugin, **x42 zeroconvo**, and Audacity's built-in convolution all load WAV impulse responses free. Large libraries of impulse responses are available under permissive licences, including from the OpenAIR project.
- **Algorithmic: Dragonfly Reverb** is open source, cross-platform, and genuinely good. The **Calf** and **LSP** suites include reverbs. **Ardour** ships with a-Reverb.
- **DaVinci Resolve**'s free Fairlight page includes a full reverb set.
- **ffmpeg** has `afir` for convolution and `aecho` for crude echoes.

The limitation

Reverb adds a space; it cannot remove one. A recording made in a bad room already contains that room, and adding a nice reverb on top gives you two rooms at once, which sounds worse than either. Dereverberation tools exist and have improved sharply, and they work by estimating and subtracting the

room response — with the same class of artefacts as noise reduction, plus a tendency to hollow out the voice.

The ordering is therefore the same as everywhere else in this module: get it right at the capture stage, because the processing stage is always a negotiation with a loss.

BEFORE YOU MOVE ON

A dialogue line with a 0.6 second reverb is hard to follow. What does increasing the pre-delay from 0ms to 30ms do, and why?

A. It shortens the perceived decay time, because the tail now begins later and therefore ends sooner relative to the word.

B. It reduces the reverb's level at the start of each word, because the reverb send is muted during the pre-delay period.

C. It moves the reverb's onset clear of the consonants, so the direct sound is heard unobscured and the tail arrives after the word is identified.

D. It implies a reflecting surface about five metres away rather than immediately adjacent, and the larger implied room is perceived as more open, which is what makes the speech easier to follow in a way that no change in timing alone could achieve.

59 · 10 min

Music under speech

THE ONE THING TO KEEP

A ducker can only react while a human automating the level can anticipate, so manual riding beats sidechain compression on edited material — and cutting a two to four decibel hole around 1.5 to 3kHz in the music costs less than the level drop it replaces.

Music under speech, without either one losing

The problem: a voice and a music bed occupying the same output. Set the music where it sounds good and the words become hard to follow; set it where the words are perfectly clear and the music is so quiet it may as well not be there.

The instinct is to reach for a ducker. There are three better moves to make first.

Move one: choose music that leaves room

Speech intelligibility lives between roughly 1 and 4kHz. A music track with a busy vocal, a bright lead synth, or dense percussion in that region is competing for the same space, and no amount of level adjustment fixes competition — it only decides who loses.

Music that sits under speech well has: no vocal, sparse midrange, and most of its energy either below 500Hz or above 6kHz. Library music is often tagged "underscore" for exactly this. Choosing the right piece is worth more than every processing decision that follows, and it is free.

Move two: cut a hole rather than turning it down

Put an EQ on the music bus with a **2 to 4dB wide cut centred somewhere between 1.5 and 3kHz**. The music loses very little apparent presence — that region is not where its character lives — and the voice gains a clear channel.

This is far less noticeable than a 6dB level drop, because you have taken away only the part that was in the way.

Move three: automate before you compress

Manual level automation on the music track — riding it down under speech and up in the gaps — is what film and broadcast mixers actually do, and it beats any automatic process for one reason: **it anticipates**.

A human knows the line is about to start and begins the move 200ms early, over half a second, smoothly. A ducker can only react after the fact, and it reacts to every breath, every lip smack, every stray noise, producing a bed that surges and retreats constantly. That pumping is the signature sound of an automated podcast intro.

When a ducker is the right tool

For live work, for long-form conversation, and for anything with too many cues to automate by hand, sidechain ducking is the pragmatic answer. Settings that behave:

- **Threshold** so it triggers on speech and not on breaths.
- **Ratio** 2:1 to 4:1. It is a level shift, not a wall.
- **Attack** 10 to 30ms — fast enough to be out of the way by the first syllable, slow enough not to clip the music's transients.
- **Release** 300 to 800ms, which is the setting that matters most. Too fast and the music jumps back between words; a longer release keeps it down through a sentence and lets it return in a real pause.
- **Depth** limited to 6 to 9dB. Many dedicated duckers have a maximum-reduction control; use it. A ducker allowed 20dB of reduction makes the mu-

sic vanish.

- **Filter the sidechain** with a high-pass around 200Hz so low-frequency thumps do not trigger it.

How far down should the bed go

A working target: the music sits **15 to 20dB below the dialogue's typical level** while speech is happening. In loudness terms, for a programme delivered at −16 LUFS, a bed under speech commonly measures around −30 to −34 LUFS short-term, rising to −18 or so in the gaps.

Those figures are a starting point. The real test is not a meter: play it on a phone speaker, at a low volume, while doing something else. A bed that is comfortable on monitors at conversational volume is frequently too loud on a phone in a kitchen, because the small speaker has no low end and the bed's midrange is all that survives.

Sound effects are the same problem

Everything above applies to effects under dialogue, with one addition: an effect that is meant to be noticed should have a **gap around it**. A door slam landing between two lines reads as loud and clear; the same slam under a line is mush, however loud you make it.

Editing the dialogue to leave the space is more effective than raising the effect, and it is how it is done on any production where the dialogue and effects are mixed by the same person.

Three checks before delivery

1. **Mute the music.** Does the dialogue sound complete? If it only works with the bed, the dialogue edit has problems the music is hiding.
2. **Mute the dialogue.** Does the bed make sense on its own, or is it a series of unexplained swells? Pumping is obvious here and hidden under speech.
3. **Listen at low volume on a small speaker.** Quiet listening is where the balance fails first, because a small speaker and a low level both remove the parts of the music that were not competing.

The free path

- **Ardour**, **Audacity** and **Reaper** all do clip-gain and automation lanes for manual riding — the technique this lesson recommends most, and it needs no plugin.
- **Calf Sidechain Compressor** and the **LSP** sidechain compressor are free and have the controls listed above, including sidechain filtering.
- **ffmpeg** has `sidechaincompress` for batch work.
- **DaVinci Resolve**'s free Fairlight page has full sidechain routing.
- Free music with clear licences: the Free Music Archive, ccMixter, and material released under Creative Commons — with the licensing caveats covered in the final module, because "free to download" and "cleared for your use" are different claims.

The limitation

Ducking manages a conflict; it does not resolve one. If the music and the voice genuinely want the same frequencies and the same moments, something has to be cut, and the honest answer is usually the arrangement — a shorter music cue, a break under the key line, a different track.

There is also a taste question nobody can settle for you. Broadcast conventions in some countries favour beds so low they are almost subliminal; in others they sit much higher. Complaints about background music drowning speech are a recurring theme in broadcaster correspondence, and they come disproportionately from older listeners and those with hearing loss, whose ability to separate speech from a competing sound is reduced. If your audience includes them, mix the bed lower than sounds right to you.

BEFORE YOU MOVE ON

A podcast's music bed surges up and down constantly, retreating on every breath and lip smack. The ducker's threshold and ratio look reasonable. What is the most likely cause, and the most likely fix?

A. The ratio is too high, so each trigger produces a large reduction; lowering it to 2:1 will make the movements proportionate to the speech level.

B. The release is too short and the sidechain is unfiltered, so it recovers between words and triggers on low-frequency mouth noise; lengthen the release and high-pass the sidechain.

C. The attack is too fast, so the ducker is catching transients that are too brief to be speech; slowing the attack to 100ms will let breaths pass without triggering.

D. The bed is too loud overall, so every duck has a long way to travel and each movement is therefore more audible; reducing the music's static level before the ducker means each reduction covers a smaller range and the surging becomes imperceptible.

60 · 10 min

Three boring sounds beat any filter

THE ONE THING TO KEEP

Room tone plus one or two ordinary foley layers raises perceived quality more than any effect, and a Content ID claim is a match, not a verdict.

Three boring sounds beat any filter

Take a shot of someone putting a cup down on a table, with only the camera's own audio. Add three things:

1. **Room tone** under the whole scene, so the space has a floor and the cuts do not drop into digital silence.

2. **The cup contacting the table**, placed one or two frames *before* the visual contact. We accept sound slightly early far more readily than sound slightly late, so early reads as perfectly synchronised.
3. **A quiet cloth layer** for the arm moving.

None of that is glamorous. It does more for how expensive the shot feels than any filter, grade or transition you could apply. Perceived production value comes largely from things having a sound at all — silence where a sound should be is what registers as amateur, even to viewers who could not name what was missing.

The same holds for motion graphics. A transition with no sound reads as a glitch. One short whoosh with a low thump on the landing frame makes a logo animation feel deliberate, and it is the thump doing the work, not the whoosh.

The honest limit: layered sound design on a video that will be watched muted in a feed is wasted labour. Know where the thing plays before you spend an evening on foley.

Cutting music without it sounding cut

Music has structure, and the joins are where you can cut.

- Cut on a downbeat — the first beat of a bar — not wherever the timeline happens to sit.
- Cut at a section boundary. A drum fill leading into a new part is an invitation.
- To shorten a track, remove a **whole number of bars**, usually four or eight, and crossfade across the transient. Removing three and a half bars is what makes edited music sound wrong even when nobody can say why.
- To end a track, do not fade out over three seconds beneath your final line. Find the last hit and let it ring.

Choose music for its energy curve against your video's, not because you like the song. A track that builds at 0:40 is useless under a piece whose point lands at 0:12.

Ducking

Music under speech should drop 6–12 dB while someone talks and come back after.

Automatic duckers exist — Resolve's Fairlight has one, Premiere's Essential Sound panel has Auto-Duck, and Audacity has an Auto Duck effect. Doing it by hand with volume keyframes usually sounds better, because you can start the dip slightly before the first word and release slowly afterwards.

Set the release long, around 400–800 ms. A short release makes the music pump up between every sentence, which is more distracting than music that was simply too loud.

And most amateur mixes have the music too loud, for a specific reason: the editor has heard the voiceover fifty times and their brain fills in the words. Check on a phone speaker in a room with some noise in it. That is where it will actually be heard.

Licensing, told straight

"Royalty-free" means you do not pay per play. It does not mean free, and it certainly does not mean uncopyrighted. Somebody owns it.

Three situations people conflate:

- **Licensed music.** You bought a licence, from a subscription library or as a one-off. Read the term. Some libraries let you keep using videos published while you were subscribed; others do not, and your back catalogue becomes unlicensed the month you stop paying. Check that clause before you build a channel on a library.
- **Creative Commons music.** Free, and conditional. CC-BY requires attribution in the form the licence specifies. Skip the attribution and you are infringing, licence or not. Some CC licences forbid commercial use or derivative works, which includes cutting the track.
- **"No copyright music" found on YouTube.** Usually a lie, frequently reuploaded from a library, and the safest assumption is that it will be claimed.

Content ID is not copyright law. It is YouTube's automated matching system, comparing your audio against registered recordings. A claim can and does appear on videos with a valid licence, because the rights-holder registered the track and the matcher does not know about your receipt. You dispute it with the licence.

The practical habits: keep every licence PDF and invoice in the project folder alongside the video file, and check whether your library requires you to whitelist your channel ID *before* uploading. Skipping the whitelist step is the most common cause of a claim on properly licensed music, and it is a two-minute job you can only do in advance.

Sources that are genuinely safe and free: the **YouTube Audio Library** (per-track attribution flags, cleared for YouTube), **Pixabay** music, **Free Music Archive** and **ccMixter** (read the per-track licence), **Incompetech** (CC-BY — attribute it properly). For effects, **Freesound**, again with per-sound licences that vary.

Building a small sound-effects habit

You do not need a library subscription. Record your own: a door, a zip, a kettle, footsteps on three different surfaces, coins, paper, a chair. Ten minutes with a phone gives you a folder of things nobody else has, and cloth and footstep layers are the two you will reach for constantly.

Name the files properly the first time. An unsorted folder of `recording_047.m4a` is a folder you will never open again.

Today

Take a ten-second clip you have already edited. Lay room tone under all of it. Add exactly one foley hit at the most obvious physical action. Listen with and without. That comparison is the argument.

BEFORE YOU MOVE ON

An editor holds a valid subscription licence for a music track, uploads the finished video, and receives a Content ID claim anyway. What is the most accurate reading?

A. Subscription licences do not cover monetised uploads, so the licence was insufficient for this use.
B. The track was mislabelled as royalty-free when it is in fact copyrighted, which is what triggered the match.
C. Content ID is an automated match against a registered recording, not a legal ruling. It can land on a licensed use, is disputed with the licence, and on many libraries is prevented by whitelisting the channel first.
D. The track was shortened and crossfaded during editing, and the altered version failed verification and was flagged.

61 · 11 min

The number every platform measures

THE ONE THING TO KEEP

LUFS is a K-weighted, gated loudness measure, so platforms normalise playback and a master squashed 6dB louder than the target simply gets turned down 6dB — and the true-peak ceiling of −1 dBTP exists because the reconstructed waveform can exceed any stored sample.

The number every platform now measures

For twenty years, mastering meant making things as loud as possible, because on a playlist the louder track drew more attention. That competition — the loudness war — ended for practical purposes when streaming services began normalising playback. They now measure every track and turn it down to a common target.

That changes the arithmetic completely. If you squash a mix to be 6dB louder than the target, the platform turns it down 6dB, and all you have achieved is delivering a flatter, more fatiguing version at the same perceived volume as everybody else.

What LUFS measures

LUFS — Loudness Units relative to Full Scale — is a measurement designed to correlate with perceived loudness, standardised as ITU-R BS.1770 and adopted as EBU R128 in Europe and ATSC A/85 in the United States. LKFS is the same thing under a different name.

Two features make it different from a simple average:

K-weighting. The signal is filtered before measurement, with a high-shelf boost above about 2kHz and a high-pass below about 100Hz, approximating the ear's greater sensitivity in the midrange. So a bass-heavy track measures quieter than its raw RMS suggests, which matches what you hear.

Gating. Passages more than 10LU below the running average are excluded from the integrated figure. Without this, a film with long quiet stretches would measure very low and be turned up until the loud scenes were painful. The gate makes the number describe the programme's typical level rather than its average including silence.

Three readings are reported:

- **Momentary** — a 400ms window.
- **Short-term** — a 3-second window. The useful one while mixing.
- **Integrated** — the whole programme, gated. The one delivery specifications name.

Loudness range (LRA) describes the spread between quiet and loud passages. A heavily compressed pop master might read 3LU; a film mix might read 15 to 20LU.

The numbers, as they stand

These change, and any list is a snapshot. Check the platform's own documentation before a delivery.

- **Spotify, YouTube, Amazon Music** — around **−14 LUFS** integrated.
- **Apple Music** — around **−16 LUFS**.
- **Podcasts** — **−16 LUFS** for stereo is the usual recommendation, with −19 sometimes specified for mono.
- **European broadcast (EBU R128)** — **−23 LUFS**, with a tolerance of ±0.5 or ±1.
- **US broadcast (ATSC A/85)** — **−24 LKFS**.

The broadcast figures are much lower because broadcast preserves dynamic range for programmes that contain both whispering and explosions, and because it is a legal requirement in several countries that advertisements do not jump in loudness.

True peak, and why −1 is not paranoia

Alongside the loudness target, specifications give a **true peak** ceiling, usually **−1 dBTP** and sometimes −2 for lossy delivery.

A sample-peak meter reports the highest stored sample. The analogue waveform reconstructed between samples can be higher than any individual sample — **inter-sample peaks** — and a lossy encoder's output can overshoot further still, because encoding and decoding do not preserve the exact waveform. A file peaking at −0.1 dBFS can therefore clip in a listener's decoder even though it never clipped in your session.

Leaving 1dB of true-peak headroom costs nothing audible and removes the problem.

Hitting the target

Mix to sound right first, measure second, and adjust with a gain change rather than by squashing.

If your mix measures −20 LUFS against a −16 target, raise it 4dB and check the true peak. If that would clip, you need a limiter — but only for the last couple of decibels. Reaching for 8dB of limiting to hit a number is where dynamics go.

```
# measure
ffmpeg -i mix.wav -af ebur128=peak=true -f null -

# two-pass normalisation to -16 LUFS with a -1.5 dBTP ceiling
ffmpeg -i mix.wav -af loudnorm=I=-16:TP=-1.5:LRA=11:print_for-
mat=json -f null -
ffmpeg -i mix.wav -af
loudnorm=I=-16:TP=-1.5:LRA=11:measured_I=-20.3:measured_LRA=9.1
:measured_TP=-3.2:measured_thresh=-30.9:linear=true out.wav
```

The two-pass form matters. In one pass, `loudnorm` works dynamically and can pump audibly. Measure first, feed the measurements back, and with `linear=true` it applies a single gain change — which is what you want.

Podcast and video specifics

For speech, the number to watch while mixing is **short-term**, not integrated. A podcast that averages −16 LUFS with one guest at −22 and another at −11 is a bad podcast with a correct number. Balance the speakers against each other first, then set the whole thing to the target.

For video, the dialogue is the anchor. Mix the dialogue to sit consistently, place music and effects relative to it, then measure the programme.

The free path

- **ffmpeg**'s `ebur128` filter and `loudnorm` do the measurement and the correction, free, and are what many web tools wrap.
- **Youlean Loudness Meter** has a free version that is a full EBU R128 meter.
- **Audacity** includes a loudness normalisation effect with a LUFS target.

- **Ardour** has built-in loudness analysis and reports integrated LUFS and true peak per range.
- **DaVinci Resolve**'s free Fairlight page has a compliant loudness meter with presets for the broadcast standards.

The limitation

LUFS is a model of perception, calibrated on a particular set of programme material, and it does not always match what people hear. Two tracks at the same integrated LUFS can sound different in loudness, particularly when one is dense and the other sparse, or when their spectral balances differ substantially. K-weighting is a broad approximation of one equal-loudness contour, and the ear's response changes with level.

And normalisation is not universal. It can be switched off by the listener on several services, applies differently to albums and to single tracks, and is applied inconsistently across the platforms that do it. So a delivery target is a sensible convention rather than a guarantee, and the real argument for respecting it is that over-compressing to beat it gains nothing and costs the mix.

BEFORE YOU MOVE ON

Why does a delivery specification ask for a true-peak ceiling of −1 dBTP rather than simply requiring that no sample reaches 0 dBFS?

A. Because the loudness measurement is gated, so peaks occurring in gated-out passages are excluded from the integrated figure and need separate headroom.

B. Because K-weighting boosts frequencies above 2kHz before measurement, so a bright programme measures a higher peak than its sample values show.

C. Because a platform that normalises playback may raise a quiet master toward its target rather than only lowering a loud one, and a file already sitting at 0 dBFS has no headroom left for that gain increase to be applied into.

D. Because the analogue waveform reconstructed between samples can exceed the highest stored sample, and a lossy encoder's output can overshoot further still.

62 · 10 min

What survives a phone speaker

THE ONE THING TO KEEP

A phone speaker produces little below 500 to 700Hz, so bass is inferred from its harmonics through the missing-fundamental effect and a pure sine sub disappears entirely — which is why the 1 to 4kHz band, where intelligibility lives, is the one that has to be right.

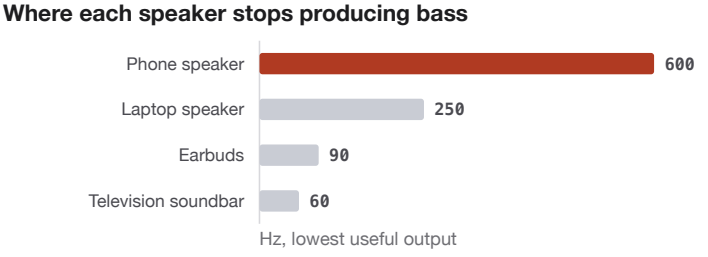
Your mix will be heard on something worse than what you mixed on

A phone speaker is a few millimetres across and sealed into a case. Its useful output typically begins somewhere around **500 to 700Hz** and falls away sharply below that. A laptop speaker does a little better, often reaching 200 to 300Hz. A pair of earbuds and a decent television soundbar do better still.

Whatever you mixed on, a substantial share of your audience will hear your work through one of those. Nothing below the speaker's limit exists for them: the bass is not quiet, it is absent.

The missing fundamental

Play a 100Hz note on a phone speaker and you will still perceive the pitch, even though the speaker cannot produce 100Hz.



Below its limit a speaker produces nothing at all: the bass is not quiet, it is absent. A pitch can still be heard from its harmonics through the missing-fundamental effect, which is why a pure sine sub disappears on a phone and a saturated bass does not. The band from 1 to 4 kHz survives every system and carries intelligibility, which is where the care belongs.

The reason is that a real sound at 100Hz is accompanied by harmonics at 200, 300, 400Hz and up. Those harmonics do come through, and the auditory system infers the fundamental from the spacing between them. This is the **missing fundamental** effect, and it is why a small speaker can convey a bass line at all.

Two consequences for mixing:

- **A pure sine sub-bass disappears entirely** on a small speaker, because it has no harmonics to infer from. If low-end content matters, it needs harmonic content above 500Hz — which is what saturation and distortion on a bass are actually for.
- **Low-frequency energy you cannot hear on a phone is still in the file**, eating headroom in the limiter and making the mix quieter after normalisation. This is the practical argument for high-passing everything that does not need low end, not just the voice.

What survives, and what to protect

The range from about **1 to 4kHz** survives every playback system and carries speech intelligibility. If the mix works there, it works everywhere. If the mix depends on anything below 300Hz to sound full, it will sound thin to most of your audience.

This tells you where to spend care: consonant clarity, the relationship between dialogue and everything else in the midrange, and whether the music bed is competing in that band.

Checking, in three steps

1. A phone speaker, at arm's length, at low volume. Not earbuds — the built-in speaker. This is the most brutal and most representative test available, and it costs nothing. Listen for: can you follow the words, does the music vanish or dominate, is anything unpleasantly harsh.

2. Mono. Covered in the capture module for phase reasons, and it applies again here because most small speakers are single. One click.

3. Quietly. Equal-loudness contours mean the ear is far less sensitive to bass and treble at low levels than in the middle. A mix balanced at a comfortable monitoring level will sound midrange-heavy when quiet — and most listening is quiet. Mixing at moderate to low levels most of the time produces mixes that translate better, and it protects your hearing, which is the other reason to do it.

Reference tracks

The most reliable single technique: import two or three professionally finished pieces in the same genre, level-matched to your mix, and switch between them.

Level-matching is the step people skip and it invalidates the comparison if you skip it: louder sounds better, so an unmatched reference always wins or always loses for the wrong reason. Match by integrated LUFS using the meter from the loudness lesson, then compare.

Listen for the relationships rather than the tone: how loud is the voice against the music, how much low end is there, how bright are the consonants. You are checking your decisions against people who had better rooms and better monitoring.

Your room is lying to you

Every domestic room has strong low-frequency modes — peaks and nulls of 10dB or more at specific frequencies, determined by its dimensions. If your

listening position sits in a null at 80Hz, you will add bass to compensate and your mix will be bass-heavy everywhere else. If it sits in a peak, the opposite.

Two cheap mitigations. **Headphones** have no room, and while they misrepresent stereo width and bass extension in their own ways, they are consistent — and a known error is more useful than an unknown one. **Multiple systems** average out individual errors: monitors, headphones, a laptop, a phone, the car. Anything that sounds right on four different systems is right.

The free path

Everything here is free. A phone you already own, headphones you already own, **ffmpeg** or any editor for level-matched references, and a mono button. **REW** measures your room's frequency response at the listening position for the cost of a measurement microphone, and even a rough measurement tells you where the big dips are.

The limitation

Translation checking tells you what is wrong and not what to do about it. A mix that sounds thin on a phone might need more midrange, less low end, a different arrangement, or nothing at all — because some material genuinely cannot survive a phone speaker and pretending otherwise ruins it everywhere else.

There is also no such thing as a mix that is optimal on every system. Decisions that help a phone — more midrange, controlled dynamics, less sub — cost something on a good system. Professional practice is to mix for the primary listening context and check that nothing breaks elsewhere, rather than to average across all of them and satisfy none.

BEFORE YOU MOVE ON

A mix contains a deep sine-wave sub under a title sequence. It is clearly audible on monitors and completely absent on a phone, while a bass guitar in the same register comes through as a recognisable pitch. What explains the difference?

A. The bass guitar is louder in the mix at those frequencies, and the phone speaker has a steep enough roll-off that a few decibels of level determines whether content crosses its threshold.

B. The phone's playback chain applies dynamic range compression that boosts quiet low-frequency content, and a sustained sine is treated as noise by the gate in that processing.

C. The bass guitar's harmonics at 200Hz and above do reach the speaker, and the ear infers the fundamental from their spacing; the sine has no harmonics to infer from.

D. The sine is a single frequency and therefore excites only one point on the speaker's diaphragm, while the guitar's broadband content excites the whole surface and produces enough total displacement to be audible even below the speaker's rated response.

63 · 10 min

The chain, and why the order is not arbitrary

THE ONE THING TO KEEP

Each stage changes what the next one sees, so repair precedes processing, subtractive EQ precedes compression, de-essing follows it, and the master carries only loudness and safety — anything more there is a decision that belonged further up.

The chain, and why the order is not arbitrary

A voice track goes through the same sequence of stages in almost every professional context. The order is not convention; each stage changes what the

next one sees, and getting it wrong makes every subsequent decision harder.

```
1  gain staging        get the track to a sensible working lev-
el
2  editing and repair  cuts, room tone, clicks, plosives, loud
breaths
3  broadband noise     the least reduction that solves the
problem
4  subtractive EQ      high-pass, remove resonances and mud
5  compression         control the dynamic range
6  de-essing           after compression, because compression
causes it
7  additive EQ         presence and air, on a now-stable signal
8  saturation          optional, for density
9  reverb / space      on a send
10 bus processing      gentle glue compression across the mix
11 loudness and limit  the final target and true-peak ceiling
```

Why each boundary is where it is

Repair before processing. A click, a plosive or a chair creak fed into a compressor triggers gain reduction that ducks the words around it. Fix the event first and the compressor never sees it.

Noise reduction early. Every later stage — EQ boosts, compression, saturation — amplifies noise along with signal. Removing it first means you are not fighting it four times.

Subtractive EQ before compression. A compressor responds to total energy, and low-frequency rumble carries a great deal of it. Compress before high-passing and the rumble drives the gain reduction, so the voice ducks every time a lorry passes. High-pass first and the compressor responds to the voice.

The voice chain, and the reason each boundary is there

1 Repair and edit	before a compressor reacts to a chair creak
2 Noise reduction	the least that solves it
3 Subtractive EQ	high-pass, or rumble drives the gain reduction
4 Compression	3:1, and 3–6 dB on the loudest phrases
5 De-essing	after compression, which is what caused it
6 Additive EQ	presence, on a now-stable signal
7 Reverb, on a send	a finished voice, placed somewhere
8 Loudness and limit	a gain change, not a squash

Each stage changes what the next one sees. Compress before high-passing and the voice ducks every time a lorry passes, because a compressor responds to total energy and rumble carries a lot of it. Boost 4 kHz before compressing and the compressor triggers harder on the band you just raised, partly undoing the boost. These are defaults rather than laws, and a departure is fine as long as you can say why.

De-essing after compression. Compression turns down the loudest parts; sibilants are rarely the loudest parts, so afterwards they stand relatively higher. De-essing first means re-creating the problem in the next stage.

Additive EQ after compression. Boost 4kHz first and the compressor now triggers harder on the boosted band, which partly undoes the boost. Compress into a stable signal, then shape it.

Reverb last among the voice processing. Reverb should be applied to a finished voice. Compressing after a reverb squashes the tail and pumps the room; EQing after it changes the space's tone rather than the voice's.

These are defaults, not laws. A deliberate departure — a compressor before EQ to catch a specific dynamic problem, a touch of reverb before a saturator for a particular effect — is fine as long as you can say why.

Gain staging through the chain

Each stage should hand on a level near where the next expects it. Plugins modelled on analogue equipment often have a sweet spot around −18 dBFS, and a signal arriving 15dB hotter will behave differently from the way its designer intended.

Practically: use a gain trim at the start, and use the output or make-up control on each plugin to leave the level roughly as it arrived. If the level is climbing 6dB at every stage, you will discover it at the limiter and have to unpick everything.

Track, bus, master

Three layers, each with a job.

Track processing solves problems specific to one voice: this speaker's sibilance, this microphone's proximity, this room's resonance.

Bus processing makes several elements behave as one. A dialogue bus with 1 to 2dB of gentle compression across all speakers makes a conversation feel like one scene rather than a series of separate recordings.

Master processing is loudness and safety only — the limiter, the true-peak ceiling, perhaps a very gentle overall EQ. Anything more on the master is a sign that a decision should have been made further up.

The commonest structural mistake is doing bus work on individual tracks: compressing each speaker hard to make them match, rather than balancing them and letting a bus compressor do the last decibel.

Working order, as opposed to signal order

The signal chain and the order you *build* it in are different.

1. **Edit first, entirely.** Get the content right before touching a plugin. There is no point in EQing a line you are going to cut.
2. **Balance with faders, no processing.** Set relative levels. A surprising amount of what people reach for a compressor to fix is a fader problem.
3. **Then process**, cheapest move first — a fader, then an EQ cut, then a compressor, and only then anything more elaborate.
4. **Then automate** the things a static process cannot handle.
5. **Then measure** and set the delivery loudness.

Mixing in this order means each tool is solving what is genuinely left rather than what an earlier tool created.

Two habits worth building

Bypass everything periodically. After an hour, you will have adapted to your own decisions. A raw-versus-processed comparison, level-matched, tells you whether you have improved the track or merely changed it — and sometimes you have made it worse and become used to it.

Leave it overnight. Fresh ears catch in thirty seconds what tired ears miss for an hour. Every experienced mixer says this and it remains the least followed advice in audio.

The free path

Ardour and **Reaper** give you full bus routing, plugin chains and automation. **Audacity** does effects destructively per clip, which makes it awkward for chain work but perfectly usable for a single voice track. The **LSP** and **Calf** plugin suites cover EQ, compression, de-essing, gating and limiting. **DaVinci Resolve**'s free Fairlight page gives a complete mixing environment with a fixed channel strip in the professional order — which is itself a useful thing to look at, because the order it enforces is the one in this lesson.

The limitation

A chain is a starting structure, not a recipe. Every recording arrives with its own problems, and a template applied blindly does harm: a high-pass at 100Hz on a deep voice recorded at a distance removes body it needed; a compressor on a track already even in level only adds noise and pumping.

The discipline that prevents this is to ask, at each stage, what problem this stage is solving on *this* track. If the answer is "it is what I always do", switch it off and see whether anybody notices.

BEFORE YOU MOVE ON

A voice track ducks noticeably every time a lorry passes outside, although the rumble itself is barely audible. The chain is: compressor, then high-pass filter, then EQ. What is happening?

A. The compressor responds to total energy, and low-frequency rumble carries a lot of it, so the gain reduction is being driven by content the high-pass has not yet removed.

B. The high-pass filter's phase response is shifting the rumble into the compressor's detection band, which would not happen with a linear-phase filter in the same position.

C. The compressor's release is too long, so each rumble holds the gain down through the following words; shortening it would resolve the ducking without reordering the chain.

D. The high-pass is set too low to remove the rumble, and moving it up to 200Hz would solve the problem in its current position without any need to place it before the compressor, since the audible symptom is the filtered output rather than the compressor's behaviour.

CASE STUDY · MODULE 7

The music bed that sounded right in the office

A ninety-second health department film in Marathi, mixed in Pune, to be watched on phones in Nanded district.

The film announces a free screening camp: where it is, which days, who should come. It goes out on WhatsApp and is played by ASHA workers on their own phones, frequently held up in front of two or three people in a courtyard with a fan running.

Sneha mixed it the way the module says to. Dialogue anchored and consistent, the music bed sitting about seventeen decibels under the speech while anyone is talking, integrated loudness at minus sixteen LUFS, true peak at minus one point five.

The communications officer listened on the studio monitors at a comfortable level and said the music felt thin and unimportant. He asked for it up six decibels. On those monitors, in that room, he was right — it did sound thin.

Sneha checked the same file on a six-thousand-rupee phone at arm's length, at low volume, with the studio's own fan switched on. At the officer's level the Marathi voiceover was hard to follow, and the reason was specific. The bed had a bright string line living between one and a half and three kilohertz, which is squarely where speech intelligibility lives. The phone produced almost nothing below about six hundred hertz, so everything that had given the bed its body and weight on the monitors was simply absent — what survived on that device was its midrange, and its midrange was the voice's.

The audience made it worse rather than better. A large share of the people who would watch this were over sixty, and the ability to separate speech from a competing sound falls with age and with hearing loss. Complaints about background music drowning speech come disproportionately from exactly those listeners.

Three ways to handle it. Do as asked, sign off today, ship on Friday, and lose the dates for the people most likely to need the camp. Refuse and explain,

which is a supplier telling a government client that his ears are wrong with a launch four days out. Or change the arrangement rather than the level.

She took the third. She cut three decibels with a wide bell at two kilohertz in the music bus, which takes away the part that was in the way and almost none of what the bed is actually made of. She raised the bed four of the six decibels he had asked for. She put a two-second hole in the music under the single line that carries the camp's dates, edited into the arrangement rather than ducked by a compressor. And she took the review to the phone.

She had tried a ducker first and taken it off again. A sidechain can only react after the fact, so it was pulling the bed down on every breath and every lip smack and letting it surge back between sentences — the pumping that makes an automated intro recognisable as one. Riding the level by hand meant she could start the move two hundred milliseconds before a line began and release it slowly, which is what a human can do and a detector cannot. On a ninety-second film there were eleven moves to draw, and it took less time than tuning the ducker had.

The deeper problem was never the level. Choosing a bed with a bright string line sitting in the intelligibility band had made every later decision harder, and a track with sparse midrange and its energy below five hundred hertz or above six kilohertz would have needed almost none of this work. Sneha now listens to candidate tracks on the phone before the edit is locked, which costs two minutes and has changed which tracks get chosen.

STOP HERE

Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

She brought the phone to the next review with two versions on it and played his version first. He asked her to turn the music down before she had said anything at all. The delivered mix has the bed four decibels louder on the me-

ter than her original and measurably less energy between one and a half and three kilohertz; the dates line is clean; integrated loudness stayed at minus sixteen LUFS, so nothing was gained or lost to any platform's normalisation. The department has since added one line to the brief for all campaign films: mixes are reviewed on a phone speaker at low volume, and the review copy is played from the phone. Sneha says the disagreement was never about taste — two people were listening to two different films, one full-range and one midrange-only, and only one of those was the film the audience would hear.

Why did the same bed sound thin on monitors and overbearing on the phone?

A phone speaker produces little below roughly five to seven hundred hertz, so the low end that gave the bed its weight on monitors does not exist on that device. What is left of the music is its midrange — and the one-to-four-kilohertz band is where speech intelligibility lives, so on the phone the bed is competing directly with the voice rather than sitting under it. Low listening level makes it worse again, because the ear is less sensitive to bass and treble when things are quiet.

Why was a three-decibel cut at two kilohertz worth more than a six-decibel level drop?

It removes only the part that was in the way. The music's character does not live in that narrow region, so it loses very little apparent presence, while the voice gains a clear channel. A broad level drop removes the parts that were not competing as well as the parts that were, which is why the bed then sounds as if it may as well not be there.

What did bringing a phone into the review change about the disagreement?

It made the listening condition the same for both people. The client's perception was accurate for his room and irrelevant to the delivery, and no amount of explaining would have moved it. Playing his own preferred version on the device the audience uses converted a difference of taste into a difference of equipment, and he heard the failure himself instead of being told about it.

MODULE 7 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 Why does a high-pass filter belong before the compressor rather than after it?

A. A high-pass filter after compression introduces pre-ringing
B. Compressors cannot process frequencies below 100 Hz accurately
C. The make-up gain would otherwise amplify the filtered band a second time
D. A compressor responds to total energy, and rumble carries a lot of it

2 Why does a de-esser sit after the compressor?

A. A de-esser cannot work on a signal with more than about 6 dB of dynamic range left
B. Compression turns down the loudest parts, and sibilants are rarely the loudest
C. Sibilance is caused by the microphone, so it must be removed first
D. The compressor's sidechain would otherwise trigger on the de-esser

3 Background hiss swells and recedes between sentences. What is happening?

A. Nothing is above the threshold in the gaps, so make-up gain lands on the noise
B. The noise reduction is set too low and keeps releasing in the gaps between phrases
C. The de-esser is ducking the whole voice on each sibilant
D. The reverb send has no high-frequency damping on its return

4 A dialogue reverb is making the words hard to follow. What is the first control to change?

A. The decay time, shortened to under half a second
B. The wet level, reduced until the tail is inaudible
C. The pre-delay, increased to 20 to 40 milliseconds
D. The high-frequency damping, opened up above 8 kHz

5 Why does manual level riding beat a sidechain ducker on edited material?

A. A ducker adds latency that a human automation curve does not
B. Duckers cannot be set to a release longer than about two hundred milliseconds
C. A human can start the move before the line, where a ducker can only react
D. Sidechain compression is not available in free editing software

6 You master a podcast 6 dB louder than the platform's target. What is the result?

A. The platform turns it down 6 dB and the extra compression remains
B. It plays 6 dB louder and stands out against other programmes
C. The platform rejects the file for exceeding its own true-peak ceiling
D. Only the loud passages are reduced, so the dynamics are restored

MODULE 8

Machines that make sound and motion

Generative and analytic models in a motion and sound pipeline: what a text-to-speech system cannot read off the page, what a voice clone's embedding does and does not capture, why a five per cent word error rate lands on the words that matter, what a separator is estimating, and where copyright, licensing and disclosure are genuinely unresolved.

> BY THE END OF THIS MODULE YOU CAN
>
> Use a generative or analytic model inside a motion and sound pipeline — synthesised speech, a cloned voice, automatic captions, source separation, generated music, interpolated or upscaled frames — and state for each what it is estimating, which characteristic failure to look for and where, what consent or licence the use requires, and which parts of the surrounding law are unsettled rather than merely complicated

64 · 10 min

What a text-to-speech system cannot read

THE ONE THING TO KEEP

Written text does not encode stress, and stress is meaning — so a synthesiser guesses prosody from statistical regularities and produces a convincing reading of the wrong sentence, which is why a pronunciation lexicon and SSML are quality control rather than decoration.

Two stages, and the second one is not where the problems are

Most text-to-speech systems work in two parts. A first model turns text into an intermediate acoustic representation — usually a mel-spectrogram, a time-frequency picture of the intended sound. A second model, the **vocoder**, turns that picture into a waveform. Newer systems compress the two into one by predicting tokens of a learned audio codec directly.

The vocoder is largely a solved problem: modern ones produce clean, natural-sounding audio. Almost everything that makes synthesised speech sound wrong happens in the first stage, and it is not about the voice at all. It is about **prosody** — the rhythm, stress and intonation that carry meaning beyond the words.

What the text does not contain

Say this sentence aloud seven times, stressing a different word each time:

> *I never said she stole my money.*

Each version means something different. *I* never said it — someone else did. I never *said* it — I implied it. She stole my *money* — not my car. The written sentence contains none of that. The stress is the meaning, and it is simply absent from the input.

A TTS model has to guess, and it guesses from statistical regularities in its training data. On ordinary declarative prose it does well. On anything where the emphasis carries the point — a contrast, a correction, an aside, a list where one item is the surprising one — it produces a plausible reading of the wrong sentence.

The specific failures

Homographs. Words spelled the same and pronounced differently, disambiguated only by meaning:

- *read* — "I read it yesterday" against "I will read it"
- *lead* — the metal against the verb
- *live* — "live broadcast" against "where they live"
- *bass*, *minute*, *wind*, *tear*, *close*, *record*, *desert*, *row*

Models get these right most of the time from context and wrong in a way that stops a listener.

Numbers and dates. "1996" is *nineteen ninety-six* as a year and *one thousand nine hundred and ninety-six* as a quantity. "3/4" is a fraction, a date, or a ratio. "10-12" is a range or a subtraction. Phone numbers, version numbers, prices and measurements all have conventions the text does not state.

Names and jargon. Proper nouns are where the errors concentrate, and they are the words a listener is most likely to be listening for.

Acronyms. Some are spelled out, some are pronounced as words, and there is no rule in the text: NASA is a word, FBI is letters, SQL is both depending on who you ask.

Long-form structure. Most systems synthesise a sentence or a chunk at a time with no memory of the paragraph. So the emphasis does not build, a list does not fall at the end, and a long read has a flat, sentence-by-sentence sameness that listeners describe as tiring without being able to say why.

SSML: the controls that exist

Speech Synthesis Markup Language is the standard way of supplying what the text lacks. Support varies by engine, but the core is widespread:

```
<speak>
  The report was published in <say-as interpret-as="date" for-
mat="y">1996</say-as>.
  She <emphasis level="strong">never</emphasis> agreed to it.
  <phoneme alphabet="ipa" ph="ˈsiːkwəl">SQL</phoneme> is one
option.
  <break time="400ms"/> The alternative is simpler.
  <prosody rate="92%" pitch="-1st">This part is an aside.
</prosody>
</speak>
```

A pronunciation lexicon — a list of your names and terms with their phonetic spellings — is worth building once for any project with recurring vocabulary, and it removes the most damaging category of error permanently.

Practical technique

- **Write for the ear.** Shorter sentences, one idea each. A clause that a human reader would handle with a subtle pitch move will defeat the model.
- **Punctuate for timing.** A full stop produces a longer pause than a comma. Splitting a long sentence into two changes the delivery more reliably than any tag.
- **Spell things out in the input.** Writing "nineteen ninety-six" is often quicker than a tag and always works.
- **Listen to the whole thing.** Errors are local and a spot check will miss them.
- **Chunk long reads at paragraph boundaries**, and check the joins for a level or timbre shift.

The free path

- **Piper** — fast, local, neural, dozens of languages, runs on a Raspberry Pi. The sensible default for free self-hosted TTS.
- **eSpeak NG** — formant synthesis, very robotic, tiny, over a hundred languages, and still valuable for languages no neural model covers.
- **Coqui TTS** — a widely used open-source toolkit; note that the company behind it shut down, so the repository is no longer actively maintained even though the models and code remain available and usable. Check the project's state before building on it.
- **Mimic 3**, **MaryTTS** — other open options.
- **Whisper** is the counterpart on the recognition side, covered in a later lesson.

The limitation

Synthesised speech is now good enough that the remaining errors are more dangerous than the old obvious ones. A robotic voice signalled clearly that this was a machine; a natural voice confidently mispronouncing a drug name or a person's surname does not signal anything, and a listener may not notice.

There is also no reliable way for the model to know what it got wrong. It does not have a confidence score for prosody, and it will read a sentence with exactly the wrong emphasis in a perfectly convincing voice. The only check is a person listening to the output against the intended meaning, which is unglamorous and is the actual quality control step.

BEFORE YOU MOVE ON

A synthesised narration reads a list of seven product features in a uniform, slightly tiring way, although each sentence on its own sounds natural. What is the structural cause?

A. The vocoder is producing identical spectral characteristics for each sentence, because it has no conditioning on position within a document.
B. Most systems synthesise a sentence or chunk at a time with no memory of the paragraph, so emphasis cannot build and a list does not fall at the end.
C. The sentences are too similar in length and structure, and varying the sentence length would give the model enough signal to produce contour variation.
D. The model has not been given an SSML prosody tag, and without an explicit rate or pitch instruction every synthesiser defaults to a fixed monotone contour derived from its training corpus's mean fundamental frequency.

65 · 11 min

The vector that is supposed to be you

THE ONE THING TO KEEP

A speaker embedding captures timbre — the resonances and average pitch that make a voice recognisable in a word — and not idiolect, the timing and stress habits of how a particular person speaks, which is why a clone convinces for three seconds and not for thirty.

A vector that is supposed to be you

Voice cloning works by separating *what is being said* from *who is saying it*.

A **speaker encoder** is trained on many thousands of voices with an objective that pushes recordings of the same person close together in a vector space and different people apart. Given a few seconds of speech, it produces an embed-

ding — commonly a few hundred numbers — that is meant to capture speaker identity and nothing else.

A synthesiser is then trained to produce speech conditioned on both the text and that embedding. At inference, you supply a short sample of a target voice, the encoder turns it into a vector, and the synthesiser speaks your text in something resembling that voice.

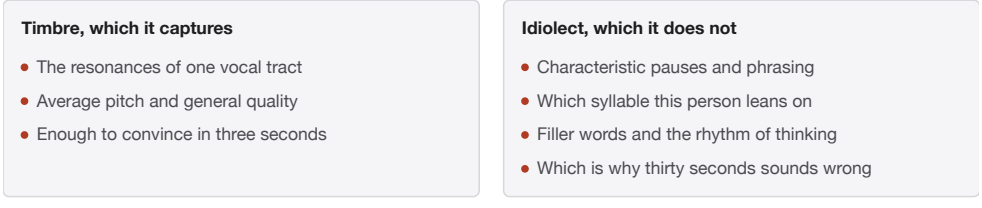
This is **zero-shot cloning**: no training on the target, seconds of reference audio, a result in seconds. **Fine-tuning** — actually training on ten minutes to an hour of the target — produces a closer match and takes compute and time.

What the embedding captures, and what it does not

It captures **timbre**: the resonances of a particular vocal tract, the average pitch, the general quality of the voice. That is most of what makes a voice recognisable in a single word.

It does not capture **idiolect**: the way a specific person actually speaks. Their characteristic pauses. Whether they run sentences together or break them. Which syllable they lean on. Their filler words. The rhythm of their thinking.

What a speaker embedding carries



A clone is the right timbre performing somebody else's habits — the generic behaviour the synthesiser learned from its training distribution. Play three seconds to a friend of the speaker and they say yes, that is them; play thirty and they say something is off and cannot name it. Fine-tuning on longer material narrows the gap and does not close it.

This is precisely why a good clone sounds like the person and not like them. Play three seconds to somebody who knows them and they will say yes, that is them. Play thirty seconds and they will say something is off, and struggle to name it. The timbre is right and the behaviour is somebody else's — the generic behaviour the synthesiser learned from its training distribution.

Fine-tuning on longer material narrows the gap because the model picks up some timing and stress habits along with the timbre. It does not close it.

Where clones degrade predictably

- **Under-represented languages and accents.** The encoder was trained on a distribution; voices far from it get embeddings that are less distinctive, and the output drifts toward the training average. In practice this means a clone of a speaker with a strong regional accent in a well-represented language often comes out noticeably softened, and clones in low-resource languages are much weaker.
- **Emotional range.** Reference audio of somebody speaking calmly gives you a calm clone. Asking it to shout produces a shout in something that no longer sounds like the person, because the mapping between calm and loud in a real voice is specific to that person's anatomy.
- **Poor reference audio.** The encoder cannot separate the speaker from the room. A reference recorded in a bathroom produces a clone with a hint of bathroom in it, because the embedding absorbed some of it.
- **Long output.** Consistency drifts across a long read, which is why production use chunks and re-conditions.

Detection, and why it is not a solution

Two families of defence are discussed, and both have real limits.

Anti-spoofing classifiers are trained to distinguish synthetic from genuine speech. They perform well on the generation methods they were trained against and degrade on new ones, which is the ordinary arms-race problem: the classifier is always fitted to yesterday's generators.

Watermarking embeds an imperceptible signal in generated audio. It works when the audio reaches the detector broadly intact. It is weakened or removed by the ordinary things audio undergoes — heavy lossy compression, re-recording through a speaker and microphone, pitch shifting, deliberate noise addition — and a determined adversary can strip it. Vendors publish robustness

figures under specific transformations; those figures are not a claim about an adversary who is trying.

The honest summary: watermarking helps with accidental and casual misuse and with platform-side filtering. It is not evidence of authenticity, and the absence of a watermark proves nothing at all.

The technical does not settle the ethical

The next lesson covers consent in detail, and it is worth stating here why it cannot be an afterthought. Everything above describes how easy this is: a few seconds of audio, freely available for anybody who has ever spoken publicly, and a free model.

The capability is available to everyone. The restraint has to come from the practice, because it is not coming from the difficulty.

The free path

- **Coqui XTTS** — multilingual zero-shot cloning; the models remain available with the maintenance caveat from the previous lesson, and note that the XTTS weights carry a licence that restricts commercial use, so read it rather than assuming.
- **OpenVoice** — open-source tone-colour cloning with separate control of style.
- **Piper** — can be fine-tuned on a target speaker for a self-hosted voice.
- **RVC** and similar voice-conversion tools convert an existing performance into a target timbre, which preserves the original speaker's timing and often sounds better than text-to-speech cloning for exactly the idiolect reason above.

Licences differ sharply across these projects and several restrict commercial use. Read the licence for each model, not just the repository.

The limitation

The most important one is not technical. A cloned voice is convincing enough for fraud, and the documented cases — relatives called by a synthesised voice asking for money, an employee instructed by a synthesised executive — turn on the fact that three seconds is enough to convince somebody who is not expecting it.

There is no technical countermeasure that an ordinary person can deploy. The practical defence that security guidance consistently recommends is procedural rather than technical: an agreed phrase within a family, a call back on a known number, a second channel for any instruction involving money. That is an unsatisfying answer and it is the one that works.

BEFORE YOU MOVE ON

A colleague says a clone of their voice 'sounds like me but is not me', and cannot say what is wrong. What does the architecture predict they are hearing?

A. The vocoder is introducing artefacts in the high frequencies that are below conscious detection but sufficient to make the voice feel unfamiliar.

B. The embedding was taken from too short a reference, so it has captured an average of the speaker rather than their voice; a longer reference would resolve it.

C. The reference recording carried the room, so the clone has a trace of that acoustic on it which reads as subtly wrong.

D. The timbre is right and the speech behaviour — pauses, stress placement, rhythm — is the synthesiser's generic default rather than theirs.

66 · 11 min

A cloned voice needs a yes you can point to

THE ONE THING TO KEEP

Consent for a voice is scoped — to a project, a set of uses, a period, and whether the model itself is kept — and disclosure to the audience is a separate duty on top of it.

Where the machines actually help

Strip out the marketing and four things have genuinely changed in this workflow.

Speech to text is the dull one and the most useful. Whisper, released with open weights, transcribes and time-stamps audio, and runs offline on an ordinary laptop through whisper.cpp on modest hardware. That gives you captions, a searchable transcript, and subtitle files, free. Captions raise completion rates on muted feeds and, in several contexts, are an accessibility obligation rather than a nicety. Auto captions get proper nouns, place names and technical terms wrong every single time. Read them before you burn them in.

Transcript-based editing — delete a sentence in the text and the audio cuts with it — is a real speed change for interviews and talking-head work.

Generated music. Suno and its competitors will produce a fifteen-second bed that most listeners cannot distinguish from a library track.

Text to speech and voice cloning. ElevenLabs is the reference commercial product. Open alternatives include Piper, which is small and fast enough to run on a Raspberry Pi, and Kokoro, which is small and permissively licensed, alongside a rotating cast of larger models. Check each model's licence individually: several well-known open voice models are released under terms that forbid commercial use, and "the weights are downloadable" is not the same as "you may sell work made with it".

Enhancement models for de-noise, de-reverb and studio-voice processing are much better than the old signal-processing versions, and fail in the same direction: pushed hard, they invent detail. On a heavily processed voice you can hear consonants being reconstructed slightly wrong.

Consent, stated plainly

A voice is a person. Cloning one without agreement is not a technique question, and there is no version of it that is a style choice.

Get the agreement in writing, and make it specific:

- For which project.
- For which uses — this video, or anything the company publishes.
- For how long.
- Whether the trained model itself is kept afterwards, and who can run it.

"You can use my voice for this video" is not permission to hold a voice model on a server for two years. The gap between those two things is where most of the real disputes sit.

Read what the service does with the samples you upload, including for your own voice. And note the second-order risk: a cloned voice on a phone call has already been used to extract money from families and payments from finance departments. Publishing long, clean, isolated samples of a colleague's voice makes that easier. Not a reason to stop working — a reason to think about what you post.

The law is moving in one direction here. Denmark moved to give people a right in their own likeness and voice. Several US states have right-of-publicity statutes that already reach a deliberate soundalike. The 2025 TAKE IT DOWN Act obliges US platforms to remove non-consensual intimate imagery including synthetic material.

Disclosure and ownership

Disclosure

Several countries now require synthetic media to be labelled visibly, not merely tagged in metadata.

India's amended IT Rules for synthetically generated information prescribe visibility, and for audio specifically a label in the opening portion of the audio. China's rules, in force since September 2025, require both a label a person can see or hear and one embedded in the file's metadata. The EU AI Act's transparency provisions require machine-readable marking of generated output and disclosure of deepfakes, on a timetable that has been amended more than once — check the current position rather than trusting any date you read, including this one.

The rule that survives all of these statutes: if a listener could reasonably take the voice for a real person speaking, say that it is not, at the start, in the audio, where they hear it before they believe it. Not in the description. Not in the credits.

Ownership

Generated music sits in the same unsettled place as generated images. In the United States a prompt alone is unlikely to make you an author; your own arrangement, your recorded parts and your edits are yours. Other countries have reached different conclusions on similar facts. Making things with AI sets out the country-by-country shape of it.

Separately from ownership, the training-data question is live: the major labels sued the generated-music companies in 2024, and licensing arrangements have been reported since. That is the shape of it. A lawyer in your country is the person who answers it for your case.

What has actually changed in the work

A one-person team can now caption a video, transcribe an hour of interview, produce a music bed and lay a temporary voiceover in an afternoon — work that used to mean booking people and a room.

What has not changed is the part that was ever the job: deciding what to cut, what the piece is about, whether the music fits the point, and whether the reader can follow it. A generated voice is good at neutral information delivery. It is still audibly wrong at sarcasm, grief and comic timing, because those live in the timing between words, and a real performance still wins there by a distance.

One habit

If you use a generated voice in anything you publish this week, write the disclosure line before you generate the audio, not after you finish the edit. Written first, it is one sentence. Written last, it is a change to a locked timeline, and it does not get made.

BEFORE YOU MOVE ON

A team clones a colleague's voice, with her spoken agreement, to narrate one internal training video. Six months later marketing uses the stored model for a public advert. In the terms of this lesson, what went wrong?

A. Nothing went wrong on the consent side. She agreed to have her voice cloned, and the model is the company's asset once it has been trained.

B. The consent was neither written nor scoped: it never named the project, the permitted uses, a time limit, or whether the trained model would be retained. Agreeing to one video is not agreeing to a stored voice model.

C. The failure is that the advert carried no synthetic-media label, which is the obligation that was breached here.

D. The fix is to delete the voice model now, which resolves the situation.

67 · 11 min

Five per cent, and which five per cent

THE ONE THING TO KEEP

Word error rate is a uniform average over errors that are not uniformly distributed — they cluster on names, numbers and technical terms — and a generative recogniser can produce fluent sentences during silence, so no automatic transcript is publishable unread.

What a WER figure is actually telling you

Automatic speech recognition is scored with **word error rate**:

```
WER = (substitutions + deletions + insertions) / words in the
reference
```

A 5 per cent WER means one word in twenty is wrong. On a 1,000-word video that is fifty errors. Stated as a percentage it sounds like a rounding error; stated as fifty wrong words in ten minutes it sounds like what it is.

The figure also understates the damage, for a structural reason. Errors are not spread evenly. They concentrate on **proper nouns, technical terms, numbers and rare words** — precisely the content words a viewer is relying on the captions for. A transcript can be 95 per cent accurate and get every name in it wrong, and the 5 per cent that failed is the 5 per cent that mattered.

Where accuracy actually goes

- **Overlapping speech.** Two people talking at once defeats most systems, and this is most of a real conversation.
- **Accents and dialects** under-represented in training data. Published research has repeatedly found substantially higher error rates for some

speaker groups than others on the same systems; this is a known and documented disparity, not an anomaly.

- **Background noise and music.**
- **Domain vocabulary.** Medical, legal, technical and organisational terms.
- **Code-switching.** Moving between languages mid-sentence, which is ordinary speech for a large part of the world and poorly handled.

Many systems accept a vocabulary or prompt hint. Supplying a list of the names and terms in your video before transcription is the single cheapest accuracy improvement available.

The hallucination problem

Whisper, the most widely used open model, has a documented failure mode that is different in kind from ordinary errors: during silence, music, or non-speech audio, it can generate fluent text that nobody said. Researchers examining transcriptions have found invented sentences, and the phrase "Thank you for watching" appearing at the ends of clips is a well-known artefact of its training data.

This matters because the output is confident and well-formed. A conventional recogniser fails visibly, producing garbage you can see. A generative one fails invisibly, producing a plausible sentence.

Mitigations that help: setting a voice-activity detector to skip non-speech, using `condition_on_previous_text=False` to stop an error propagating forward, and checking any segment of silence in the output. None of them removes the need to read the transcript.

Captions as a craft, not an export

An auto-generated caption track is a starting point. The conventions that make captions readable are well established:

The conventions that make a caption readable

32 to 42 characters a line	two lines on screen at most
17 to 20 characters a second	about 160 to 180 words a minute
One second minimum	six or seven at the outside
Break at a grammatical boundary	not where the width happened to run out
Clear of the mouth	and of any burned-in text
Name the speaker when unclear	and describe sound, for SDH

An automatic track is a starting point, not an export. Errors concentrate on names, numbers and technical terms — the words a viewer is relying on the captions for — so a transcript can be 95 per cent accurate and get every name in it wrong. Reading a ten-minute transcript takes about four minutes.

- **32 to 42 characters per line**, at most two lines on screen.
- **Reading rate** around 160 to 180 words per minute for adult viewers, often expressed as 17 to 20 characters per second. Faster than that and people are still reading when the caption changes.
- **Minimum one second** on screen, even for one word; **maximum about six or seven seconds**.
- **Break lines at grammatical boundaries** — after a clause, before a preposition — not wherever the width runs out. A line broken mid-phrase measurably slows reading.
- **Do not cover the speaker's mouth** or burned-in text.
- **Identify speakers** when it is not obvious, and describe significant non-speech sound for SDH.

Subtitles assume the viewer can hear and translate or transcribe dialogue. **SDH** — subtitles for the deaf and hard of hearing — additionally carry speaker identification and sound effects. They are different deliverables.

Burned-in or sidecar

Sidecar files — SRT, WebVTT, TTML — are separate and referenced by the player. They can be toggled off, restyled, searched, indexed by search engines, and translated. They are the correct default.

Burned-in captions are pixels. They survive platforms that strip sidecar tracks, they cannot be turned off, they cannot be restyled by a viewer who needs larger text, and they are baked into every version of the file.

Social platforms where video autoplays muted are the real case for burning in, and many accept a sidecar as well. The defensible approach for those is both: burned-in for the feed, sidecar for accessibility.

```
# transcribe locally
whisper-ctranslate2 --model medium --language en --vad_filter
True talk.mp4

# attach as a soft track, no re-encode of the video
ffmpeg -i talk.mp4 -i talk.srt -c copy -c:s mov_text out.mp4

# burn in, with styling
ffmpeg -i talk.mp4 -vf "subtitles=talk.srt:force_style='Font-
Size=22,Outline=2'" out.mp4
```

The free path

- **Whisper** and the faster **faster-whisper** / **whisper.cpp** implementations run locally on modest hardware, are free, and handle many languages.
- **Subtitle Edit** is free and open source, and is a genuinely good caption editor with waveform view, timing tools and format conversion.
- **Aegisub** for styled subtitles.
- **ffmpeg** for muxing and burning in.
- **DaVinci Resolve**'s free edition includes transcription and a subtitle track.

The limitation

Captions are a legal requirement in many contexts, and the requirements differ by country, by sector and by whether the content is broadcast, educational, governmental or commercial. Several jurisdictions have specific standards and enforcement histories. This is not something to resolve from a general

course: check the rules that apply to your jurisdiction and your kind of publication, and take advice if the answer matters.

The technical limitation is simpler and is the one to hold on to: no automatic system is accurate enough to publish unread. Reading a ten-minute transcript takes about four minutes and catches the names, the numbers and the sentences nobody said.

BEFORE YOU MOVE ON

A transcript produced by Whisper contains a well-formed sentence during a fifteen-second passage where nobody is speaking. Why is this failure different in kind from an ordinary recognition error?

A. It is a timing error rather than a recognition error: the model has assigned a correctly recognised sentence to the wrong timestamp, which a re-alignment pass will correct.

B. The model is generative, so it fails by producing plausible text rather than visible garbage, and a reader has no signal that anything went wrong.

C. The error is an insertion rather than a substitution, and insertions are weighted more heavily in the word error rate calculation than other error types.

D. Silence contains no acoustic features, so the model falls back to its language model alone, and a pure language-model output is by definition unrelated to the audio and therefore excluded from the word error rate denominator.

68 · 10 min

Taking a mixture apart

THE ONE THING TO KEEP

Separation estimates a mask per source per time-frequency bin, so where two sources had energy in the same bin the split is a learned guess rather than a recovery — and because the masks partition the mixture, the stems sum back cleanly, which is what makes separate-fix-recombine a gentler repair than processing the whole mix.

Taking a mix apart, and what "apart" means

Source separation takes a finished mixture and produces estimates of its components: vocals, drums, bass, other. Or, for the case that matters most in video work, speech and everything else.

The standard approach is **masking**. Convert the audio to a time-frequency representation — a spectrogram — and for each bin, estimate what proportion of the energy there belongs to each source. Multiply the mixture by that mask and you have an estimate of one source. The masks for all sources roughly sum to one, because the energy has to go somewhere.

Spleeter works this way. **Demucs** is a hybrid: it operates partly on the waveform directly and partly in the spectrogram domain, with a transformer in the middle, which is why it handles transients better than purely spectrogram-based methods.

The word doing the work is "estimate"

Here is the fundamental limit. If a vocal and a guitar both have energy at 800Hz at the same instant, the mixture contains their sum and only their sum. The individual contributions are not recorded anywhere. A separator is *guessing* the split, from learned statistics about how vocals and guitars usually behave.

Where the sources rarely overlap — a bass line and a hi-hat — separation is close to clean. Where they overlap heavily — a vocal and a string pad in the same register — it is a guess, and the guess shows.

The artefacts, and how to recognise them

Bleed. Traces of one source in another's stem. Most audible on the drums, because their transients are broadband and hard to assign.

Watery or phasey texture on separated vocals. This is masking error changing from frame to frame: the estimate is slightly different each analysis window, so the residual moves about. It is the same family of problem as musical noise in denoising.

Transient smearing. Percussive attacks lose their edge, because a transient is spread across many frequency bins and the mask cannot be sharp everywhere.

Reverb tails going to the wrong place. The tail of a vocal's reverb is spectrally similar to the accompaniment and frequently ends up with it, which makes a separated vocal sound drier and more clinical than it was.

The quality metric usually quoted is **SDR** — signal-to-distortion ratio — in decibels, measured against ground truth on a standard test set. Demucs variants report figures in the region of 9 to 10dB SDR for vocals on the MUSDB18 benchmark. That is genuinely good and it is also an average over test material; on your specific mix it may be considerably better or worse.

What it is genuinely useful for

- **Rescuing dialogue from a music bed** on archive material where the stems are lost.
- **Removing a cough, a phone or a door** from a take that is otherwise good, by separating, editing the offending band, and recombining.
- **Repurposing.** Pulling a clean vocal for a lyric video, or an instrumental for a backing track.
- **Practice and transcription.** Isolating a part to learn it.

- **Dialogue isolation in a noisy environment**, where speech-specific models now outperform general separators.

What it is not good for

Anything where the result will be scrutinised at high quality and at full level. A separated vocal placed nakedly in a quiet mix reveals every artefact; the same vocal sitting inside a new arrangement hides most of them. This is the practical rule: separation output survives being buried and does not survive being exposed.

Recombining without phase problems

A useful property of mask-based separation: the stems sum back to something very close to the original, because the masks were constructed to partition the mixture. So you can separate, process one stem, and add the others back, and the parts you did not touch are essentially unchanged.

This is the technique behind "remove the cough": separate, fix the stem containing it, sum. It is far less destructive than processing the whole mixture.

The free path

- **Demucs** — open source, state of the art for music separation, runs on CPU slowly and on a GPU quickly.
- **Spleeter** — older, faster, lower quality, still useful for a rough split.
- **Ultimate Vocal Remover** — a free graphical front end bundling several models, which is the easiest starting point.
- **DeepFilterNet** and **RNNoise** — speech enhancement rather than music separation, free, and better for the dialogue case.
- **ffmpeg** has no learned separator, but `afftdn` and the phase-cancellation trick for centre-channel removal are crude free fallbacks.

The centre-channel trick is worth knowing as a baseline: subtracting one stereo channel from the other cancels anything panned dead centre, which on many older recordings is the lead vocal. It also cancels the bass and kick, and

leaves a mono result. It costs nothing and demonstrates how much better the learned methods are.

The limitation

Separation cannot recover information that was destroyed in the sum. Two sources that produced identical energy in the same bin are not separable even in principle; the output is a plausible reconstruction, and where the model is unsure it invents something reasonable rather than reporting uncertainty.

The rights position deserves stating plainly as well. Separating a commercial recording produces a derivative of a copyrighted work, and the fact that a tool makes it easy says nothing about whether the use is permitted. Private study, transcription and practice sit differently from distribution and monetisation, the rules vary by country, and none of it is settled by the existence of the tool. Check what applies where you are before publishing anything derived this way.

BEFORE YOU MOVE ON

A separated vocal sounds acceptable inside a new arrangement and unpleasantly watery when exposed on its own in a quiet passage. What is the mechanism behind the watery quality?

A. The separator works in the time domain, so it introduces phase rotation in the vocal that is masked by other instruments and revealed when they are removed.

B. The mask estimate changes slightly from frame to frame, so the residual error moves about — the same family of problem as musical noise in denoising.

C. The vocal's reverb tail has been assigned to the accompaniment stem, and the missing tail is what makes the exposed vocal sound unnatural.

D. The separator's output is reconstructed from a compressed latent representation rather than from the original spectrogram, so it is a resynthesis of the vocal rather than an extraction of it, and every resynthesis carries codec artefacts that only become audible without masking content around them.

69 · 11 min

Generated music, and three separate legal questions

THE ONE THING TO KEEP

Whether training was lawful, whether an output infringes, and whether an output is owned are three different questions with different answers in different jurisdictions — and the practical response is records, terms, indemnities and checking the output, because the law is genuinely unsettled rather than merely complicated.

How a music generator works, briefly

Modern music generators do not synthesise waveforms directly. They work in a compressed representation: a neural audio codec encodes sound into a stream of discrete tokens at a manageable rate, a large model is trained to predict those tokens — from a text description, from an audio prompt, from both — and a decoder turns the predicted tokens back into audio. Diffusion-based systems do something analogous in a spectrogram or latent space.

The consequences you can hear follow from this. The model has learned the statistics of a very large corpus, so it produces things that sound like plausible music in a described style. It has no representation of song structure beyond what fits in its context, so long outputs wander. It has no notion of a performer's intention, so a solo is a stylistically appropriate sequence rather than an argument. And it reproduces the biases of its corpus: styles that are heavily represented come out well, and styles that are not come out as approximations of something else.

Three separate legal questions

Public discussion runs them together. They have different answers, different jurisdictions and different degrees of settlement.

1. Was training on copyrighted material lawful?

This is actively contested. In the United States, several suits involving music and audio generators are working through the courts, with fair use as the central argument, and the outcomes so far do not form a single coherent rule. In the European Union, the Digital Single Market Directive provides text-and-data-mining exceptions, with Article 4 allowing rightsholders to reserve their rights — which makes the question turn on whether an opt-out was expressed and how. The United Kingdom has consulted repeatedly on the question and has not settled it. Japan's exception is comparatively broad. Other jurisdictions differ again.

Anybody who tells you this question is resolved is describing their preferred outcome.

2. Does the output infringe?

Separate from training. A generated piece that closely reproduces a recognisable melody, hook or arrangement raises the same substantial-similarity question any other piece would. Models can and do memorise, particularly for material that appeared many times in training. This risk is about the specific output, not about the technology.

3. Who owns the output, if anyone?

The United States Copyright Office has stated that material generated by a machine without sufficient human authorship is not registrable, while works combining human authorship with machine-generated elements may be protected as to the human contribution. Several other jurisdictions have reached similar positions. The United Kingdom is unusual in having a statutory provision for computer-generated works with no human author, giving a 50-year term — a provision that has been questioned and consulted on.

So in some places a purely generated track may have no copyright owner at all, meaning you may have nothing to stop anybody else using it.

What this means in practice

You cannot resolve the law and you can manage your exposure.

- **Read the service's terms.** They state what rights you are granted in the output, whether commercial use is permitted, and on what tier. They vary substantially and they change.
- **Look for an indemnity**, and read what it excludes. Some providers indemnify business customers against copyright claims arising from outputs, with conditions.
- **Keep records.** Which tool, which version, which prompt, which date, what you did to the output afterwards. If a question arises in two years, contemporaneous records are the difference between a defensible position and a guess.
- **Check the output.** For anything prominent, search for the melody. Tools exist for audio matching, and a hook that feels familiar often is.
- **Do not assume the output is yours to license to others.** Delivering a generated track to a client with a warranty of ownership may be a warranty you cannot honour.
- **Expect platform enforcement to differ from the law.** Content ID and its equivalents are matching systems operated by private companies under their own rules; they can flag your generated track because it resembles something in their database, and being legally in the clear does not stop that.

The disclosure question

Separate from copyright, a growing set of rules concerns telling people that something is synthetic. The European Union's AI Act contains transparency obligations for certain synthetic content; several jurisdictions have or are considering rules aimed at synthetic media in elections and advertising. These are new, phased, and not uniform.

Two honest positions coexist and neither is wrong: that generated background music is a production tool no more requiring disclosure than a synthesiser, and that audiences have an interest in knowing. Where a rule applies, follow it. Where none does, decide deliberately rather than by default.

The free path

Open-weight music generation exists — **MusicGen** and **AudioCraft** from Meta, and **Stable Audio Open**, among others — and their licences differ, with some explicitly restricting commercial use. Read each one.

The alternative free path is not generation at all: **Free Music Archive**, **ccMixter**, and Creative Commons repositories contain large quantities of human-made music with clear licences, and a clear licence from a named person is a much simpler legal position than a generated track of uncertain provenance. The next lesson covers reading those licences properly.

The limitation

This lesson describes a moving position and will date. The cases in progress will produce rulings, legislatures will act, and the picture in two years will differ from the one here.

Nothing above is legal advice, and it cannot be: the answer depends on your jurisdiction, your use, your contract and facts nobody in a course knows. What is offered is the shape of the disagreement, so that you can recognise which question you are actually asking and know that a confident answer from any single source — including a vendor's marketing — is a claim rather than a settlement.

BEFORE YOU MOVE ON

A studio delivers a generated music bed to a client with a standard warranty that it owns the work and can licence it. Why is this warranty risky even if the generation service's terms grant broad usage rights?

A. Generative services retain a perpetual licence to outputs in their terms, so ownership can never transfer to a client under any tier.
B. Content ID will claim the track regardless of its origin, so the client cannot use it commercially whatever the warranty says.
C. In several jurisdictions a purely machine-generated work may have no copyright owner at all, so there may be nothing to own and nothing to stop a third party using it.
D. A warranty of ownership is only enforceable where the work has been registered with a copyright office, and generated works cannot be registered in most jurisdictions, so the warranty is void from the outset rather than merely difficult to honour.

70 · 11 min

What a music licence actually covers

THE ONE THING TO KEEP

A recording needs both a sync licence for the composition and a master licence for the performance, royalty-free means no recurring royalty rather than no conditions, and a Creative Commons attribution requirement is a legal condition whose omission leaves you unlicensed.

Two rights, and people usually buy one

Putting recorded music into a video requires permission for two distinct things.

The composition — the underlying song, owned by the writers and their publishers. Permission to combine it with moving images is a **synchronisation licence**.

The recording — the specific performance, owned by whoever made it, usually a label. Permission to use that recording is a **master use licence**.

Both are required. Buying one does not give you the other, and this is why you can legally record your own version of a famous song under a mechanical licence and still not be able to use it in a video without a sync licence, and why you can never use the famous recording without the label's agreement.

Library and stock music simplifies this by having both rights in one place, which is most of what you are paying for.

"Royalty-free" does not mean free

It means **no recurring per-use royalty**. You pay once and do not pay again per broadcast, per stream or per copy.

It does not mean no cost, no conditions, or no limits. A royalty-free licence still specifies:

- **Territory.** Worldwide, or one country.
- **Term.** Perpetual, or five years, after which the video must come down or be relicensed.
- **Media.** Online video is often included; **broadcast television, cinema and paid advertising are commonly excluded** or priced separately. This is the exclusion that catches people out, because a project that starts as a web video becomes an advert.
- **Number of productions.** One track, one project, unless the licence says otherwise. A subscription that lets you download unlimited tracks usually still licenses each *use*, and often ties the licence to an active subscription.
- **Exclusivity.** Almost never. Your competitor can use the same track.

Read the licence for the specific track. Libraries have several tiers and the tier decides the terms.

Creative Commons, read properly

CC licences are genuinely free and genuinely conditional. The condition is in the name.

- **CC BY** — use it, including commercially, if you attribute as specified. The attribution is a legal condition, not a courtesy: title, creator, source, licence, and any modifications. Omitting it means you are using it without a licence.
- **CC BY-SA** — additionally, derivatives must carry the same licence. For a video containing the music, whether this reaches your whole video is a question that has been argued at length and is not uniformly settled.
- **CC BY-NC** — no commercial use. A monetised video is generally commercial. A video on a business's channel is generally commercial. If there is any money anywhere near it, assume NC excludes you.
- **CC BY-ND** — no derivatives. Synchronising to picture is commonly treated as creating a derivative, so ND is usually unsuitable for video.
- **CC0** — dedicated to the public domain where that is possible. The closest thing to unconditional.

Two practical cautions. **Check the licence at the source**, on the creator's own page, not on an aggregator that may have copied it wrongly. And a CC licence can only be granted by the rights holder — an uploader who did not own the track cannot licence it to you, and your reliance on their claim is not a defence.

Public domain is narrower than it looks

A composition falls into the public domain after its term expires, and the term varies by country — commonly the author's life plus 70 years, with exceptions.

The **recording** is separate and has its own term. A 1950 recording of a Beethoven symphony is a recent recording of a public-domain composition: the composition is free, the recording is not.

So "it is Beethoven" is not a licence. Record it yourself, or find a recording explicitly released under CC0 or an equivalent — several projects exist precisely

to provide these.

Content ID is a matching system, not a ruling

Automated identification on video platforms compares your audio against a database of reference files. A match produces a claim. This is a private matching process operated under a platform's own rules, and it is not an adjudication of whether you have the right to use the music.

Two consequences that surprise people:

- **You can be claimed on music you licensed properly**, because the library's own distribution put the track in the reference database. Libraries usually provide a clearance process; it takes time and the claim sits on your video meanwhile.
- **You can be claimed on music you wrote and performed yourself**, if a version of it or something similar is in the database.

Disputes go through the platform's process. Keep your licence documents and receipts where you can find them quickly, because that is what the process asks for.

The routine that prevents problems

For every piece of music in a project, record in one place: the track, the source, the licence type, the licence number or the URL of the CC deed, the date, the permitted territory and term, and the attribution text if one is required. A spreadsheet is enough.

It takes a minute per track and it is the difference between answering a claim in ten minutes and spending a day reconstructing where a file came from.

The free path

- **Free Music Archive**, **ccMixter**, **Wikimedia Commons** — CC-licensed music, with the licence stated per track. Check it at the source.
- **Musopen** — public-domain and freely licensed classical recordings, which solves the recording-term problem above.

- **YouTube Audio Library** — free for use on that platform under its own terms; read them, because they are platform-specific.
- **Make it yourself. LMMS**, **Ardour**, **Hydrogen**, **Surge XT** and a large body of free sample packs will produce a serviceable bed, and a track you made has no licensing question at all.

The limitation

None of this is legal advice and the rules genuinely differ between countries: term lengths, what counts as a derivative, what exceptions exist for education, criticism or parody, and how moral rights operate. A position that is safe in one jurisdiction can be wrong in another, and material published online reaches all of them.

The defensible practice is the boring one. Use material whose licence you have read, keep the record, and where the stakes are real — an advertisement, a broadcast, anything with a budget — get the clearance properly rather than relying on a summary written by somebody who does not know your jurisdiction.

BEFORE YOU MOVE ON

A producer wants to use a famous 1965 recording of a song whose composer died in 1940 and whose composition is now out of copyright in their country. What is their position?

A. They are clear, because the composition is in the public domain and a recording cannot be protected independently of the work it records.

B. They need only a sync licence, since the master right expired with the composition's term and the label's interest lapsed at the same moment.

C. They need a mechanical licence to reproduce the composition, which the collecting society will grant at a statutory rate without the label's involvement.

D. The composition is free to use but the 1965 recording has its own separate term, so they need the label's permission or a different recording.

71 · 10 min

Frames that were never photographed

THE ONE THING TO KEEP

Interpolation assumes every pixel has a correspondence in the neighbouring frame, so it tears at occlusion boundaries and fragments thin structures — and an upscaler does not reveal detail but fabricates it, which makes upscaled output unusable as a record of anything.

Frames that were never photographed

Three related operations invent picture information that did not exist: **interpolation** makes new frames between real ones, **upscaling** makes new pixels within a frame, and **video generation** makes everything. All three are estimation, and all three fail in ways worth being able to name before you rely on them.

Interpolation, and where it breaks

Classical frame interpolation estimates **optical flow** — a vector per pixel describing where it moved between two frames — and warps the two frames toward an intermediate time. Learned methods such as RIFE and FILM replace the hand-built flow estimator with a network, and generally produce better results on difficult motion.

Both fail in the same four places, and all four follow from the same cause: flow assumes every pixel in one frame has a corresponding pixel in the other.

- **Occlusion boundaries.** Where something appears from behind something else, the newly revealed pixels have no correspondence in the earlier frame. The estimator has to invent them, and the result is a smearing or tearing around the edges of moving objects. This is the most common visible artefact.

- **Thin structures.** A railing, a wire, a spoke, a strand of hair. A thin object can move further than its own width between frames, so there is no overlap to match, and it breaks into fragments or disappears.
- **Repetitive texture.** A picket fence, brickwork, a striped shirt. Many candidate matches score equally well, and the flow field picks inconsistently across the region, producing a rippling wobble.
- **Large motion and blur.** Fast pans and heavily blurred frames give the estimator nothing sharp to correspond.

The useful rule: interpolation works well on smooth, moderate motion of large solid objects, and badly on everything else. For genuine slow motion, shoot at the higher rate — the frame-rate lesson makes the same point and it is the same reason.

Upscaling invents plausible detail

A super-resolution model has learned what high-resolution images look like, and given a low-resolution input it produces a high-resolution image consistent with it. Consistent is not the same as correct.

The consequence needs stating plainly: **an upscaler does not reveal detail, it fabricates detail**. If a number plate is six pixels across, the information identifying its characters is not in the file. An upscaler will produce sharp, legible characters, and they will be a plausible guess. Any use that treats upscaled output as evidence — forensic, journalistic, identifying a person — is treating an invention as a measurement.

The characteristic artefacts:

- **Hallucinated text.** Small illegible lettering becomes sharp nonsense, or sharp wrong words.
- **Faces.** Models trained heavily on faces reconstruct a face, which can be a different person's face. This has real consequences and has been demonstrated repeatedly.
- **Over-sharpened texture.** Skin becomes plastic, foliage becomes a repeating pattern, film grain becomes a crawling texture.

- **Temporal instability on video.** A per-frame model reconstructs slightly differently each frame, so the invented detail shimmers. Video-specific models reduce this and do not eliminate it.

Where upscaling is legitimately useful: restoring archive material for viewing, enlarging for a delivery format, cleaning a compressed source. Where it is not: anything where the detail is the point.

Generated video

Text-to-video and image-to-video models produce clips of a few seconds. The state of the art moves quickly and the structural limitations have been persistent:

- **Temporal consistency.** Objects drift, change colour, and lose or gain parts across a clip. There is no persistent representation of a scene.
- **Physics.** Contact, weight and momentum are approximated from appearance rather than modelled. Things pass through each other and liquids behave oddly.
- **Hands, text and counting.** All three remain unreliable, for related reasons: they require a structural constraint the model has learned only statistically.
- **Duration.** Coherence degrades with length, which is why outputs are short and why longer pieces are assembled from shots.

Treated as a source of short, controllable elements inside an edit — a texture, a background plate, an abstract transition — this is genuinely useful. Treated as a replacement for shooting a sequence, it is not there.

Judging any of the three

Play it at full speed, and play it twice. All three failure classes are temporal: a frame-by-frame inspection shows nothing, and the wobble, the shimmer and the drift only exist in motion.

Then look at the specific places named above — occlusion edges, thin objects, text, hands, repeated texture — rather than at the picture in general. The arte-

facts are predictable, so you can check for them directly.

The free path

- **RIFE** and **FILM** — open-source interpolation, run locally, both available as command-line tools and as plugins for free editors.
- **ffmpeg**'s `minterpolate` filter — classical motion-compensated interpolation, free and built in, lower quality than the learned methods but present everywhere.
- **Real-ESRGAN**, **BasicVSR++**, **waifu2x** — open-source upscalers, including video-specific ones.
- **DaVinci Resolve**'s free edition has optical-flow retiming and a super-scale function.
- Open-weight video generation models exist and change too fast for a list here to be useful; check their licences, which frequently restrict commercial use.

The limitation

All three operations share one property: they produce confident output with no signal about how much of it is invented. There is no per-pixel uncertainty map in the file, no warning that this region was extrapolated, no distinction in the output between a pixel derived from the source and one produced from the prior.

That is manageable for entertainment and dangerous everywhere else. The discipline that follows is to keep the original, to say in any handover that the material has been interpolated or upscaled, and never to let a generated frame be used as a record of something that happened.

BEFORE YOU MOVE ON

Why does frame interpolation produce its most visible artefacts around the edges of moving objects rather than in their interiors?

A. Edges contain the highest spatial frequencies, and the interpolator's warping operation attenuates high frequencies more than low ones.

B. Where something is revealed from behind something else, the newly visible pixels have no counterpart in the earlier frame, so there is nothing to warp and the model has to invent them.

C. Optical flow is estimated on a coarse grid and refined, and the refinement stage cannot resolve motion boundaries finer than the coarsest grid cell.

D. Objects near an edge are usually at a different depth from the background, and any depth discontinuity means the two regions move by different amounts, which violates the smoothness assumption that flow estimation relies on and causes the vectors in that neighbourhood to be averaged into a single incorrect value.

72 · 10 min

Provenance, watermarks and what to tell people

THE ONE THING TO KEEP

Provenance is a signed record of what cooperating tools reported doing and can be stripped by an ordinary re-encode, while detection is an inference that always trails the generators — so an absent credential or an undetected watermark is evidence of nothing, and disclosure is a separate question from either.

Two different problems, often confused

Provenance is a record of where a file came from and what was done to it. **Detection** is working out after the fact whether something was synthesised.

They are different approaches with different reliability, and the discussion mixes them constantly.

Provenance is a chain of claims made by cooperating tools. Detection is an inference made about an uncooperative artefact. Provenance can be strong and can be stripped; detection is weak and is always catching up.

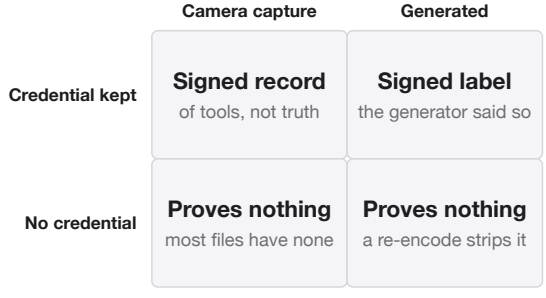
Content Credentials and C2PA

The **Coalition for Content Provenance and Authenticity** publishes a specification for attaching a signed manifest to a media file. The manifest can record what captured or created the file, what edits were applied, which tools made them, and — where the creator chooses — who they are. Each step is cryptographically signed, so tampering with the record is detectable.

Adoption is real and growing: several camera manufacturers, several editing applications, and several generative services now write Content Credentials. The specification is open and implementations exist.

Three limits are important and are not obscure:

What a content credential proves, and what it does not



A manifest attests to what cooperating tools reported doing, signed so that tampering is detectable. It does not attest that the content is true — a camera can faithfully photograph a staged scene and sign it. And because the manifest is metadata, a screenshot or an ordinary transcode removes it, so an absent credential is evidence of nothing at all.

Stripping. The manifest is metadata. Screenshot the image, re-encode the video, or upload it to a platform that rewrites files, and it is gone. Absence of a credential means nothing at all — most files in the world have none.

Scope. A signed manifest attests to what tools reported doing. It does not attest that the content is true. A camera can faithfully photograph a staged scene and sign it.

Trust. The signature is only as meaningful as the certificate behind it and your reason to trust the signer.

Durable content credentials — combining the manifest with an invisible watermark and a fingerprint lookup, so the record can be recovered after stripping — are being developed to address the first limit. They are a mitigation, not a fix.

Watermarking

An invisible watermark embeds a signal in the content itself: imperceptible perturbations in pixels, or in audio, that a detector can recognise. Google's SynthID and several others work on this principle, and some are now applied by default to generated output.

Published robustness figures describe survival through ordinary transformations — compression, cropping, resizing, re-recording. Those are the right tests for accidental loss. They are not tests against somebody actively trying to remove the mark, and academic work has repeatedly shown that adversarial removal is possible against most schemes.

So the honest statement is the same as for cloned voices: a detected watermark is meaningful evidence that content came from a particular generator; **an undetected watermark is not evidence of anything.**

Detection classifiers

Models trained to distinguish generated from captured media perform well on the generators in their training set and degrade sharply on new ones. They also produce false positives, and a false accusation of fabrication is a serious harm in its own right.

The practical position for anybody who has to decide whether something is real: a detector's output is one weak signal among several, alongside where

the file came from, who is circulating it, whether there is corroborating material, and whether the content is consistent with things that can be checked independently. Journalism's existing verification methods have not been superseded by a classifier.

Disclosure, which is a separate question

Whether to tell your audience is not settled by any of the above.

Some of it is law. The European Union's AI Act includes transparency obligations for certain kinds of synthetic content, with phased application. Various jurisdictions have introduced or proposed rules targeting synthetic media in elections and in advertising. Several platforms require creators to label realistic synthetic content and apply their own labels. These differ by place, by sector and by how realistic the content is, and they are changing.

Where a rule applies, follow it and check the current text rather than a summary.

Where none applies, the question is editorial and is genuinely contested. One defensible position: a generative tool used to make background music, clean up noise or interpolate frames is a production tool, and disclosing it is as unnecessary as disclosing which EQ you used. Another defensible position: audiences have an interest in knowing when a voice or a face is synthetic, because those specifically carry an implicit claim that a person did something.

A workable line that most people find they can defend: **disclose when the synthesis is about a person or an event.** A synthesised voice, a face, a depiction of something that is presented as having happened. Do not clutter every credit with the tools used for ordinary production work.

What to do on your own projects

- **Keep the originals.** The unprocessed capture, in a separate place, with the date. This is the most useful provenance record you will ever have and it costs storage.
- **Keep a log.** Which tools, which versions, which prompts, which models, for anything generated. A text file in the project folder.

- **Preserve credentials where they exist**, and know that your export settings may be discarding them.
- **Get consent in writing** for anything involving a real person's voice or likeness, covering the scope of use — the previous lesson on cloned voices covers what that scope has to include.
- **Say so in the credits** when a voice or a face is synthetic, whether or not you are required to.

The free path

The C2PA specification is open, and the **c2patool** command-line utility is free and open source: it reads and writes manifests, which is enough to inspect what a file carries and to attach your own. **ExifTool** reads metadata generally. **ffmpeg** is what strips it by default, so if you care about preserving credentials through a transcode, check what survives — usually it does not.

The limitation

None of this establishes truth. A perfectly signed provenance chain on a file that faithfully records a staged event is a trustworthy record of an untrustworthy thing. A stripped file with no credentials may be entirely genuine. Watermarks are removable by anyone motivated, detectors are behind the generators, and every technical measure here helps most against casual misuse and least against a determined actor.

The field is also moving faster than any description of it. Standards, platform rules and legislation will all have changed by the time this matters to you, and the only durable advice is to check the current position rather than a remembered one — and to keep your own records, which is the part that does not depend on anybody else's system continuing to work.

BEFORE YOU MOVE ON

A newsroom receives a video with no Content Credentials and no detectable watermark. A detection classifier scores it as probably genuine. What can be concluded?

A. Very little: absence of a credential is the normal state of most files, watermark absence proves nothing, and the classifier is one weak signal among several.

B. That the file is probably genuine, since three independent indicators point the same way and their combination is stronger than any one of them.

C. That the file has been re-encoded at some point, because a genuine capture from a modern device would carry credentials through to delivery.

D. That the file is unlikely to have come from a major generative service, because those apply watermarks by default and a watermark that survives ordinary compression would still be detectable, so the absence narrows the origin to either a genuine capture or an open-weight model run locally.

CASE STUDY · MODULE 8

Forty more lessons in a voice that had left

A district institute of education in Kozhikode, halfway through a sixty-part narrated science series in Malayalam.

Twenty of the sixty lessons had been narrated by Ms Beena, a teacher on deputation, in four hours and twenty minutes of clean, close-miked recordings. Then her deputation ended and she was posted three hundred kilometres away.

Three ways to finish. A new narrator, which changes the voice in the middle of a series that children would work through in order. Bringing Beena back for four Saturdays, at an honorarium of eight hundred rupees a day plus travel she would have to arrange herself. Or cloning her voice from the material already recorded.

The technical assistant built a test clone and played it to three colleagues who knew her. At three seconds all three said it was her. At thirty seconds all three said something was wrong and none could say what — the timbre was right and the timing and stress were somebody else's, the generic behaviour of the synthesiser's training distribution. On one line with a contrast in it, the stress landed on the wrong word, in a completely convincing voice.

The director insisted the decision be taken as three separate questions rather than one.

The first was consent. Beena said yes on the phone in under a minute. The director asked for it in writing and scoped: for which project, for which uses, for how long, and whether the trained model itself would be kept, where, and who could run it. Beena agreed to the remaining forty lessons of this series, for three years, with the model held on the institute's own machine rather than a service and deleted at the end. She declined a clause offering the voice for future departmental material — the clause she said afterwards she would never have noticed in a one-line permission.

The second was quality. The assistant built a pronunciation lexicon of about sixty recurring terms — element names, three place names, the word for pre-

cipitate — which removed that category of error permanently. He then marked stress by hand in any sentence containing a contrast, because the written text does not encode stress and stress is the meaning. And every lesson would be listened to end to end against the script before publication, about forty minutes a lesson: twenty-seven hours of listening across the series, budgeted in advance rather than discovered.

The third was disclosure. India's amended IT rules for synthetically generated information prescribe a label a person can perceive, and for audio one in the opening portion. They recorded a single line at the head of each lesson — in Beena's own real voice, cut from the existing material — saying that the narration that follows is a synthesised version of her voice, used with her permission.

That line was written before a single lesson was generated, which the assistant now says was the only reason it exists. Written first it is one sentence. Written after forty lessons have been rendered and timed against slides, it is a change to forty locked timelines, and it does not get made.

The costs were not evenly distributed, and the institute counted them honestly afterwards. Bringing Beena back for four Saturdays would have been about six thousand rupees in honorarium and perhaps three weeks of calendar time waiting on her availability, with no consent question at all and no listening pass beyond the ordinary one. The clone cost five weeks of an assistant, twenty-seven hours of listening, a written agreement somebody had to draft, and a deletion to remember eighteen months later. What it bought was a series that sounds like one series, finished before the exam term, in the voice the first twenty lessons had already established.

STOP HERE

Before turning the page: what would you have done, and what would it have cost if you were wrong? Write it down. The point of the next section is to compare.

WHAT HAPPENED

One assistant finished the forty lessons in five weeks. The lexicon removed the mispronunciations outright. Eleven of the forty went back for a re-render after the listening pass, and every one of the eleven was a stress error on a contrast rather than a wrong word — exactly the failure the written text cannot prevent, caught only because somebody listened to the whole thing. The disclosure line, which the director had expected to generate complaints, produced two emails, both asking how it had been done. The model was deleted eighteen months later when the series was reissued, and the deletion was minuted. Asked afterwards what had mattered, Beena did not talk about the cloning. She said she had been ready to say yes to "use my voice for this", and it was the scoping — the term, and the question of whether a model of her voice would still exist next year — that she would not have thought to ask about herself.

The clone convinced colleagues for three seconds and not for thirty. What is the mechanism?

A speaker encoder produces an embedding that captures timbre — the resonances of a particular vocal tract and its average pitch — which is most of what makes a voice recognisable in a single word. It does not capture idiolect: the person's characteristic pauses, phrasing, filler words and which syllable they lean on. So a clone is the right voice performing somebody else's habits, and the longer it speaks the more of the behaviour you hear.

Why did the director insist consent be scoped rather than simply given?

"You can use my voice for this" is not permission to hold a trained model of a person's voice on a machine for two years, and the gap between those two things is where the real disputes sit. Scoping means naming the project, the permitted uses, the term, whether the model is retained, where it lives and who may run it. It matters more than usual for a voice, because a convincing clone is also a fraud tool: documented cases turn on a few seconds being enough to convince somebody who is not expecting it.

Why was listening to all forty lessons non-negotiable, given the voice sounded right?

A synthesiser has no confidence score for prosody. It will read a sentence with exactly the wrong emphasis in a perfectly natural voice and give no signal that any-

thing is wrong, and the errors are local, so a spot check misses them. That is precisely what happened: eleven lessons had a stress error on a contrast, and only an end-to-end listen against the script would have found them.

MODULE 8 · TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record: answers and the reason behind each are at the back. If two questions from the same module go wrong, re-read that module before the exam rather than after.

1 A synthesiser reads the sentence with the stress on the word said, changing its meaning. What has gone wrong?

A. The vocoder has mispredicted the waveform for that word
B. The sample rate is too low to carry the stressed syllable
C. The pronunciation lexicon is missing an entry for that particular word
D. Written text does not encode stress, so the model guesses it

2 An automatic transcript contains a fluent sentence nobody said, over a passage of silence. What is this?

A. A substitution error, counted normally in the word error rate for the file
B. A generative recogniser producing plausible text from non-speech audio
C. A sample-rate mismatch between the audio and the model
D. The result of transcribing with the wrong language selected

3 Why can a separator not cleanly split a vocal and a string pad in the same register?

A. Masks must sum to one, so a bin can only belong to a single source
B. The mixture contains only their sum, so the split is a learned guess
C. Spectrograms cannot represent two sources inside the same frequency bin
D. Demucs works on the waveform and ignores overlapping frequencies

4 A number plate six pixels wide is upscaled and the characters come out sharp. What have you got?

A. The original characters, recovered by the model's prior
B. A lossless enlargement, since super-resolution is reversible
C. The characters at lower confidence, flagged in the output file
D. A plausible guess, consistent with the input but not evidence

5 A colleague says yes, use my voice for this video. What has not been agreed?

A. Nothing further — spoken permission covers the recorded use
B. Whether the audio may be normalised to the platform's loudness target
C. Whether the trained model is kept afterwards, and for how long
D. Whether the video may be published outside working hours

6 A widely shared video carries no content credentials. What does that tell you?

A. Nothing — most files have none, and a re-encode strips the rest
B. That it was generated, since generators do not sign their output
C. That it was edited after capture and the chain was broken
D. That the signing certificate expired before it was published

EXAMINATION

Motion, Sound and Music — final examination

This paper covers the whole course: reading timing and spacing, motion that carries meaning in an interface, choosing and shipping a delivery format, compositing two images so the seam does not show, codecs and colour and delivery, capturing a voice in an ordinary room, shaping and mixing it, and using generative and analytic models honestly. Every question asks about a mechanism you have been taught rather than a definition you could guess. You will be given twenty-five questions drawn at random from a larger pool, so no two attempts are the same paper. The pass mark is seventy per cent. There is no timer: read at whatever pace you read, in whatever language you think in, and go back and change an answer if you want to. If you do not pass, you may take it again after thirty minutes with a fresh set of questions, as many times as you need. Passing opens the next course in the track, and a teacher can open that course for you at any time regardless of this paper.

On the website a sitting is 25 questions drawn from this pool and the pass mark is 70%. There is no time limit, there and here. Passing opens the next course in the track.

1 `1. How motion reads as real`

Which of these is genuinely better with linear interpolation?

A. A loading spinner rotating continuously

B. A modal appearing after a tap

C. A card settling into a list

D. A drawer sliding open from the screen edge

2 1. How motion reads as real

A 200-pixel move reads well at 250 ms. What duration keeps the perceived speed for an 800-pixel move?

A. 1000 ms, in proportion to the distance
B. 250 ms, because duration should not follow distance
C. About 500 ms, since duration scales sub-linearly
D. About 350 ms, because exits run faster than entrances

3 1. How motion reads as real

The one-key ease sets about a third of influence on both sides. Why does the move still feel mushy?

A. Symmetric easing is only valid on rotation, scale and opacity changes
B. Both ends are slow, so all the travel is crammed into the middle
C. The influence value is too high for a short duration
D. The spatial interpolation has defaulted to bezier

4 1. How motion reads as real

A spring has a damping ratio of exactly one. How does it behave?

A. It oscillates three or four times before settling
B. It creeps in slowly and never reaches the target
C. It overshoots once and settles in about half a second
D. It arrives as fast as possible with no overshoot

5 1. How motion reads as real

Elements moving in two dimensions arc and elements moving in one do not. Why should a horizontal drawer not arc?

A. Because arcs are only visible on moves longer than 600 pixels
B. Because a curve would trigger a layout recalculation
C. Because bowing upward always reads as a heavy object
D. Because an arc implies a freedom the drawer does not have

6 1. How motion reads as real

Which principle is most damaged by the fact that an interface is watched hundreds of times?

A. Timing, the primary carrier of weight
B. Exaggeration
C. Staging, which controls where the eye goes
D. Arcs, still true and still ignored

7 1. How motion reads as real

A small animated badge sits in a corner while a large static headline is meant to be read. What happens?

A. The badge wins, because onset of motion beats everything else
B. The headline wins, because text captures attention once fixated
C. Both are read, because the fovea covers about ten degrees
D. Neither, because peripheral vision cannot detect small motion

8 2. Motion that carries meaning

A row slides out and the rows below close the gap. Which job is that motion doing?

A. Saying the system is alive
B. Saying where the thing came from
C. Saying what just happened
D. Nothing; it is decoration

9 2. Motion that carries meaning

Why does FLIP invert with a transform rather than animating top and left?

A. Because top and left cannot be animated in modern browsers
B. Because getBoundingClientRect only reports transformed positions
C. Because transform can be animated without re-running layout
D. Because the inverted position must be measured after the paint

10 2. Motion that carries meaning

A 14-pixel caption goes blurry and oversized mid-flight in a container transform. What is the standard fix?

A. Counter-scale the child by the inverse of the parent's scale
B. Cross-fade the two captions at the midpoint of the move
C. Animate the container's width and height independently
D. Increase the caption's font size before the transition starts

11 2. Motion that carries meaning

A list entered top-first. How should it leave?

A. Top-first again, so the two match
B. All at once, since exits should be instant
C. In a random order, which reads as organic
D. Bottom-first, so the exit is not a replay

12 2. Motion that carries meaning

A three-second lower-third has a 1.2-second entrance and a 0.8-second exit. What is wrong?

A. The exit is too slow relative to the entrance
B. One second of still text is not enough to read a name
C. Three seconds is longer than any lower-third should be
D. The entrance should be shorter than the exit, not longer

13 2. Motion that carries meaning

Why name a motion token by its role rather than by its value?

A. Because CSS custom properties cannot contain digits
B. Because the value will change and the name would become a lie
C. Because role names compile to smaller CSS output
D. Because durations must be expressed in seconds, not milliseconds

14 2. Motion that carries meaning

Most web requests have unknown duration. Which indicator is honest, and why?

A. A determinate bar, because users prefer a percentage
B. A determinate bar that decelerates toward the end
C. A skeleton screen, because it always matches the layout that will arrive
D. An indeterminate indicator, which claims only that work is happening

15 3. Shipping the motion

A transparent video has to play in Safari and in Chrome. What does that require today?

A. A single WebM file with VP9 alpha, which both browsers now support
B. An MP4 with a premultiplied alpha channel
C. Two files: HEVC with alpha for Safari, VP9 alpha for the rest
D. A luma matte encoded into the file's metadata

16 3. Shipping the motion

What does declaring a path length of 100 on an SVG path buy you?

A. All dash arithmetic pretends the path is 100 units long
B. It caps the number of points the browser will render
C. It normalises the stroke width across different viewBoxes
D. It makes the path animate at a constant speed along its length

17 3. Shipping the motion

A Lottie with 400 shapes stutters on a phone. What is the most useful change?

A. Switch from the SVG renderer to the canvas renderer
B. Re-export at 60 fps so the keyframes are denser
C. Compress the JSON with gzip before serving it
D. Convert the file to dotLottie to reduce its download size

18 3. Shipping the motion

If a GIF is unavoidable, what makes the largest difference to how it looks?

A. Exporting at a higher resolution so that dithering is less visible
B. Using error-diffusion dither, which compresses best
C. Generating the palette from the whole clip rather than one frame
D. Raising the frame rate so motion is smoother between frames

19 3. Shipping the motion

A developer cannot recolour an animation, pause it, or place it on a dark background. What does that tell you?

A. The animation needs a reduced-motion variant
B. The wrong file format was delivered
C. The animation is too long for the interaction
D. The layers inside the file were not named

20 3. Shipping the motion

Which of these can a screen recording of an animation never tell a developer?

A. How long the entrance takes
B. Which direction the element travels in from
C. Roughly how far the element travels
D. What happens when the user interrupts it

21 3. Shipping the motion

Why is a scroll-driven CSS animation preferable to a JavaScript scroll listener?

A. It fires fewer events, so the listener has to be called less often
B. It works in every browser, where listeners do not
C. It can animate layout properties that listeners cannot
D. It runs on the compositor, so it stays smooth while scripts run

22 4. Putting two images together

In the over operation, what is happening at an alpha of nought point four?

A. The foreground is hidden and only the background shows

B. A weighted mix, which is what a real soft edge looks like

C. The matte is broken, since alpha should be zero or one

D. The foreground is shown at full strength over the background

23 4. Putting two images together

Why gain a matte up by five or ten times before judging it?

A. To reveal areas that should be solid white and are 0.95

B. To sharpen the edge band before the despill stage

C. To convert it from premultiplied to straight alpha

D. To check that the keyer has not clipped any of the highlights

24 4. Putting two images together

A fire element filmed on black is screened over a bright sky and nearly disappears. Why?

A. Screen requires a premultiplied source to work correctly

B. The element's black background is being added rather than dropped

C. Screen can only lighten, and the sky is already near the top

D. Screen is base-dependent, so the layer order has been reversed

25 4. Putting two images together

What is a light wrap actually built from?

A. A blurred copy of the foreground, screened back over itself at the edges

B. A despilled copy of the subject, tinted to the new scene

C. A blurred copy of the background, restricted to the matte's edge band

D. The matte's inverse, eroded and added to the composite

26 4. Putting two images together

What is the fastest way to rotoscope a 200-frame shot?

A. Set the first and last frames, then halve the intervals

B. Work forward frame by frame from the first frame

C. Use one shape with forty points spread around the silhouette

D. Track the outline rather than the form beneath it

27 4. Putting two images together

What are machine segmentation models genuinely good for in a roto workflow?

A. Producing the final edge matte on hair and motion blur

B. Removing temporal instability from a hand-drawn matte

C. Replacing planar tracking on flat, deforming surfaces

D. Generating the garbage and core mattes in seconds

28 4. Putting two images together

A 1080p camera solve reports 2.4 pixels of reprojection error. What does that mean?

A. It is excellent, since anything under 5 pixels is sub-frame

B. It is marginal and the insert will slide; find the cause

C. The focal length was supplied, which always raises the figure

D. The solve is fine but the scene orientation has not been set

29 5. Codecs, colour and delivery

Why does a grainy source need roughly twice the bitrate of a clean one?

A. Grain raises the average luminance of every frame, which costs bits

B. Grain is unpredictable, so the residual after prediction is large

C. Grain forces the encoder to insert more keyframes

D. Grain defeats chroma subsampling and forces 4:4:4

30 5. Codecs, colour and delivery

Why does adaptive streaming want scene-cut keyframes off and a fixed interval?

A. Because scene-cut keyframes are visually worse than regular ones
B. Because scene detection adds a full extra encoding pass
C. Because variable intervals prevent the player from seeking at all
D. Because segments must begin on keyframes at predictable points

31 5. Codecs, colour and delivery

Why is two-pass meaningfully better than one pass at the same target bitrate?

A. The second pass encodes at a higher internal bit depth
B. Two passes halve the quantiser at every block
C. The first pass logs where the difficult material is
D. The first pass removes noise before the second one encodes

32 5. Codecs, colour and delivery

When is a hardware encoder the right choice over x264 at a slow preset?

A. When encode time is the constraint, as in live streaming
B. For anything that will be downloaded many times
C. When the source is 10-bit and needs a wider gamut
D. Whenever the file has to be smaller at the very same quality

33 5. Codecs, colour and delivery

Why does applying a look LUT to under-exposed log footage destroy detail permanently?

A. A LUT is a fixed table with a defined input domain, so values land wrong
B. Look LUTs are one-dimensional, so they cannot change any hue relationships
C. Log footage is always recorded in limited range
D. A LUT applies gamma twice when the input is log

34 5. Codecs, colour and delivery

What shutter speed gives the conventional look at 30 frames a second?

A. 1/30, matching the frame interval exactly
B. 1/125, to keep each frame sharp
C. 1/60, half the frame interval
D. 1/15, to double the motion blur

35 5. Codecs, colour and delivery

You need five-times slow motion. What gives the honest result?

A. Shoot at 24 and interpolate up to 120 with optical flow
B. Shoot at 120 and play at 24, using real frames
C. Shoot at 24 and conform the timeline to 120
D. Shoot at 60 and duplicate every other frame

36 6. Capturing sound

Why are lavalier microphones usually omnidirectional?

A. Omni capsules are cheaper to manufacture at that size
B. A cardioid would pick up the speaker's own chest and clothing reflections
C. Directional capsules cannot be made small enough to hide
D. They sit 20 cm from the mouth, where direct sound dominates anyway

37 6. Capturing sound

A neighbour's drill is audible in every take. What will acoustic foam do?

A. Reduce it by roughly ten decibels if the whole wall is covered in it
B. Reduce the low frequencies while leaving the high ones
C. Reduce it only if mounted with an air gap behind it
D. Nothing useful — foam absorbs reflections, it does not block sound

38 6. Capturing sound

Why does 5 cm of foam do almost nothing at 100 Hz?

A. Foam reflects low frequencies rather than absorbing them
B. A quarter wavelength at 100 Hz is about 86 cm
C. Low frequencies travel faster and pass through before absorption
D. Room modes at 100 Hz are standing waves, which foam cannot reach

39 6. Capturing sound

Why is recording at minus 18 dBFS average cheap in 24-bit and expensive in 16-bit?

A. 24-bit gives about 144 dB of range, more than any preamp delivers
B. 16-bit files cannot be normalised without adding dither
C. 24-bit recordings are automatically gain-compensated when imported
D. 16-bit sample rates are lower, so the noise floor is higher

40 6. Capturing sound

Why does heavy spectral noise reduction leave short tonal chirps?

A. The algorithm resamples the file and introduces aliasing
B. Consonants are broadband and are mistaken for tones
C. The noise is random, so subtracting its average leaves isolated residuals
D. The noise profile was taken from the wrong part of the recording altogether

41 6. Capturing sound

A 512-sample buffer at 48 kHz. What is the round-trip latency, roughly?

A. About 5 ms, which is fine for monitoring
B. About 43 ms, which is unusable for anything
C. About 21 ms, past where most speakers notice
D. About 11 ms, because latency is paid only on output

42 6. Capturing sound

Why record scratch audio on the camera when a separate recorder is running?

A. It is the sync reference for waveform matching, and the backup
B. It carries the timecode that the recorder cannot generate
C. Camera preamps are better than field recorder preamps
D. It prevents the two devices from drifting apart over a long take

43 7. Shaping and mixing sound

How do you find the frequency of a honk in a voice?

A. Cut a narrow band by 10 dB and sweep until it improves
B. Use a wide bell and listen for a change in overall tone
C. Compare the track against a reference inside a spectrum analyser
D. Boost a narrow band by 10 dB and sweep until it is unbearable

44 7. Shaping and mixing sound

Why cut with a narrow Q and boost with a wide one?

A. Narrow boosts are much more likely to clip the mix bus output stage
B. Problems are resonances; pleasantness is a shape across a region
C. Wide cuts cannot be made without phase distortion
D. A narrow boost is inaudible below about 500 Hz

45 7. Shaping and mixing sound

A voice sounds oddly soft and dull after compression. What is the most likely setting?

A. The release is far too long and is holding the gain reduction down
B. The attack is far too fast and is dulling the consonants
C. The ratio is too low to catch the loudest phrases
D. The threshold is above every phrase in the take

46 7. Shaping and mixing sound

What prevents a plosive, as opposed to reducing its aftermath?

A. A high-pass filter at 80 Hz applied across the recording chain
B. A pop filter held hard against the capsule
C. Lowering the preamp gain by six decibels
D. Angling the microphone 15 to 30 degrees off the mouth's axis

47 7. Shaping and mixing sound

Why put reverb on a send rather than as an insert?

A. Sends are processed before inserts in every mixer

B. An insert reverb cannot have its pre-delay or decay time adjusted

C. The dry signal is untouched and sources can share one space

D. Send reverbs use less CPU than insert reverbs do

48 7. Shaping and mixing sound

Why press the mono button before delivering a mix?

A. Phones and laptops are mono, where comb notches appear in full

B. Mono playback halves the file size on most platforms

C. Loudness meters can only measure a programme correctly when mono

D. Stereo dialogue moves when the camera cuts

49 7. Shaping and mixing sound

Two speakers in a conversation do not sound like one scene. What is the structural fix?

A. Balance them, then let a dialogue bus do the last decibel

B. Compress each speaker harder until their levels match

C. Put a limiter on the master to even out the whole programme

D. Add the same reverb as an insert on each speaker's own track

50 8. Machines that make sound and motion

A script contains a year and the synthesiser reads it as a quantity. What is the quickest reliable fix?

A. Increase the speaking rate so the digits run together

B. Split the sentence in two at the number

C. Write the year out in words in the input text

D. Add a 400-millisecond break before the number

51 8. Machines that make sound and motion

Why do clones of speakers with strong regional accents often come out softened?

A. Accent is carried by the vocoder, which is language-specific
B. The encoder was trained on a distribution, so distant voices drift
C. Regional accents contain frequencies above the model's output ceiling
D. The reference audio is usually too long for the encoder

52 8. Machines that make sound and motion

A transcript is 95 per cent accurate. Why is that worse than it sounds?

A. Word error rate excludes insertions from its total count
B. Accuracy is measured per character, not per word
C. The figure is averaged across speakers, not files
D. Errors cluster on names, numbers and technical terms

53 8. Machines that make sound and motion

Why is separating, fixing and recombining gentler than processing a whole mixture?

A. Separation is lossless, so nothing in the mixture is altered by the split
B. Stems are stored at a higher bit depth than the mixture
C. Masking removes reverb, which makes editing more precise
D. The masks partition the mixture, so untouched stems sum back unchanged

54 8. Machines that make sound and motion

Which of these is a question about the output rather than about training or ownership?

A. Whether a text-and-data-mining exception applied to the training
B. Whether the piece reproduces a recognisable hook
C. Whether a prompt alone makes you an author
D. Whether a rightsholder expressed a reservation

55 8. Machines that make sound and motion

What does royalty-free actually guarantee?

A. No recurring per-use payment; everything else is in the licence

B. That the track may be used in any medium, worldwide

C. That no attribution is required for commercial use

D. That both the composition and the recording have been cleared already

56 8. Machines that make sound and motion

You are claimed on music you licensed properly. What has happened?

A. The licence was void because the library ended that track's distribution

B. The platform has ruled that your licence does not cover video

C. A private matching system found the track in its reference database

D. The composition is cleared but the master licence is missing

Answers

Each entry gives the correct option, then what each wrong one reveals about how somebody was thinking. The second part is worth more than the first: a wrong answer here is a named misreading, not a mistake.

Lesson checks

1. Nothing in the world starts at full speed — C

A — Treats duration as the only variable, when the same duration reads fine elsewhere in the same product.

B — Blames rendering performance for a problem the curve creates even at a perfect frame rate.

D — Adds decoration at the end of the move when the problem is at the beginning of it.

2. Motion blur will not fix a floaty logo — A

B — Assumes weight is a function of how long a move takes rather than how its distance is distributed.

C — Treats a rendering setting as the source of perceived mass rather than a way of stopping fast motion strobing.

D — Confuses static depth cues with the temporal cues that make something feel heavy.

3. Frames, not seconds — A

B — Confuses a temporal sampling artefact with a compression artefact; the strobing appears in an uncompressed render too.

C — Notices the rounding, which is real, but 8ms of duration cannot account for visible strobing — the per-frame displacement can.

D — Treats easing as something sampled into existence rather than a continuous function evaluated at whatever frames exist; the curve is identical in both.

4. The panel most people never open — C

A — Invents a coupling between timing and path; easing redistributes time along a path and never changes the path's geometry.

B — Treats the speed graph as a different kind of animation rather than a different view of the same keyframes.

D — A real cause of unexpected movement, but it produces an arc around the anchor during rotation, not a curved path on a pure position move.

5. Why pressing Easy Ease still looks wrong — B

A — Invents a duration penalty; the card is in motion for the whole 400ms, just at varying speeds.

C — Overshoot adds weight but is not required for decisiveness; an asymmetric ease with no overshoot reads as committed.

D — A bezier with collinear handles produces constant velocity perfectly well; the issue is where the curve puts its fast section, not that it has one.

6. Springs, and what you give up for them — C

A — Conflates a rendering-performance issue with a state issue; the stick happens even when both run at a full 60fps.

B — Treats the problem as probabilistic; the stick occurs on every interruption regardless of how long the animation is.

D — Overshoot is achievable with keyframes and is not what continuity during interruption depends on.

7. Straight lines and the things that lag behind — D

A — Focuses on aggregate time rather than the per-interaction perception that produces the complaint; users are reacting to each tap, not to a daily total.

B — Attributes the complaint to a visual artefact; users described slowness, which is a timing perception, not an appearance one.

C — Inverts the rule: scenery is exactly where follow-through belongs, and the control is exactly where it does not.

8. The old principles, filtered for a screen seen four hundred times — B

A — Bundles two separate principles; anticipation is a movement away from the target and needs no deformation at all.

C — Says it is merely useless; the complaint is that it is felt as latency, which means it is visible enough to cost something.

D — Attributes the problem to the technique rather than the viewing contract; anticipation interpolates perfectly well and is used in motion graphics.

9. Where the eye already is — B

A — Applies a threshold about perceiving flicker to a problem about gaze position; the element is perfectly visible to anyone looking at it.

C — Treats it as an easing-duration problem, but lengthening the entrance only means more of it is missed while the eye is still travelling.

D — Inverts a real fact: the periphery is unusually good at detecting motion onset, which is what eventually draws the eye there.

10. The three jobs an animation can do — B

A — Assumes overlapping work means the shorter of the two is the wait; the user cannot read the content until the slower of the two finishes.

C — Believes animation and network work are serialised; the network stack runs independently of the compositor.

D — Treats a threshold as a budget that can be spent freely rather than as a ceiling on the total response.

11. How long an animation should take — C

A — Reaches for a performance explanation when the measurement has already confirmed the duration is correct.

B — Applies the sub-100ms instantaneity threshold to a large moving element, where it produces a flicker rather than a transition.

D — Correct that exits should be shorter, but the complaint here is about the entrance, and the rule is a refinement rather than a cause of perceived sluggishness.

12. The transition is the explanation — A

B — True for static positioning but irrelevant — the technique works on absolutely positioned elements too, and the reason is cost, not validity.

C — Confuses interpolation of colour with interpolation of geometry; colour spaces have nothing to do with position.

D — Misreads the measurement step: getBoundingClientRect reports the final rendered box regardless of which properties produced it.

13. Staggering without making people wait — B

A — Confuses delay with duration; a delay postpones the start and leaves the animation's own length unchanged.

C — Reaches for a rendering explanation; staggered delays on transform and opacity are compositor work and 24 of them are not a load problem.

D — Overstates a real guideline: top-to-bottom is correct for a list, and ordering does not create a duration complaint.

14. Why your lower-third looks like a template — B

A — Diagnoses legibility when the text was never held still long enough to be read at any size.

C — Reaches for contrast when the binding constraint is how long the words are available.

D — Spends more of a fixed budget on movement, which is the thing already crowding out the still time reading requires.

15. Naming the motion so it stops drifting — B

A — Invents a specificity rule; custom properties follow ordinary cascade and inheritance, and the benefit is coverage rather than priority.

C — Media queries are live rather than evaluated once, and layer creation is unrelated to how the duration was set.

D — Describes a real hazard of the JavaScript approach, but CSS overrides per component need no listener at all, so this is not the reason.

16. Spinners, skeletons and honest waiting — C

A — Animated transforms are explicitly excluded from layout shift scoring; the metric measures unexpected movement of laid-out content.

B — Treats a measurable regression as an inherent cost of the technique rather than as a sign the skeleton does not match.

D — A real source of layout shift, but it would have existed before the skeleton was added and cannot explain a change that appeared with it.

17. Motion that makes people ill — C

A — Reads the setting as a speed control; it is a request for less motion, and removing a trigger entirely is legitimate for parallax.

B — Points at a real coverage gap, but the central problem is what the rule breaks rather than what it fails to reach.

D — Inverts the mechanism: the trigger is large-field movement signalling self-motion, not abruptness.

18. Why transform is cheap and width is not — B

A — Attributes the difference to where the animation was declared rather than to which pipeline stages the property forces.

C — Overstates compositor independence into a claim that the animation is precomputed and detached from the document.

D — Singles out text rasterisation; layout itself is the main-thread bottleneck, and paint is not the distinguishing factor here.

19. Five ways to put motion on a screen — C

A — Video plays from start to finish and takes no instruction beyond play and stop, so it cannot freeze mid-gesture on an interruption.

B — Gives frame-accurate control but no blending on interruption, and a character-sized sheet costs tens of megabytes of decoded memory.

D — Overstates what requires a custom renderer; state-machine runtimes and even CSS transitions blend from the current value on interruption.

20. What Lottie carries and what it drops — A

B — Export frame rate affects keyframe density and file size, but the runtime redraws at the display rate regardless of the authored frame rate.

C — Expressions are baked to keyframes at export or unsupported; they inflate the file rather than running live.

D — Inverts the two renderers: canvas is generally faster on complex files, and SVG is the one whose cost grows with node count.

21. Animating a drawing that is also a document — C

A — Confuses the dash-arithmetic normalisation with point-count matching, which is what morphing actually requires.

B — Invents a quantisation effect; the attribute changes the units the dash arithmetic uses, not how the stroke is rasterised.

D — Describes a plausible performance optimisation that the attribute does not perform; its effect is purely on the units used for dash and offset values.

22. Why a GIF costs twenty times what a video costs — B

A — GIF indexes into a palette at a fixed bit depth; the number of entries actually used does not shrink the pixel data the way this implies.

C — Both dither methods are applied per frame; the difference is the statistical structure of the noise, not how often it is computed.

D — Local colour tables are a real GIF feature, but dithering does not trigger them; it stays inside the existing palette.

23. Transparent video, and why it is two files — C

A — A real class of bug, but it produces fringing on edges rather than a fully black background, and it would not change the reported pixel format.

B — Plausible as a delivery fault, but ffprobe was run on the file itself, so the answer is in the encode rather than in source selection.

D — Describes how HEVC-alpha stores its layer; VP9 carries alpha in the pixel format itself, which is exactly what the probe reports.

24. Frames in a grid, and the memory they cost — C

A — Reasons from file size, which is the number that is not causing the problem here.

B — A background-position change repaints the element, not the document, and repainting does not evict decoded images.

D — Texture limits are a real hazard at this size, but they cause the sheet itself to fail or downscale rather than producing repeated eviction of unrelated images.

25. Motion that responds to state — A

B — Segments are a workable technique but they do not by themselves decide where the state lives; the duplication is the fault.

C — Tests would have caught the symptom, but the design fault existed before any test was written.

D — Splitting into two files preserves the duplication it claims to solve and adds a second coordination problem.

26. Send the developer a file, not a video — D

A — Treats a feature-support problem as a settings mismatch that could be corrected on re-export.

B — Assumes any missing capability is a version gap rather than something the file format never carried.

C — Blames device capability for content that was never present in the delivered file.

27. The specification a developer can build from — B

A — Overstates the case: easing differences are visible at full speed, which is why easing matters at all.

C — Slowing the animation changes the interpolation, not the number of rendered frames, and it is not a performance diagnostic.

D — Repeat exposure is real, but slowing playback does not correct it, and duration judgement is precisely what quarter speed cannot test.

28. The greyscale image that decides everything — C

A — Pushing the keyer globally to fix an interior problem degrades the edge band, which is the part only the keyer can produce.

B — Erode and blur are edge treatments; they do not raise a partially transparent interior to solid.

D — Describes the premultiplication bug, which darkens edges rather than making a solid interior transparent.

29. The dark fringe and what causes it — A

B — Colour space mismatches shift overall values; they do not selectively change apparent edge hardness in proportion to alpha.

C — Most colour tools leave alpha untouched by default, and the symptom appears even when alpha is explicitly excluded.

D — Inverts the direction: the destructive division happens when unpremultiplying, and a straight file read as premultiplied produces bright edges before any grading.

30. Every blend mode is one line of arithmetic — C

A — Invents a normalisation step; Screen is a fixed per-pixel formula with no reference to the frame's average.

B — A raised black point causes a visible grey wash over dark backgrounds, which is the opposite of the symptom described.

D — Confuses Screen with the base-dependent contrast modes; Screen applies the same formula regardless of the background's value.

31. A green screen is fixed on set, not in the edit — B

A — Reaches for the usual lighting fault when the clean still frame already shows the lighting was adequate.

C — Confuses colour contamination across the subject with a lack of edge resolution in the matte itself.

D — Assumes a parameter can restore detail the recording never contained.

32. Spill is real light, and light wrap is an honest fake — C

A — Despill order affects edges, not a solidly keyed interior region, and a grey jacket would persist after re-ordering.

B — Would produce transparency rather than desaturation, and the jacket would show the background through it.

D — Reasons about the formula's behaviour under a colour cast rather than about the real cause, which is that the operation cannot tell green light from a green object.

33. Drawing the matte by hand — A

B — Invents a constraint; interpolation works across any interval.

C — Confuses roto with tracking; spline interpolation does not analyse image sharpness.

D — Binary subdivision deliberately produces irregular spacing, placing keyframes only where the error demands them.

34. What a tracker can and cannot see — C

A — A pattern larger than the search area is a configuration error that causes failure or jumping, not steady drift with a high score.

B — Blur degrades correlation scores; the high score here indicates the tracker is matching well, just without lateral information.

D — Sub-pixel fitting is less precise on soft features, but that produces jitter rather than a consistent one-directional slide.

35. Solving for a camera, and when you cannot — B

A — Names an ambiguity between two motions, when the actual problem is that depth cannot be recovered from rotation at all.

C — Focal length estimation is genuinely harder on long lenses, but supplying it does not make a nodal pan solvable.

D — Track lifetime does weaken a solve, but a slow pan with long-lived tracks still cannot be solved in 3D.

36. Black level, grain, focus, and the order to do them in — B

A — Double softening is real, but it does not explain why the grain has disappeared, which is the specific symptom.

C — A real grain-matching error, but it would show as wrongly distributed noise rather than as an absence of noise after a blur.

D — Premultiplication affects edge pixels only; the symptom described covers the whole element.

37. What a codec throws away — C

A — Motion vectors are compact; the cost is in the residual detail that prediction fails to capture.

B — Invents a metadata-driven allocation; encoders respond to the picture, not to recorded ISO.

D — Subsampling is applied uniformly across the frame regardless of hue, so it cannot explain a five-fold difference between clips.

38. Frames that stand alone, and frames that do not — B

A — Over-insertion is a real risk but a bitrate concern, not the reason the flag is standard for packaging.

C — Look-ahead affects latency in live encoding but is unrelated to why packagers require fixed keyframe positions.

D — Describes an internal reference-structure problem that packagers do not face; segments simply need to start at aligned keyframes.

39. Quality target or size target — C

A — Reads the preset as a quality dial; with CRF fixed, the quality target does not move.

B — Overcorrects: the preset genuinely changes the result, just in size rather than in quality.

D — More references cost almost nothing in headers and are used precisely because they reduce the residual, which shrinks the file.

40. Why your export looks washed out — A

B — Bit-depth conversion scales rather than shifts, and it does not produce the asymmetric crush-and-clip described.

C — A primaries mismatch shifts hue and saturation rather than crushing black and clipping white.

D — Premultiplication errors affect partially transparent edge pixels only and cannot change values where alpha is one.

41. Counting the steps in a gradient — B

A — True of the delivered file's regrade potential, but the question is about how the file looks as encoded.

C — Bit depth and chroma subsampling are independent; Main 10 is commonly used with 4:2:0.

D — Conflates two genuine remedies: dither is a separate step the encoder does not insert on its own.

42. Log footage and what a LUT is not — C

A — A LUT changes pixel values rather than writing metadata, and nothing about it overrides later operations.

B — Log curves are invertible; the loss here is clipping at the table's boundaries, not the curve's mathematics.

D — Gamut is about colour rather than tonal range, and the crush described is a tonal clip at the bottom of the table.

43. Frame rates, shutter and judder — D

A — Inverts the shutter argument: 120fps footage has a shorter exposure per frame, which is exactly why real high-speed footage looks crisp.

B — Picks a real but minor consequence; slow-motion inserts rarely carry sync sound in the first place.

C — Invents a resolution constraint; both approaches operate at whatever resolution the footage was captured at.

44. Your hero video is why the page feels slow — C

A — Assumes size tracks the size of the palette rather than how much changes from one frame to the next.

B — Believes an encoder changes frame size to protect quality, rather than spending more bits inside a fixed frame size.

D — Reads a small file as a quality failure rather than as content that genuinely needs fewer bits.

45. The tool under most of the others — B

A — A long GOP delays seeking to arbitrary points, but the first keyframe is at the start of the file.

C — An unsupported pixel format usually means the video does not play at all rather than playing late.

D — Attributes to rate control what is really an index-placement problem; players locate byte ranges from the index, not from the rate-control mode.

46. Pressure, gradient, and everything that follows — B

A — Capsule size affects high-frequency directivity rather than creating a low-frequency roll-off that cancels proximity.

C — Proximity effect depends on distance from the source, and 20cm is well within range for a gradient microphone.

D — Some lavaliers have presence lifts, but the reason for the absence of proximity effect is the transducer principle rather than a corrective filter.

47. People forgive a soft picture and leave over bad sound — A

B — Attributes a distance problem to the microphone's specification rather than to where it was placed.

C — Treats level as the issue, when raising gain lifts the voice and the room by exactly the same amount.

D — Expects a tool that learns a steady noise profile to remove reverb, which is the speaker's own voice arriving late.

48. Direct sound, reverberant sound, and the distance between them — C

A — Porous absorption works by air movement through the material, which is why thickness rather than coverage sets the low-frequency limit.

B — Correctly notes the tonal imbalance but implies the boom is only apparent, when the real point is that the absorber cannot reach those frequencies.

D — An air gap genuinely extends an absorber's reach downward, but it does not make thickness irrelevant, and 5cm plus a gap still cannot reach 100Hz.

49. Where the level should sit — C

A — A real consequence of digital attenuation, but minor compared with what has already happened to the peaks.

B — Dither is applied at final bit-depth reduction, not at every fader, and modern hosts work in float where a 6dB cut is lossless.

D — Both are multiplications by a constant factor; the decibel is a ratio in either domain, so this describes a difference that does not exist.

50. Two numbers doing two different jobs — B

A — Folding relocates energy rather than concentrating it, and amplitude is not what makes the result sound wrong.

C — Filter phase behaviour is a real and debated effect, but it is a property of the filter rather than an explanation of aliasing's character.

D — Once folded, aliases are ordinary in-band content; no special interaction is needed to explain why they sound wrong.

51. What the noise floor costs you later — B

A — A contaminated profile is a real error, but it removes vowel content broadly rather than producing isolated chirps.

C — Repeated passes worsen both symptoms, but neither symptom requires a second pass to appear.

D — Window length is a genuine tuning parameter, but no window setting removes the fundamental problem of subtracting an average from a random signal.

52. Two microphones and a comb filter — A

B — Invents a routing constraint; inversion applies to the channel's signal wherever in the chain it sits.

C — Pattern is unrelated to whether an inversion corrects a timing difference.

D — Accurately describes what inversion does to the comb, but presents a consequence as the reason, and the underlying cause is that the difference is temporal rather than polar.

53. Latency, and why a speaker starts stumbling — B

A — Halving the block's duration also halves the time available to process it, so the dropouts return.

C — Compensation aligns tracks in the recorded timeline; it cannot make a live monitor path arrive earlier.

D — Cites the range of maximal disruption as though it were the threshold; speakers notice and stumble well below it.

54. Two devices, one event — C

A — Ordinary crystal tolerance gives fractions of a second per hour, an order of magnitude too small for half a second in eight minutes.

B — A sample-rate mismatch on a scratch track is a real hazard, but it affects only the reference track, not the separately recorded audio being synced.

D — A genuine and common cause, but it is characteristic of phones and screen recorders rather than a camera already set to a named cinema frame rate.

55. Three EQ moves that cover most voices — B

A — Q choice is a real guideline, but no width of boost can recover energy that cancelled.

C — Phase shift around a boost is real but small, and it is not why the boost fails to restore the missing content.

D — Invokes precedence-effect fusion, which affects localisation; the cancellation is genuinely in the waveform.

56. Compression does not make anything louder — D

A — Blames processing order for a level fluctuation the compressor produces on its own.

B — Has the direction reversed: a long release keeps the noise pushed down rather than letting it rise.

C — Treats an audible change in level as a frequency-balance problem.

57. Wind, hiss, and why both are placement problems — C

A — Attack behaviour affects transients, but a slow attack would let the sibilant through unchanged rather than making it louder relative to the voice.

B — Describes a characteristic of some analogue-modelled compressors rather than the general mechanism.

D — Some compressors do have sidechain filtering, but gain reduction is applied to the whole signal including the sibilant band; the effect is relative, not an exemption.

58. Reverb is information about where — C

A — Pre-delay shifts the tail's start; it does not shorten the decay, which continues for its set duration.

B — Pre-delay is a delay before the reverb begins, not a gating of the send; the full reverb still arrives.

D — The implied geometry is correct arithmetic, but the intelligibility gain comes from the temporal separation rather than from the perceived room size.

59. Music under speech — B

A — Ratio scales the depth of each move rather than how often they happen, so the surging would persist.

C — A slow attack delays every duck including the ones you want, and lets the first syllable of each line through undipped.

D — Lowering the bed reduces how much the movement matters but leaves the ducker triggering on the same events at the same frequency.

60. Three boring sounds beat any filter — C

A — Assumes the arrival of a claim proves the licence did not cover the use.

B — Reads 'royalty-free' as meaning uncopyrighted, so a claim must mean the label was false.

D — Imagines the claim arises from editing the audio rather than from matching the underlying recording.

61. The number every platform measures — D

A — Gating applies to the loudness measurement, not to peak measurement, which is continuous.

B — K-weighting is applied for loudness measurement only; peak metering is unweighted.

C — Normalisation upward is real on some services, but the headroom exists for reconstruction and codec overshoot rather than for the gain change.

62. What survives a phone speaker — C

A — Reasons from level, when the difference is that one sound has harmonics and the other does not.

B — Invents a specific processing behaviour; the mechanism is in the speaker's response and the listener's hearing.

D — Describes speaker behaviour in terms that do not correspond to how a diaphragm radiates; the explanation lies in the harmonic structure of the source.

63. The chain, and why the order is not arbitrary — A

B — A filter after the compressor cannot affect what the compressor detected, and phase shift does not relocate energy between bands.

C — Release length changes how long the duck lasts but not that the rumble is triggering it in the first place.

D — The symptom is gain reduction applied to the voice, which happens upstream of the filter wherever the filter's corner is set.

64. What a text-to-speech system cannot read — B

A — The vocoder converts an acoustic representation to a waveform; the flatness originates in the stage that produced that representation.

C — Varying the writing helps in practice, but the absence of cross-sentence context is the cause rather than the uniformity of the input.

D — Systems produce varied contours within a sentence without any tags; the problem is scope, not a default monotone.

65. The vector that is supposed to be you — D

A — Modern vocoders are clean, and the described effect appears over time rather than as a timbral flaw.

B — Longer references improve the match but do not supply the behavioural information the embedding is not designed to carry.

C — A real contamination effect, but it would sound like a space rather than like the wrong person's behaviour.

66. A cloned voice needs a yes you can point to — B

A — Treats consent as a one-time switch about the technique rather than a permission scoped to particular uses.

C — Names a real and separate duty, but disclosure to an audience does not supply the speaker's permission.

D — Offers a remedy in place of a diagnosis; the failure happened when the permission was taken, not when the model was stored.

67. Five per cent, and which five per cent — B

A — Misalignment is a separate real problem; here the words were never spoken at any timestamp.

C — All three error types are weighted equally in the standard formula; the significance is perceptual rather than arithmetic.

D — Describes a plausible mechanism, but such output is scored as an ordinary insertion and counts against the transcript.

68. Taking a mixture apart — B

A — Hybrid separators do operate partly on the waveform, but the artefact described comes from frame-to-frame mask variation rather than phase rotation.

C — A real and common separation error, but it produces dryness rather than a moving, watery texture.

D — Mask-based separators multiply the original mixture rather than resynthesising from a latent, which is why the stems sum back to the source.

69. Generated music, and three separate legal questions — C

A — Terms vary and some do grant transferable rights; the problem is upstream of what the service can grant.

B — Platform matching is a real operational risk, but it is separate from whether the warranty of ownership is sound.

D — Registration is not a precondition for copyright subsisting or for a contractual warranty binding; the difficulty is whether any right exists to convey.

70. What a music licence actually covers — D

A — Treats the recording as inseparable from the composition; they are distinct rights with distinct terms.

B — Inverts which right survives: it is the composition that has expired and the recording that has not.

C — A mechanical licence covers making a new recording of a composition, not using somebody else's existing recording.

71. Frames that were never photographed — B

A — Warping does soften detail, but softening is not the tearing and smearing seen at moving edges.

C — Coarse-to-fine estimation does blur motion boundaries, but the fundamental problem is missing correspondence rather than grid resolution.

D — Describes the smoothness-violation half of the problem accurately, but the irreducible issue is that disoccluded pixels have no source to come from at all.

72. Provenance, watermarks and what to tell people — A

B — Treats three null or weak results as corroborating evidence; two of them carry no information in the negative direction.

C — Assumes universal credential adoption and survival; most capture devices and pipelines write and preserve nothing.

D — Reasons from a default that is neither universal nor robust, and treats a negative watermark result as informative when it is not.

Module test — 1. How motion reads as real

1. A

Linear interpolation spreads the distance evenly, so the midpoint in time is the midpoint in space. A real ease-out puts the object roughly three-quarters of the way across by then. The panel can say Easy Ease while the spacing says otherwise, which is why the frame test beats the label.

2. C

Ease-in-out is slow at both ends, so on anything a user initiates the first hundred milliseconds carry almost no movement. The user has clicked and seen nothing, and they register that as lag rather than as elegance. Ease-out starts the instant they act, which is why it is the curve to learn first.

3. B

Overshoot means going past the value, and on a progress bar the value means something — you have just told the user the download is a hundred and seven per cent done. Springs are excellent for gesture-driven motion because they carry velocity through an interruption, and wrong wherever the number being animated is a real quantity.

4. D

After Effects keeps keyframes at their time values, so a key that sat exactly on frame seven at 24 lands between frames at 25 and is sampled to whichever is nearest. Blender does the opposite and keeps frame numbers, so the duration changes instead. Neither is a bug; set the rate before the first keyframe.

5. D

Delaying the thing under the finger by three frames adds about a hundred and twenty-five milliseconds to the perceived latency of every tap, and under about a hundred milliseconds is what reads as instantaneous. The split is simple: the control responds immediately, and the scenery follows through.

6. B

There is a latency of a hundred and fifty to two hundred and fifty milliseconds before the eye even starts to move, and twenty to forty more to get there. At 24 frames a second that is four to seven frames of an entrance nobody watched. Bring the next element in near where the last one left, or hold a beat.

Module test — 2. Motion that carries meaning

1. A

Michotte's launching effect — A moves, stops beside B, B moves, and people see causation rather than two events — holds only inside a tight window. Insert more than roughly a hundred to a hundred and fifty milliseconds and the impression collapses, and the user has to reason about the connection instead of seeing it.

2. C

The two run together, so the total is the longer of them, and here that is the animation you added. Most people assume an animation fills time already being spent; it usually does not. Start the work on the press, then check by counting frames from touch to readable content.

3. C

Twenty items at forty milliseconds puts the last one more than three-quarters of a second behind the first, which is a queue rather than a gesture. Either divide a maximum total window by the item count, or cap how many items stagger at all and let the rest appear together. Only stagger what is visible.

4. A

Layers moving at different speeds is exactly the depth cue the visual system uses to judge self-motion, which makes parallax the strongest trigger in the set. Slide becomes cross-fade and long travel becomes short travel, but parallax has no reduced form — and the stronger position is to keep it out of the default experience entirely.

5. D

Transform and opacity describe how an already-painted layer is placed and blended, so the browser skips style, layout and paint and can do the remaining work on the compositor thread, which keeps running while the main thread is blocked. Animating left or width forces layout on the main thread, so a long task freezes it.

6. B

If the response arrives inside the window where the interaction still feels instantaneous, an indicator is a visible flash that makes a fast interaction feel faulty. Delaying it removes the flicker from every fast case and costs nothing on the slow ones, which is one of the cheapest improvements to perceived speed there is.

Module test — 3. Shipping the motion

1. B

A video or an animated image plays from start to finish and takes no instruction beyond play and stop. If the motion must pause at forty per cent, reverse, branch on application state or follow a scroll position, that removes video, GIF, WebP, APNG and a plain sprite sheet in one move — and it is usually the question asked last.

2. D

Images inside a Lottie are base64-encoded into the JSON, which adds about a third on top of losing the benefit of image compression. Shape count is a rendering cost rather than a size one: a file under fifty kilobytes that still stutters has a shape problem, and a file in the hundreds of kilobytes has an image in it.

3. C

On an SVG element the transform origin is resolved against a box that transform-box chooses, and the default is the element's own user-space coordinates rather than its bounding box. So the gear rotates around the viewBox origin and swings. One line fixes it, and most reports of an SVG animation flying off the screen are this.

4. A

Width times height times four bytes, held for as long as the element exists, and disk size is irrelevant to it. This is the usual way a sprite-sheet animation kills a phone, and the symptom is not a slow animation but the tab reloading, or unrelated images flickering as they are evicted and decoded again.

5. A

With twenty-four steps the animation is divided into twenty-four equal holds and the last hold shows the twenty-fourth frame; the step after that would be the wrap. Animating to the full sheet width is correct. Subtracting a frame leaves a hold with nothing in it, which is the blank flash.

6. C

Interruption is the expensive part of interactive motion, not the animation itself. Blending from wherever the element has got to is what makes a redirected gesture feel continuous, and it is tedious to write by hand for every pair of states. CSS transitions give you the same thing free, because the engine starts from the current computed value.

Module test — 4. Putting two images together

1. B

Asked to solve a whole frame, a keyer has to compromise between a dark jacket that goes see-through and a light stand that will not leave. Confined to the band between a hand-drawn garbage matte and a solid core, it can be pushed much harder than it could globally. Core, or key and garbage, is the combination.

2. D

Premultiplied RGB has already been multiplied by alpha, so the straight formula multiplies every edge pixel a second time — at an alpha of a half the foreground contributes a quarter instead of a half. The reverse error gives a bright halo. The colour of the fringe is the whole diagnosis, and the fix is the interpretation setting rather than choking the matte.

3. D

The algorithm is one line: where green exceeds the average of red and blue, clamp it to that average. No algorithm distinguishes green light bounced off the screen from a green object, because in the image they are the same photons. The professional move is to tint the despilled luminance from the new environment afterwards, and to call it a hue shift rather than a recovery.

4. B

Through a small window onto a straight edge, motion parallel to that edge changes nothing in the pixels: the patch at one position looks identical to the patch five pixels along. The tracker can measure the component perpendicular to the edge and has no information about the one along it. A good feature has contrast in two directions.

5. A

With no translation, a near object and a distant mountain shift by the same amount on the sensor, so the footage carries no depth information at all. It is not a software limit. A nodal or tripod mode solves rotation only, which is fine for a sky replacement and useless for placing an object on the floor. A metre of lateral movement on set fixes it.

6. C

The camera is the outermost thing in the chain, so anything applied after grain and distortion is physically happening inside the lens. Grain in particular must come after any blur, because blurring grain destroys it — which is also why the most accurate grain is not synthesised but lifted from a flat area of the plate by subtracting a blurred copy of itself.

Module test — 5. Codecs, colour and delivery

1. C

A codec predicts and then codes the difference between prediction and reality, so unpredictable content — rain, confetti, foliage in wind, grain — costs real bits while a static interview costs almost none. Content, not resolution, sets the bitrate a clip needs, and no encoder setting changes that.

2. A

To show frame forty-three of a group of pictures starting at frame one, the decoder must decode everything from the keyframe forward, and scrubbing backwards is worse because each step back repeats that work. All-intra codecs make every frame independent at the cost of size, and a proxy buys the same responsiveness for a fraction of the storage.

3. D

CRF sets the quality the encoder aims at; the preset sets how much search effort it spends finding a cheap way to reach it. So a slower preset gives a smaller file of the same quality rather than a better-looking one. Medium to slow is typically five to ten per cent of the size for roughly double the encode time.

4. B

Full-range data read as limited has sixteen to two hundred and thirty-five expanded across the whole range, so everything below sixteen clamps to black and everything above two hundred and thirty-five to white. The opposite error — limited read as full — is the milky, low-contrast failure. Tag the range on export, because an untagged file leaves every player to guess.

5. B

An eight-bit channel has two hundred and fifty-six levels and a realistic gradient uses only part of them, so fifty levels over a thousand pixels is a step every twenty-one pixels — and the eye detects an edge of about one per cent in luminance between two large flat areas. Ten bits makes the same range steps of about five pixels; dither trades the structured edge for random error.

6. D

An MP4 holds an index of the whole file, and ffmpeg writes it at the end by default because it does not know the final sizes until it has finished. A browser streaming that file has to reach the end before it can play a frame. Moving the index to the front costs one rewrite, and people routinely diagnose its absence as a slow server.

Module test — 6. Capturing sound

1. C

An interference tube cancels off-axis sound only where its length is comparable to the wavelength, so a thirty-centimetre tube is directional above roughly a kilohertz and broadly cardioid below five hundred hertz. Reflections arrive from every direction at every frequency and the tube colours them unevenly, which is the hollow, boxy sound people report indoors.

2. A

The information at the top of a clipped waveform is not attenuated, it is absent, and no processing recovers the shape. Recording quietly only moves you closer to a noise floor that twenty-four bits has plenty of room above. The two ends of the meter are not symmetrical, which is why peaks belong around minus twelve to minus six.

3. A

Frequencies above half the sample rate do not disappear; they fold back, mirrored around Nyquist, so forty-four point one minus twenty-five gives nineteen point one kilohertz. It is audible, it was never in the room, and it bears no harmonic relationship to anything else, which is why aliasing sounds metallic rather than merely bright.

4. C

Room noise is diffuse and roughly constant wherever you stand, while the voice obeys the inverse square law and gains about six decibels per halving of distance. Sixty to thirty to fifteen centimetres is two halvings, so the voice comes up twelve decibels against a noise level that has not moved. It costs nothing.

5. B

A fixed time delay is a different fraction of a wavelength at every frequency, which is what produces the comb: a first cancellation at three hundred and forty-three divided by twice the path difference, then odd multiples of it. A polarity flip applies equally at all frequencies, so it cannot undo a difference caused by time. Move a microphone, or slide one track.

6. D

The ratio of a thousand to a thousand and one is three point six seconds an hour, and it is by far the commonest reported drift. It is not a clock problem at all but a project setting. Genuine crystal drift is a fraction of that, around a sixth of a second an hour, and timecode fixes the offset while doing nothing about either.

Module test — 7. Shaping and mixing sound

1. D

Rumble you cannot hear still drives the gain reduction, so the voice ducks every time a lorry passes or a foot touches a desk leg. High-pass first and the compressor responds to the voice. Each stage in the chain changes what the next one sees, which is where the whole order of operations comes from.

2. B

A compressor pulls the loudest material down, and sibilants are usually not the loudest material, so everything else comes down around them and they stand proud. De-essing first means re-creating the problem at the next stage. A de-esser is a compressor with a filtered sidechain: three to six decibels on the worst syllables and nothing the rest of the time.

3. A

A compressor only ever turns loud material down; the make-up gain afterwards is what raises everything, including whatever noise sits between the words. During speech the level is being pulled down, and in the gap nothing is, so the full make-up gain lands on the room. Use less gain reduction, gate gently, or fix the noise floor at the source.

4. C

Pre-delay is the gap between the direct sound and the start of the reverb, at about thirty-four centimetres of distance per millisecond. Increasing it separates the voice from its own tail in time, so the consonants stay intelligible even with a generous reverb. Reaching for the level first gives up the space before trying the control that caused the problem.

5. C

A ducker reacts after the fact and reacts to every breath, lip smack and stray noise, which produces the surging bed everybody recognises in an automated intro. A person knows the line is coming and starts the move two hundred milliseconds early over half a second. For live work, a ducker with a long release and a capped depth is the pragmatic answer.

6. A

Platforms measure integrated loudness and normalise playback, so squashing buys a flatter, more fatiguing version arriving at the same perceived volume as everybody else's. Mix to sound right, measure, and correct with a gain change; reach for a limiter only for the last decibel or two, and leave a decibel of true-peak headroom.

Module test — 8. Machines that make sound and motion

1. D

Stress is meaning — the same seven words mean seven different things depending on which one is emphasised — and none of it is in the input. The vocoder is largely a solved problem; almost everything that sounds wrong happens in the first stage, where the model guesses prosody from statistical regularities in its training data.

2. B

A conventional recogniser fails visibly and produces garbage you can see; a generative one fails invisibly and produces a well-formed sentence. A voice-activity detector to skip non-speech helps, and so does stopping an error propagating forward, but neither removes the need to read the transcript.

3. B

If two sources have energy in the same time-frequency bin, the file holds their sum and nothing else; the individual contributions were never recorded. Separation estimates the split from learned statistics, so it is close to clean where sources rarely overlap and a guess where they do. Buried in an arrangement the artefacts hide; exposed at full level they do not.

4. D

The information identifying those characters is not in the file. A super-resolution model produces a high-resolution image consistent with the input, and consistent is not the same as correct. There is no per-pixel uncertainty map and no warning in the output, which is why upscaled material is fine for viewing and dangerous as a record.

5. C

Consent for a voice is scoped: to a project, to a set of uses, to a period, and to whether the model itself is retained and who can run it. The gap between permission for one video and a voice model sitting on a server for two years is where most real disputes sit, and disclosure to the audience is a separate duty on top of it.

6. A

The manifest is metadata: screenshot the image, re-encode the video, or pass it through a platform that rewrites files, and it is gone. Absence of a credential is evidence of nothing. A detected watermark is meaningful evidence that content came from a particular generator; an undetected one is not evidence either way.

Motion, Sound and Music — final examination

1. A ***1. How motion reads as real***

Linear is right whenever there is no start and no stop. A spinner, a ticker or a looping background has no moment of acceleration or deceleration to describe, and easing them makes them lurch once per cycle. Everything that begins or ends because something happened wants a curve.

2. C ***1. How motion reads as real***

Doubling the travel wants roughly one point four times the duration, not double, so two doublings is about twice the time. Scaling in proportion produces a crawl on large screens and keeping the duration fixed makes a long move read as a snap. What is being held constant is perceived speed.

3. B ***1. How motion reads as real***

Distance and duration are fixed, so slowing both ends forces the travel into a burst in the middle. The eye reads the burst as the real motion and the slow ends as hesitation. Dragging the outgoing handle's influence from a third to about three-quarters turns a mushy move into one that commits and then settles.

4. D ***1. How motion reads as real***

The ratio is damping divided by twice the square root of stiffness times mass. Below one is underdamped and overshoots, which is what people mean by springy; above one is overdamped and creeps. Exactly one is critical damping: the fastest arrival with no overshoot at all.

5. D ***1. How motion reads as real***

An arc is a claim about how something moves through space, and a drawer runs on one axis. A card that travels across and down genuinely moves in two dimensions and arcs naturally. Which way it bends carries meaning too — upward reads as light, downward as heavy — so it is a choice rather than a default.

6. B ***1. How motion reads as real***

Exaggeration is what makes a single viewing legible and memorable, and it is what makes the four-hundredth unbearable. The rule is that the more often a motion will be seen, the closer to literal it should be: a splash screen can be exaggerated, a keyboard cannot. Timing, staging and arcs survive untouched.

7. A ***1. How motion reads as real***

The periphery resolves competition roughly in this order: onset of motion first, then faces, then luminance contrast, then text once it is fixated. A thing that starts moving beats a thing already moving, and both beat something still. This is why a small animated element in a corner destroys a piece.

8. C ***2. Motion that carries meaning***

The same row vanishing instantly could have been deleted, filtered out, or a rendering bug. Motion that answers what just happened, where a thing came from, or whether the system is alive earns its place. Everything else is paid for out of the user's attention with nothing in return.

9. C ***2. Motion that carries meaning***

Steps three and four of FLIP use transform alone, which the browser can animate on the compositor. Animating geometry properties directly means a full layout calculation on every frame, which is where the dropped frames come from. The four steps are also where every shared-element transition goes wrong, so they are worth knowing even when a framework does them.

10. A ***2. Motion that carries meaning***

Scaling a container scales its contents, so type grows and snaps back at the end. Counter-scaling the child by the inverse of the parent's scale each frame keeps the text at its true size throughout. Animating width and height independently is the other classic failure: it squashes anything whose aspect ratio changes.

11. D ***2. Motion that carries meaning***

Reversing the stagger on exit keeps the two movements distinguishable; replaying the entrance in the same order reads as a loop rather than a departure. Randomising is the other tempting mistake — the eye looks for the pattern, cannot find one, and reads it as broken rather than organic.

12. B ***2. Motion that carries meaning***

A viewer needs roughly two and a half to three seconds of still, settled text to take in a name and a role, and longer in a second language. Animation time is not reading time. Budget the hold first, then add the entrance and exit on top: six seconds feels long in the timeline and correct to a viewer.

13. B ***2. Motion that carries meaning***

The first time somebody decides a hundred and eighty milliseconds was too slow, a value-named token becomes a contradiction in the codebase, which is worse than a raw number. Role names also survive platform differences, and redefining them inside a reduced-motion query is the cheapest way to make every future component inherit the right behaviour.

14. D ***2. Motion that carries meaning***

A determinate bar claims you know how far along you are, so use it when you do — a byte count, a step count. The remembered failure is a bar that stalls at ninety per cent, and one that decelerates toward the end feels much longer than one of identical duration that accelerates. An indeterminate indicator makes no promise it cannot keep.

15. C ***3. Shipping the motion***

VP9 with alpha in WebM covers Chrome, Firefox and Edge; Safari needs HEVC with alpha and the hvc1 tag. Both have to be produced and both have to be tested, and if either is missing a large share of viewers see a black box. Ask first whether the background is genuinely arbitrary — very often it is one known colour.

16. A ***3. Shipping the motion***

The self-drawing line trick needs a dash and a gap each as long as the whole path, then slides the pattern with the dash offset. Without a declared path length you have to measure the real one in JavaScript. With it, the stylesheet can use a hundred everywhere whatever the geometry, which removes the one awkward step.

17. A ***3. Shipping the motion***

The SVG renderer creates a DOM node per shape, group and mask, and the browser does style and layout work on all of them every frame. Canvas draws to one element and is far faster on complex files, at the cost of per-shape CSS control and of redrawing on resize. Bytes are not the problem when a small file stutters.

18. C ***3. Shipping the motion***

Most converters compute one palette from one frame. Generating it from the whole clip, weighted toward the parts that move, is the single biggest quality change. Bayer dithering then compresses far better than error diffusion because the pattern is regular. After that the levers are dropping the frame rate to twelve or fifteen and dropping the resolution.

19. B ***3. Shipping the motion***

Those three tests catch the whole class of problem at once: a video has a fixed size, a background and baked colours, so it fails all three. A Lottie or an SVG scales, recolours in code, and can be paused, reversed and scrubbed, and is smaller. The animation was not wrong; the file was.

20. D ***3. Shipping the motion***

A recording shows one playthrough of one happy path. It does not show the trigger, what interrupts it, the exit, the loop count, the behaviour at other sizes, the reduced-motion substitution, or which property is moving — a scale and a width look similar and behave completely differently. That is what the written specification is for.

21. D ***3. Shipping the motion***

A listener recalculating on every scroll event is main-thread work at exactly the moment the main thread is busiest. Declaring the timeline lets the browser run the animation on the compositor. Where support is missing it degrades to the animation not running, so the default state has to be correct on its own.

22. B ***4. Putting two images together***

Foreground times alpha plus background times one minus alpha. At one you get the foreground, at zero the background, and in between a weighted mix — which is exactly what a semi-transparent edge is. The grey values are the whole art of compositing, because edges are where a bad composite gives itself away.

23. A ***4. Putting two images together***

A region at nought point nine five where it should be one is invisible at normal exposure and shows in the composite as a faint transparency of the background through your subject. Gaining down or viewing the inverse finds the opposite fault: background contamination at nought point nought two, which reads as a faint glow where there should be nothing.

24. C ***4. Putting two images together***

Screen is one minus the product of the inverted values, so one minus nought point one times nought point two is nought point nine eight — barely brighter than the sky already was. Light elements only read against darker backgrounds. For fire over a bright sky you need a real matte rather than a blend mode.

25. C ***4. Putting two images together***

Blur the background substantially, multiply it by the edge band — the matte minus an eroded copy of itself — and screen or add it onto the foreground. A bright window behind the subject then throws a faint bloom onto their shoulder in the right colour and place. It is a frank fake and the highest-return step after despill.

26. A ***4. Putting two images together***

The software interpolates between keyframes, so you should only be fixing what interpolation got wrong, and the error is largest at the midpoint. Halving intervals typically needs a keyframe every five to fifteen frames rather than every one. Split the subject into simply-moving parts and keep the point count low.

27. D ***4. Putting two images together***

They are scored on region-level accuracy, so being right about the interior scores well while hair and motion blur — the pixels a composite is judged on — go wrong. They are also temporally unstable, because a per-frame model knows nothing about the previous frame, so the edge crawls. For garbage and core mattes they are a several-fold speed-up, free.

28. B ***4. Putting two images together***

Project each solved point back through each solved camera and measure how far it lands from where the feature actually was. Under about half a pixel on a 1080p plate is something to build on, one to two is marginal, and above three no amount of extra tracks will help. Look for unhandled distortion, tracks on moving objects, or rolling shutter.

29. B ***5. Codecs, colour and delivery***

A codec predicts and then codes the difference between prediction and reality, and noise is by definition unpredictable, so every noisy pixel costs bits spent reproducing noise. A light denoise before encoding moves those bits from the noise to the picture, and it is why grain added in compositing partly disappears after upload.

30. D ***5. Codecs, colour and delivery***

HLS and DASH cut the video into segments and switch quality at segment boundaries, which must be keyframes. If the interval does not divide evenly into the segment length, the packager inserts extra keyframes or the segments drift. For ordinary viewing you want scene-cut keyframes on, because a cut is where a fresh frame is cheap and useful.

31. C ***5. Codecs, colour and delivery***

In a single pass the encoder does not yet know that a difficult scene is coming, so it cannot save bits for it. The first pass analyses the whole file and records how complex each part is, and the second distributes the budget accordingly. Two-pass is what fitting a file into a size means; CRF is right whenever size is not the constraint.

32. A ***5. Codecs, colour and delivery***

The silicon implements a smaller search, so at the same file size a hardware encode is usually described as one to two CRF points worse, and more on difficult content. It is ten to twenty times faster, which is exactly what live streaming and screen recording need. Software for anything distributed; hardware where the clock is the problem.

33. A ***5. Codecs, colour and delivery***

The table was built expecting a particular input. Footage a stop under lands in the wrong part of it, and values outside the domain are clamped or badly extrapolated, so highlights and shadows are discarded before you can grade them. The order is conversion, then exposure and white balance, then the look, then a final trim.

34. C ***5. Codecs, colour and delivery***

A 180-degree shutter exposes for half of each frame's duration, so every frame carries blur covering half the interval to the next and consecutive frames overlap in what they record. Shutter speed is one over twice the frame rate. Outdoors that usually needs a neutral-density filter, which is why they are standard kit rather than an accessory.

35. B ***5. Codecs, colour and delivery***

Interpolation invents four frames out of every five and fails in characteristic places — occlusion boundaries, thin structures, repetitive texture, heavy blur — which look like melting or tearing. Shooting at the rate you intend to slow to gives real frames. Conforming just plays the same frames slower, and duplicating produces judder.

36. D ***6. Capturing sound***

Close to the source the direct sound already dominates, so directionality buys little — and omni brings real advantages: a flat response, no proximity effect as the microphone rides up the chest, and the least wind and handling noise. A pressure capsule is omnidirectional because pressure is a scalar with no direction.

37. D ***6. Capturing sound***

Absorption reduces reflections inside a room by converting air movement to heat. Isolation stops sound passing between spaces and needs mass, airtightness and decoupling. A duvet over your head will transform how your voice sounds and will not touch the drill. If noise is coming in, treatment is the wrong tool.

38. B ***6. Capturing sound***

A porous absorber works by air movement, which is greatest a quarter wavelength from a hard surface. At a hundred hertz the wavelength is three point four metres and a quarter of it is eighty-six centimetres; at a kilohertz it is eight and a half. Thin foam removes exactly the frequencies that were not the problem.

39. A ***6. Capturing sound***

Each bit adds about six decibels, so sixteen gives roughly ninety-six and twenty-four roughly a hundred and forty-four. The best front ends manage a noise floor around a hundred and twenty decibels down and most interfaces nearer a hundred, so the bottom bits of a 24-bit file record circuit noise. Giving away twelve decibels costs nothing you had.

40. C ***6. Capturing sound***

Spectral subtraction measures the noise's average level per band and reduces each band by that amount. Because the noise is random, a band's actual level at any instant is above or below its average, so the residual is sometimes nothing and sometimes a short burst in one band — heard as chirps. Six to ten decibels is usually inaudible; twenty is not.

41. C ***6. Capturing sound***

Buffer size divided by sample rate is ten point seven milliseconds, and that is paid on the way in and on the way out, plus a millisecond or two of converter latency. Above about twenty milliseconds most people notice something is wrong even if they cannot name it. Direct monitoring in hardware sidesteps the whole chain.

42. A ***6. Capturing sound***

Waveform matching cross-correlates the camera's low-quality track with the recorder's good one and slides the clips together, which is free in Resolve, Kdenlive, Shotcut and Blender. It needs the camera to have recorded something overlapping in content, and it is also the only thing that saves a take when a card fills or a recorder is knocked out of record.

43. D ***7. Shaping and mixing sound***

A resonance is narrow, so amplifying it makes it unmistakable. Note the frequency, then cut there by three to six decibels with a moderate Q. The trap is that a ten-decibel boost makes everything sound bad and there is always a worst frequency — if nothing jumps out as obviously wrong, there is nothing to fix.

44. B ***7. Shaping and mixing sound***

A resonance is a spike at one frequency, so a surgical cut removes it without touching anything else. Tonal character is a shape across a region, which is why a pleasant boost is broad — and a narrow boost sounds like a resonance because it has just created one. Roughly: cuts at a Q of two to six, boosts at nought point seven to one point four.

45. B ***7. Shaping and mixing sound***

Around ten milliseconds is fast enough to catch the peaks and slow enough to let the consonant transients through. A very fast attack clamps down on exactly those transients and the voice loses its edge without the meter showing anything alarming. Aim for three to six decibels of gain reduction on the loudest phrases and none on the quiet ones.

46. D ***7. Shaping and mixing sound***

A plosive is a jet of air rather than a loud sound, and the jet is directional while the sound is not — so speaking past the capsule loses almost nothing and avoids the blast entirely. A pop filter needs five to ten centimetres of gap to break the airflow. A high-pass reduces what is left rather than preventing it.

47. C ***7. Shaping and mixing sound***

The dry voice goes to the output untouched, so intelligibility is preserved and the balance stays adjustable. Several sources sharing one reverb is what makes them sound as if they are in the same room. And the return can be processed on its own: rolled off above five to eight kilohertz, and high-passed below two or three hundred hertz.

48. A ***7. Shaping and mixing sound***

Panned apart, two signals reach your ears separately and cancellations are much less obvious. Summed to mono they collapse into the same signal and the notches appear in full. A phone speaker, a laptop, a smart speaker and most soundbars in their default mode are all mono, so a voice that disappears in mono disappears for a large share of the audience.

49. A ***7. Shaping and mixing sound***

Track processing solves what is specific to one voice; bus processing makes several elements behave as one, and one or two decibels of gentle compression across a dialogue bus is what makes a conversation feel like one scene. Master processing is loudness and safety only — anything more there is a decision that belonged further up.

50. C ***8. Machines that make sound and motion***

Numbers, dates, ranges and version strings all have conventions the text does not state. A say-as tag works where the engine supports it, and writing the words out is often quicker and always works. A pronunciation lexicon is worth building once for any project with recurring names and terms.

51. B ***8. Machines that make sound and motion***

A speaker encoder maps a voice into a space learned from the voices it saw. A voice far from that distribution gets a less distinctive embedding and the output drifts toward the training average. The same mechanism makes clones in low-resource languages much weaker, and it is an honest limit rather than a tuning problem.

52. D ***8. Machines that make sound and motion***

Five per cent is one word in twenty, which on a thousand-word video is fifty errors. Worse, they are not spread evenly: they concentrate on the content words a viewer is relying on the captions for, so a transcript can be right about almost everything and wrong about every name in it.

53. D ***8. Machines that make sound and motion***

Mask-based separation constructs masks that roughly sum to one, so the stems add back to something very close to the original. You can separate, fix the stem containing a cough or a phone, and add the others back with the parts you did not touch essentially unchanged — far less destructive than processing the whole mix for one event.

54. B ***8. Machines that make sound and motion***

Three questions get run together in public discussion and have different answers in different places: was the training lawful, does this output infringe, and does anyone own it. Substantial similarity is about the specific output, not about the technology, and it is the one you can manage by checking what you generated.

55. A ***8. Machines that make sound and motion***

It means no royalty per broadcast, stream or copy. Territory, term, permitted media, number of productions and exclusivity are all still specified, and broadcast, cinema and paid advertising are commonly excluded or priced separately — which is the clause that catches people when a web video becomes an advert.

56. C ***8. Machines that make sound and motion***

Content ID compares your audio against registered recordings and produces a claim. It is a matching process run under a platform's own rules, not an adjudication of your rights. Libraries usually provide a clearance process, and many require you to whitelist your channel before uploading — a two-minute job you can only do in advance.