The Same-Prompt Model Test Kit

The brief template, the six-point scoring sheet and the setup for running one video prompt through two Claude models.

Why I run this test before I trust a model

I gave Claude Fable 5.2 and Claude Opus 5.5 the exact same prompt and asked each one to build a short video. Both finished. Both looked like my brand. The two files were still clearly different videos, and that gap is the thing nobody warns you about when they tell you to just switch models.

The way most people compare models is to ask each one a question, read the two answers, and pick the one that sounds smarter. That tells you nothing about video. A short video has a hook, a pace, a caption track and a word count, and a model can write a beautiful paragraph while getting every one of those wrong. You only find out by building the same video twice and watching both.

So this is the kit I use. One brief written once, run through two models in two separate sessions, scored on six things that actually move watch time. It takes about twenty minutes and it ends the argument for your own content instead of someone else's.

01

How it actually works

1. You write the brief one time and save it to a file. Topic, length, style, voice, caption rules and the call to action all go in writing, because anything you leave out is exactly where the two models will quietly disagree.
2. You paste that file into a fresh session on model A, let it build, and save the output. Then you close the session, open a new one on model B, and paste the identical file. Separate sessions matter, since anything the first model learned would otherwise bleed into the second run.
3. You watch both videos against a fixed six-point sheet instead of going on gut feeling. The score tells you which model needs less of you, and that is the one worth keeping in a pipeline you run every week.

02

The one thing almost everyone gets wrong

They improve the brief between the two runs. The first model trips on something, you spot the hole in your wording, you patch it, and then you hand the better version to the second model. Now you have tested two prompts on two models and learned nothing about either one.

> **WATCH OUT**
>
> Write the brief down before the first run and do not touch it until both runs are finished. If the first build exposes a hole in your wording, note it on paper and fix it afterwards, then rerun both models on the corrected brief. One variable at a time or the test is decoration.

The second most common mistake is judging on one viewing with sound on. Most people on Instagram and TikTok see your video muted first. Watch both cuts muted before you watch them with audio, because a video that only works with the voiceover is a video that loses the scroll.

03

The prompts, in full

Three prompts do the whole test. Save the first as a file, since you will paste it twice and it has to be identical both times. Square brackets are the parts you replace.

BRIEF.MD

```
# Short video brief

TOPIC: [one sentence on what the video teaches]
AUDIENCE: [who this is for and what they already know]
LENGTH: [45–55 seconds, roughly 110–135 spoken words]
FORMAT: [avatar explainer | split A-roll and B-roll | fully faceless]
ASPECT: 9:16 vertical

HOOK:
- The first 2 seconds must state the payoff or the problem.
- No greeting, no channel intro, no "in this video".

VOICE:
- First person, plain sentences, uneven rhythm.
- No em dashes. No "not just X, it's Y". No hype adjectives.
- Concrete nouns over abstractions.

VISUALS:
- Render everything in code. No image generation.
- [number] scenes maximum.
- On-screen text only repeats words that are actually spoken.
- Never put a number, price or statistic on screen that the script does not say.

CAPTIONS:
- Word-level timing, aligned to the voiceover.
- Maximum [number] words visible at once.

CALL TO ACTION (last 5 seconds, spoken and on screen):
- [exact wording, including the keyword people comment]

DELIVER:
- The script first, as plain text, so I can read it before anything renders.
- Then the rendered vertical MP4.
- Then a one-line note on anything you had to guess.
```

The second prompt is the scoring pass. Run it in a third, clean session with both videos attached, so the model judging the work is not the model that made it.

> **SCORE BOTH CUTS**
>
> ```
> Here are two vertical shorts built from the same brief. Video A is [model A], video B
> is [model B]. Score each one from 1 to 5 on these six things, and give me a one-
> sentence reason per score:
>
> 1. Does the first 2 seconds state a payoff or a problem?
> 2. Scene count and pacing: static, right, or frantic?
> 3. Do the captions stay in sync with the voiceover the whole way?
> 4. Does the spoken word count land inside the brief's range?
> 5. Is there any number, claim or label on screen that the script never says?
> 6. Would a viewer who watches it muted understand the point?
>
> Finish with a table of the two totals and one line on which cut you would post. Do not
> be polite about it.
> ```

The third prompt is the one that turns the test into a better brief. This is where the real value is, because the fix outlives whichever model won.

> **TURN THE GAP INTO BRIEF TEXT**
>
> ```
> Both builds came from the brief I am pasting below. Here are the six scores for each
> cut: [paste the scores].
>
> For every score under 4, work out which sentence of the brief left room for the model
> to get it wrong. Then write me the replacement sentence, worded so the instruction is
> unambiguous. Give me only the changed lines, in the order they appear in the brief. Do
> not rewrite the parts that worked.
>
> [paste brief.md]
> ```

Set it up in Claude Code or Codex

1. Make a folder for the test and put brief.md in it. Anywhere is fine, a plain ~/model-test folder works. Keeping the brief as a file rather than scrollback is the whole point, since you need to paste it twice without a single character drifting.
2. If you want this as a reusable skill rather than a loose file, put it in ~/.claude/skills/same-prompt-test/ for Claude Code, with a SKILL.md inside. The folder name has to match the name: line in SKILL.md exactly or the agent will not load it.
3. On Codex the same folder lives in your project as .codex/skills/same-prompt-test/, and you add a line in AGENTS.md pointing at it so the agent knows it exists.
4. Open a terminal in the test folder and start the agent. Set the model before you say anything else: in Claude Code that is the /model command. Codex sets it from its own model picker. Confirm the model name is showing before you paste, because switching after the build starts contaminates the run.
5. Paste the contents of brief.md and nothing else. If the agent asks a clarifying question, answer it with the words already in the brief rather than new guidance, or reply once with "use your judgment and note the assumption at the end".
6. Let it write the script first and read that before anything renders. A script that is already off-brief will not be rescued by a good render, and catching it here saves you the render time.
7. Save the output as a-[model name].mp4. Write down how many follow-up prompts it took to get something you would post. That number is a score in its own right.
8. Quit the session completely. Start a new one, switch to the second model, and repeat from step five with the same file. Save as b-[model name].mp4.
9. Open a third clean session, attach both files, and run the scoring prompt. A successful run ends with two totals and a one-line verdict, and usually at least one score you disagree with, which is useful too.

If the build needs an API key for a voice or avatar service, do not paste the key into the chat. Ask the agent to add it to your shell profile for you, then close the terminal and open a new one so the key is loaded. A key set in an old window will not exist in the new session you start for model B, and that failure looks exactly like a model problem.

> 💡 **QUICK TIPS**
>
> - Run both models on the same day. Model behaviour shifts between releases, so a build from three weeks ago is not a fair opponent.
> - Use a topic you actually know well. You cannot spot an invented claim in a subject you are learning from the video.
> - Keep the two MP4s. When you repeat the test after the next model release you will want the old cuts to compare against.

04

The commands I actually run

START THE BUILD

```
Read brief.md in this folder and build it. Show me the script as plain text and stop there. Do not render anything until I say go.
```

CHECK THE SCRIPT AGAINST THE BRIEF

```
Before we render: count the spoken words in that script and tell me the number. Then list every line in brief.md that the script does not satisfy. If the list is empty, say so plainly instead of finding something to fix.
```

CATCH INVENTED FACTS

```
Go through the script and the on-screen text and flag every number, price, date, product name or statistic that I did not give you in brief.md. For each one, tell me where it came from. Remove anything you cannot source.
```

FIX WITHOUT A FULL REBUILD

```
Keep the script exactly as it is. Change only [the caption timing on scene 3 / the hook wording / the CTA on screen] and re-render. Tell me which files you touched.
```

LOG THE RUN

```
Append one line to results.md: the model name, the number of follow-up prompts this build needed, the final spoken word count, and anything you had to guess because brief.md did not say. Plain text, no commentary.
```

When it breaks

ERROR	FIX
The second model builds something totally different in shape	You almost certainly edited the brief between runs, or answered a clarifying question with new guidance. Reset to the saved brief.md and run both again.
One model keeps asking questions, the other just builds	That is a real difference and it belongs in your score, not in your workflow. Answer with words already in the brief, count the exchanges, and write the count down.
The script runs 40 percent over the word count	Your LENGTH line gave a duration but no word range. Give both, as in "45-55 seconds, roughly 110-135 spoken words", since models count words more reliably than they estimate seconds.
A number appears on screen that the script never says	Add the explicit line to the brief and rerun. Models fill visual slots with plausible figures when the brief asks for a stat card and the script has no stat.
Captions drift out of sync partway through	The caption track was timed against the script rather than the rendered audio. Ask for word timings taken from the finished voiceover file, not from the text.
The build fails on a missing API key in the second session only	The key was set in the first terminal window and never written to your shell profile. Add it to the profile, open a new terminal, and rerun that model.
Both videos look fine and you cannot pick	Watch both muted. The one you can still follow with no sound is the one to post, and if they are still tied, post both a week apart and let watch time decide.
You rerun the same model and get a different video	Expected. The model writes new code each run. Only the render step repeats exactly, so score a model on two runs rather than one if the first result surprises you.

Ready to go deeper?

Want the videos made for you? Paste any link or idea into Content Engine and get a published video in under 3 minutes, with DM automation armed to capture leads.

Meet Content Engine →

https://builds.digicuratoragency.com/content-engine/