THE BUILD PACK

Faceless course videos, built by Claude Code.

Paste in a transcript — or a recording of someone explaining something — and get back a branded, narrated, animated instructional video. No camera. No editor. No video software.

Eight prompts, in order. Every one of them asks you what it needs. You never fill in a blank.

NEXTLEVEL

NextLevel Digital · free · nothing to sign up for

This is for the training video nobody ever gets around to making. Onboarding, course lessons, "how this feature works," internal SOPs. You already know the material — you just don't want to be on camera and you don't want to learn video software.

How to use this

Open Claude Code in an empty folder and paste the prompts **in order**. Read what comes back before you move on.

You never have to fill anything in. Every prompt is written to interview you — it asks for your colors, your fonts, your logo, your topic, one thing at a time, and waits for your answers. Copy the whole

block exactly as it is.

---

01 What you need before you start

Two accounts you actually have to create

SERVICE	WHAT IT DOES	WHAT YOU DO
ElevenLabs	The AI narrator. Reads your script out loud.	Paid account. Pick one voice from their library, copy your API key. Billed by how much speech you generate — the bigger of the two costs.
OpenAI	Transcription (Whisper). Listens to the narration and reports when every word is spoken.	API key. Billed per minute of audio transcribed — much cheaper. **Must return word-level timestamps.**

Everything else Claude Code sets up — no accounts, no keys, no cost

THING	WHAT IT IS
Node.js	Runs the scripts. Install it once if you don't have it.
Playwright	Free open-source. Runs an invisible browser that screenshots your page. Not part of Claude Code — Claude installs and drives it.
ffmpeg	Free. Glues frames and audio into an MP4.

And your source material

Either works: a **written transcript** you paste in, or a **recording** of someone talking through the topic. It does not need to be polished — rambling is fine, and honestly rambling sounds more human once it's cleaned up. If it's a recording, transcribe it first (Claude can do that too) and paste the text.

> **The one thing that will sink this if you get it wrong**
>
> Your transcription has to return timings **per word**, not per sentence or per segment. The entire timing system depends on knowing the exact moment each individual word is spoken. If your transcription tool only does sentences, stop and switch tools.

02 The prompts

PROMPT 1 · SET UP THE MACHINE — PASTE THIS FIRST

I want to build a reusable system that turns a transcript into a finished, branded, animated instructional video. Think faceless course lessons, onboarding videos, product walkthroughs. No camera, no video editor, no NLE. I'll be making more of these, so build it as a reusable machine, not a one-off.

DO NOT BUILD ANYTHING YET. First, interview me. Ask me these one at a time, wait for each answer, and don't move on until you have it:

1. What is the video about, and who is watching it?
2. Do you have a transcript ready, or a recording I should transcribe first?
3. What are your brand colors? Ask for a dark background color, a primary accent, and an optional second accent. Hex codes if I have them, and if I don't, ask me for my website and pull them.
4. What fonts? Offer to suggest a pairing if I don't know.
5. Do you have a logo file? Where is it?
6. Do you have screenshots of whatever we're explaining? These matter more than people expect.
7. Are there any words your industry uses that a text-to-speech voice would mispronounce?
8. Any house terms - words you always use, or never use?
9. Should the narration use "I", "we", or neither?
10. Do you have an ElevenLabs API key and an OpenAI API key yet? If I don't, walk me through getting them before we continue.

Save all of my answers into a file called PROJECT.md at the root, and re-read it at the start of every future step so I never have to repeat myself.

Then set up this structure:

```
  scripts/                 the narration scripts
  shared/engine.js         the timing + animation engine   (REUSED)
  shared/brand.css         the design system               (REUSED)
  shared/assets/           logo + screenshots              (REUSED)
  videos/<slug>/index.html the scenes for one video
  videos/<slug>/           its vo.mp3, words.js, out/<slug>.mp4
  gen-audio.mjs  make-words.mjs  render.mjs
```

The shared/ folder is the whole point: the first video pays for it, every video after that just adds a folder under videos/. Never let brand or engine code live inside a single video's folder.

Here is how the finished pipeline works, so you understand what you're building toward:

1. A rough transcript gets cleaned into a narration script
2. ElevenLabs reads that script and produces a voiceover mp3

```
   3. OpenAI Whisper transcribes that mp3 back with WORD-LEVEL
      timestamps
   4. Slides are built as an HTML page, each anchored to a spoken
      phrase rather than to a clock time
   5. Headless Chromium seeks the page frame by frame, screenshotting
   6. ffmpeg muxes the frames with the voiceover, music and SFX

Standing rules for everything you build in this project:
- Every text element is sized for a 1920x1080 VIDEO FRAME, not a web
  page. Body text 32-40px minimum. Headlines 80px+. Nothing below
  16px ever. Web-sized type is unreadable once compressed.
- Keep the bottom-left corner of the frame clear.
- On-screen text stays short. The narration carries the detail, the
  visual carries the idea.
- No placeholder or lorem content, ever. If you need a real value you
  don't have, stop and ask me.

Start by asking me question 1.
```

Why this one prompt does so much

It creates `PROJECT.md`. Everything after this reads from that file, so you answer each question once and never retype your brand colors or your house rules again. If you come back to this project in a month, Claude picks up where you left off.

PROMPT 2 · THE DESIGN SYSTEM — YOU ONLY EVER DO THIS ONCE

```
Read PROJECT.md, then build shared/brand.css — the design system
every video in this project will use.

Build classes for: a full-screen scene, a big display headline, a
sub-headline, body copy, a small uppercase eyebrow label, chips and
badges, a stat callout, a two-column comparison, and a caption bar
pinned to the bottom of the frame.

Sizing is for a 1920x1080 video frame, not a webpage:
  headlines 80px+ (the display font can go 120px)
  body 32-40px
  captions and diagram labels 24-30px
  absolute floor 16px, and only for tiny eyebrow labels

Use ONLY the colors in PROJECT.md. Do not introduce a new accent
color, and do not invent a lighter or darker shade of one of mine.
If you need a tint that isn't in my list, ask me first.

Then build one sample scene using it, screenshot it at 1920x1080,
and show me before you write anything else. Ask me what to change.
Expect me to want two or three rounds on this.
```

Let this one take a few rounds

This is the step that decides whether the output looks like *your* company or like a template with your logo on it. It's also the step that pays for every future video, since they all inherit it. Look at the sample, say what's wrong in plain language, repeat.

PROMPT 3 · THE SCRIPT

```
Read PROJECT.md. Here is the raw transcript:

--- PASTE YOUR TRANSCRIPT BELOW THIS LINE ---


--- END OF TRANSCRIPT ---

Turn it into a narration script and save it to scripts/ using a short
slug you pick from the topic. Tell me the slug you chose.

Rules:
- Keep their voice and their points. Do NOT rewrite this into
  marketing copy. It must still sound like the person who recorded
  it, only cleaned up. If it sounds like a robot wrote it, you went
  too far.
- Fix grammar. Cut filler, repeated words, and the sign-off.
- Untangle sentences that double back on themselves.
- Do NOT soften or weaken any warning, risk, or caveat they gave.
  Tighten how it's said, never how strong it is.
- Apply the house terms and the first-person rule from PROJECT.md.

Then show me a before/after of every change you made, and flag
anything you were unsure about.
```

PROMPT 4 · VOICE + WORD TIMINGS

```
Read PROJECT.md, then build two scripts.

First, if I haven't already given you a voice, help me pick one:
list a few ElevenLabs voices that would suit this material, tell me
how to preview them on their site, and save my choice to PROJECT.md.

gen-audio.mjs — takes a video slug and:
  1. Reads the TTS copy of the script
  2. Sends it to ElevenLabs using my key from .env
     - model: eleven_multilingual_v2
     - output_format: mp3_44100_128
     - voice_settings: stability 0.55, similarity_boost 0.75,
       style 0.28, use_speaker_boost true
     - apply_text_normalization: on
  3. Runs the result through ffmpeg with atempo=1.06 — a subtle
     speed-up that stops informational narration dragging
  4. Saves videos/<slug>/vo.mp3
  5. Sends that mp3 to OpenAI Whisper
     - model: whisper-1
     - response_format: verbose_json
     - timestamp_granularities[]: word          ⟵ REQUIRED
  6. Saves videos/<slug>/vo.whisper.json

make-words.mjs — reads vo.whisper.json AND the clean script, and
writes videos/<slug>/words.js containing:
  window.__WORDS     every word with start and end times
  window.__VO_DUR    total duration
  window.__CAP_TEXT  the clean script text, whitespace-normalized

Critical: captions are built from the CLEAN script but TIMED using
the Whisper words. Timing comes from what was actually said; spelling
comes from what I actually wrote. They are allowed to disagree.

Now the pronunciation step. Make a second copy of the script with
".tts" in the filename — this copy is only ever read by the voice
engine, never shown on screen. Using the mispronounced words I listed
in PROJECT.md, respell them phonetically in that copy only. Then read
back the whole script yourself and tell me any OTHER word you expect
a text-to-speech voice to get wrong, and respell those too.
```

The two-file trick — the detail that makes it shippable

Text-to-speech mispronounces industry vocabulary. Ours reads *lead* (a sales lead) as *led*, like the metal, every single time. So the file the voice reads spells it `leed`, and the file the captions come from spells it `lead`. Two files, one word different. It's the gap between "pretty good" and something you'd actually publish.

PROMPT 5 · THE ENGINE — THE PART THAT MAKES THIS A MACHINE

```
Build shared/engine.js. It drives all animation from the word timings
in words.js. This is the most important file in the project.

Scenes and elements:
- A scene is <div class="scene" data-anchor="short phrase">. The
  engine finds when that phrase is spoken in __WORDS and starts the
  scene at that moment. Timing comes from the voice, never from me.
- Any element can carry data-cue="phrase" to appear at the exact
  moment that phrase is spoken.
- Or data-in="0.6" for N seconds after its own scene starts.
- Optional data-out="2.4" to exit early.
- data-anim controls entrance: up, slam, pop, drop, grow, draw,
  blur, fade.
- data-sfx fires a sound effect on that element's reveal.
- Use real easing — ease-out cubic and quart, expo-out, back-out,
  elastic-out. No linear anything.

Resolve every data-cue INDEPENDENTLY by searching the whole word
list. Do NOT use a single advancing cursor. Page order does not match
narration order — connector graphics are usually declared before the
nodes they connect — and a cursor overshoots, then silently fails to
resolve every cue after it. The symptom is a diagram that pops in all
at once instead of building piece by piece.

Captions:
- Build them from __CAP_TEXT, timed to __WORDS.
- Break on natural phrases, not fixed word counts.
- Normalize the pronunciation respellings when matching, so the
  respelled word in the audio still matches the correct spelling in
  my script.
- Color keywords by type: numbers, warnings, positives, key terms.

Two modes:
- Open index.html in a browser and it plays in real time on estimated
  timing, with no voiceover. This is how I check the design.
- With window.__FRAMESEEK = true it switches to exact-seek mode.

Expose:
- window.__renderAt(t)  puts the page in its exact visual state at
  time t. Must be pure — calling it twice with the same t gives the
  same result. No animation loop, no elapsed-time dependency.
- window.__renderReady   true once it's safe to capture.
- window.__SFX           the list of sound events with their times.

Force the first scene to start at t=0 so there is never a blank
opening frame.
```

```
When you're done, verify every data-anchor and data-cue in my scenes
actually resolves against the word list, and list any that don't.
```

PROMPT 6 · THE SCENES

```
Read PROJECT.md and the narration script, then build the index.html
for this video using shared/brand.css and shared/engine.js.

One scene per idea in the script. For each one, pick the visual that
actually teaches the point:
  a before/after comparison
  a diagram that builds piece by piece as it's described
  a number or stat callout
  a real screenshot of the thing being explained

Not decoration. If the visual is just "a relevant image," it's an
icon, not an illustration — replace it with something that argues
the idea on its own.

Anchor every scene to a phrase from the script. Inside diagrams,
anchor each individual node to the words that describe it, so the
diagram assembles as it's narrated instead of appearing whole.

Keep on-screen text to a headline plus one line. The voice carries
the rest.

If you need a screenshot I haven't given you, ask me for it rather
than drawing a fake version of it.

When it's done, screenshot each scene at its midpoint so I can check
the layout without rendering the whole video.
```

PROMPT 7 · THE RENDER

```
Build render.mjs. For a given video folder:

Capture:
- Launch headless Chromium via Playwright at 1920x1080,
  deviceScaleFactor 1, sRGB forced, font hinting off
- Set window.__FRAMESEEK = true before the page loads
- Wait for window.__renderReady AND document.fonts.ready
- Total length = voiceover duration + 1.4s tail pad
- At 30fps, for each frame i: call window.__renderAt(i/30), then
  screenshot to JPEG at quality 92
  (JPEG not PNG — roughly 6x faster, and the difference is invisible
  at this quality)

Mux with ffmpeg:
- Voice at full volume, padded to the total duration
- Music bed at volume 0.05, looped, fade in 1.5s, fade out 1.8s
- Duck the music under the voice with sidechaincompress:
  threshold 0.02, ratio 8, attack 20, release 400
- Each SFX event delayed to its exact millisecond with adelay, then
  mixed in. Gains around: whoosh 0.18, pop 0.14, thump 0.24
- Film treatment on the video: vignette=angle=PI/5 and a light
  grain, noise=alls=4:allf=t+u
- Encode libx264, preset medium, CRF 19, yuv420p, AAC 192k,
  +faststart
- Output to out/<slug>.mp4

Then two safety gates:
- Check the first second of the finished video is not static. If it
  is, FAIL the render loudly — a frozen opening frame is the most
  common silent failure.
- Do NOT copy the file anywhere on its own. Halt and tell me to
  review it. Only copy it to a delivery folder when I pass --deliver.

Ask me where I want finished videos delivered, and save that to
PROJECT.md. Also ask me whether I want a music bed at all, and if so
where my music file is.
```

PROMPT 8 · THE QA PASS — DO NOT SKIP THIS

```
Before I look at this video, check it yourself:

1. List every data-anchor and data-cue in the scene file and confirm
   each one resolves against the actual word list. Report any that
   don't — those elements will never appear.
2. Extract a frame every 2 seconds from the finished MP4 and actually
   look at each one. Flag: text running off frame, elements
   overlapping, anything unreadable at video size, any frame that is
   blank or nearly blank.
3. Confirm the audio runs the full length and doesn't cut off before
   the last word.
4. Confirm the first second has motion.

Report what you found. Do not tell me it's done until you've looked
at the frames.
```

---

03 The gotchas that will bite you

These are real failures from the real build. You will probably hit at least two.

The transcriber writes numbers as digits

Your script says "twenty to thirty a day." The transcript says "20 to 30." A slide anchored to the words "twenty to thirty" will never fire — silently, with no error. Anchor phrases have to match what the *transcriber* wrote, not what you wrote. Re-verify every anchor after any script change. The same applies to "OK" versus "okay," and it will not reliably preserve your pronunciation respellings.

The cue cursor trap

If the engine resolves cues with one advancing pointer through the word list, a single mismatch breaks every cue after it. Diagrams then appear all at once instead of building. Resolve each cue independently. This is in Prompt 5 for a reason.

Sizing for a browser instead of a frame

Type that looks right in a browser window is half-size on a video frame, and compression eats the rest. Check it at an actual 1920×1080 viewport and read it as if it were a video, not a webpage.

CSS selectors that reach too far

A rule like `.diagram svg` also hits the little icon SVGs inside your diagram's nodes and stacks them on top of your labels. Scope to direct children: `.diagram > svg`.

A frozen first frame

If your opening scene's entrance animation starts after t=0, the video opens on a still image. It looks broken and it's easy to miss. That's why the render fails loudly on it.

> **Why this can't drift out of sync**
>
> It never records a screen. For every output frame it computes the exact visual state at that timestamp and photographs it, taking as long as it needs. Output time always equals source time, so the picture cannot drift off the audio no matter how slow the machine is. That's the single biggest reliability difference versus screen-recording a playing animation.

---

04 Making the next video

This is the part that matters. The first video is real work — mostly the design system. After that:

YOU WANT TO…	DO THIS
Make a whole new lesson	Run Prompt 3 with the new transcript, then Prompts 6, 7, 8. The engine and the brand already exist.
Fix wording on a slide	Edit the scene file, screenshot it, re-render
Change the script	Edit both script files → re-run audio → re-run words → re-render. **Scene timing re-syncs itself.** Re-verify your anchors first.
Rebrand everything	Edit `shared/brand.css` → re-render every video

---

05 Be honest about what this isn't

- **It's not a talking head.** If the job is trust, put a person on camera.

- **It's not a software demo.** Sometimes people need to watch the real clicks happen.
- **The first build is real work.** Hours, mostly on the design system. The leverage arrives on video two.
- **It won't fix a bad explanation.** If whoever recorded it doesn't actually know the topic, you get a beautifully animated mess.

What it's genuinely great at is the course lesson in the middle — onboarding, training, "how this feature works." The video everyone agrees should exist and nobody ever makes.

Use any of this. No attribution needed, no email required, nothing to sign up for.

Built and documented by **NextLevel Digital** — we build the CRM and marketing systems that insurance agents and agencies run their lead flow on.
More of how we build things: **youtube.com/@nextleveldigtal**