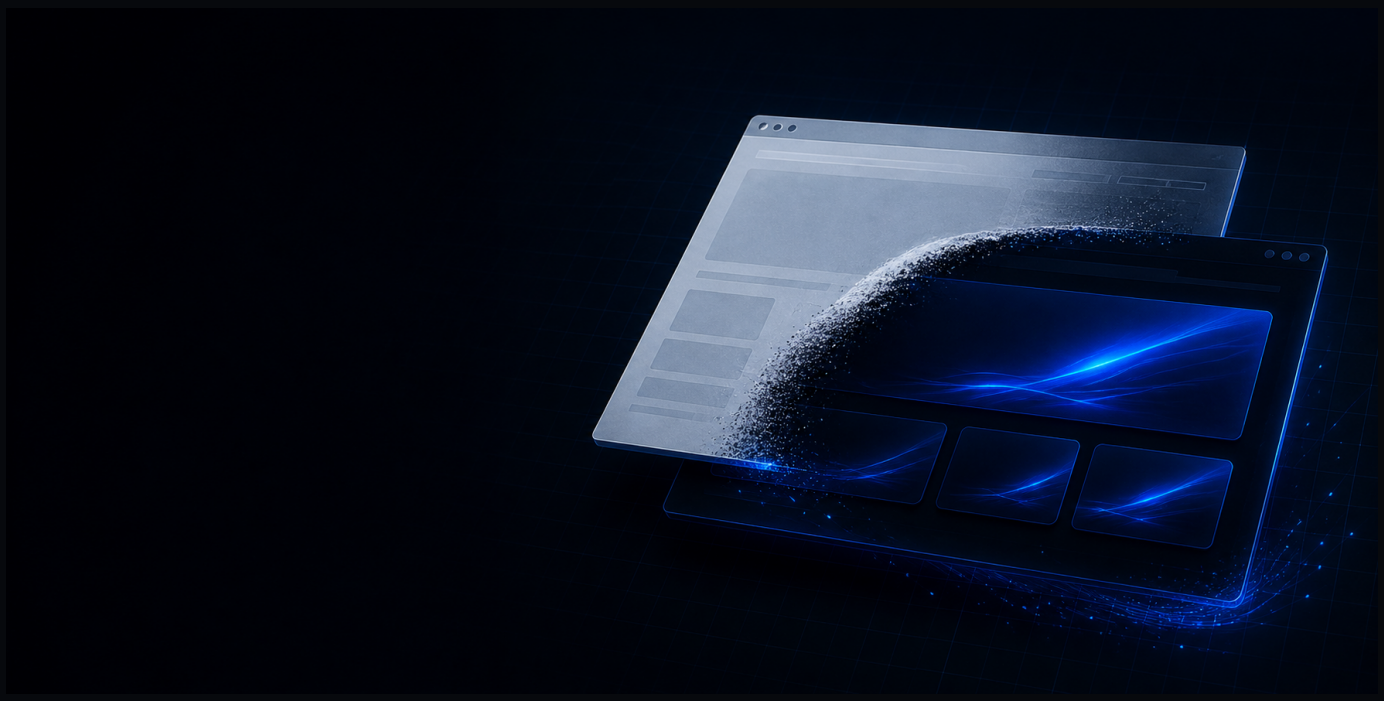

FREE BUILD GUIDE + AI AGENT PROMPT

Build a Layered Video Reveal Hero

Create the video underneath, paint the old website above it, and erase the surface with pointer movement and scroll.

ABRAHAM XIONG | ABRAHAMXIONG.COM

Inside this resource

Part One explains the effect from first principles. Part Two gives you the complete prompt to hand to a coding agent inside your own project.

PART ONE

The Human Build Guide

Understand the architecture, produce the underlying film, build the canvas cover, and verify the interaction across devices and motion preferences.

What you are building

The hero is two full-screen experiences stacked in the same position:

1. The bottom layer is the new world. It contains a looping film, the current brand, real page copy, and real links.
2. The top layer is the old world. It is drawn on a transparent HTML canvas and visually covers the new world.
3. Moving a pointer erases temporary holes in the canvas.
4. Scrolling expands a permanent circular wipe until the old canvas disappears.

The important idea is simple: do not try to reveal a video inside the canvas. Keep the video and the accessible page content in normal HTML. Use the canvas only as a removable cover.

```
Top:    canvas containing the old website
        erase pixels with destination-out
Bottom: new video + gradient + logo + CTA + semantic copy
```

The current Agenseo version paints a recognizable 2015 corporate template on top. Underneath it, a dark film holds the phrase "SOMETHING NEW" in changing physical materials. The pointer reveal is playful. The scroll reveal guarantees that every visitor can finish the transition.

What you need

- A short 16:9 source film for the new world
- A still poster taken from that film
- A web project that can render HTML, CSS, and client-side JavaScript
- An animation library with pinning support, or the dependency-free sticky technique below
- A real headline and CTA in the HTML under the canvas

The effect does not require WebGL. A 2D canvas is enough.

Part 1: Create the video underneath

Step 1: Lock the composition before adding motion

Start with a 1920 by 1080 or 1280 by 720 canvas. Use 24 or 30 frames per second. Build the composition so it can be cropped by `object-fit: cover` without losing the message.

Keep the following rules:

- Hold the camera still.
- Center the new message in the same area as the old message above it.
- Keep essential text inside the central 80 percent of the frame.

- Use a simple background so a partially erased hole remains readable.
- Let material, light, or surface treatment move. Do not move the whole phrase around the frame.
- Avoid rapid flashes or aggressive camera movement.

The Agenseo film uses a black background and a fixed two-line "SOMETHING NEW" composition. The letters change between inflated, fabric-like, chrome, and other tactile surfaces. That makes the lower layer feel alive without breaking the visual match with the old words above it.

Step 2: Choose how to produce the moving type

Use either of these production paths.

Controlled path

Create the words as real type in Blender, Cinema 4D, After Effects, Cavalry, or another motion tool. Add material changes, soft deformation, lighting changes, or displacement while keeping the camera and letter positions fixed.

This is the safest path when spelling and typography must remain perfect.

AI-assisted path

Generate a locked-camera material animation with an AI video tool, then composite real typography over it or use a clean typographic reference frame as the first-frame constraint. Generative video often changes letterforms and spelling, so inspect every frame before accepting the result.

A useful direction is:

```
Locked front-facing camera. A centered two-line typographic sculpture reading
"SOMETHING NEW" on a pure black background. The letter positions and spelling
remain completely fixed. Only the physical material changes slowly between
inflated rubber, soft fabric, liquid chrome, and glossy translucent glass.
Studio lighting, strong silhouette, no camera motion, no added objects,
no extra text, seamless slow movement, 16:9.
```

The production Agenseo film used generated material motion and was made loopable in post-production. The code does not depend on a particular video provider.

Step 3: Turn the source into a seamless palindrome loop

A normal generated clip often jumps when it returns to the first frame. A palindrome loop plays the clip forward and then in reverse, so its end joins its beginning naturally.

For a short silent source clip, use FFmpeg:

```
ffmpeg -i source.mp4 \
  -filter_complex
    "[0:v]fps=24,format=yuv420p,split=2[forward][reverse_input];[reverse_input]reverse[reverse];[forward][reverse]concat=n=2:v=1:
    \
  -map "[loop]" -an -c:v libx264 -crf 22 -preset slow \
  -movflags +faststart reveal-loop.mp4
```

This is best for clips under roughly 15 seconds because the reverse filter buffers frames. If the motion looks like an obvious rewind, use a crossfade loop or create a true seamless cycle in the motion tool instead.

Step 4: Create a poster frame

The poster appears before playback begins and becomes the complete reduced-motion experience.

```
ffmpeg -ss 1 -i reveal-loop.mp4 -frames:v 1 -q:v 2 reveal-poster.jpg
```

Choose a frame with the clearest message, good contrast, and no awkward transitional material.

Step 5: Compress and inspect the delivery file

Use H.264 with a `yuv420p` pixel format for broad browser support. Remove audio if it carries no meaning. Add `faststart` so playback can begin before the whole file downloads.

Check all of the following:

- The phrase stays legible at every frame.
- The loop has no visible jump.
- The crop works on a tall phone and a wide desktop.
- The poster matches the video closely enough to avoid a visual jump.
- The file is small enough for the project's performance budget.

The current Agenseo delivery is 1280 by 720, 24 fps, silent H.264, about 30 seconds after palindrome assembly, and about 1.9 MB.

Part 2: Build the bottom layer

Use normal HTML for the new world. The canvas will be decorative and removable, so the real message must live here.

```
<div class="reveal-story">
  <section class="reveal-hero">
    <div class="new-world">
      <video
        class="new-world__video"
        src="/media/reveal-loop.mp4"
        poster="/media/reveal-poster.jpg"
        muted
        loop
        playsinline
        preload="metadata"
        aria-hidden="true"
      ></video>

      <div class="new-world__shade" aria-hidden="true"></div>

      <div class="new-world__content">
        <nav aria-label="Hero navigation">
          <a href="/">Brand</a>
          <a href="/contact">Book a call</a>
        </nav>

        <h1 class="visually-hidden">
          From something old to something new. We rebuild dated websites into
          modern, interactive ones.
        </h1>

        <p>Your website was new once. We can do that again.</p>
        <p aria-hidden="true">Wipe the page or just scroll</p>
      </div>
    </div>

    <canvas class="old-world" aria-hidden="true"></canvas>
  </section>
</div>

.reveal-story {
  height: 260svh;
}

.reveal-hero {
  position: sticky;
  top: 0;
  height: 100svh;
  overflow: clip;
  background: #050a08;
```

```
  touch-action: pan-y;
}

.new-world,
.new-world__video,
.new-world__shade,
.old-world {
  position: absolute;
  inset: 0;
}

.new-world__video {
  width: 100%;
  height: 100%;
  object-fit: cover;
}

.new-world__shade {
  background: linear-gradient(
    to bottom,
    rgb(5 10 8 / 78%) 0%,
    transparent 20%,
    transparent 76%,
    rgb(5 10 8 / 80%) 100%
  );
}

.new-world__content {
  position: relative;
  z-index: 1;
  display: flex;
  min-height: 100%;
  flex-direction: column;
  justify-content: space-between;
  padding: 1.75rem clamp(1.25rem, 4vw, 2.5rem);
  color: white;
}

.old-world {
  z-index: 2;
  width: 100%;
  height: 100%;
  pointer-events: none;
}

.visually-hidden {
  position: absolute;
  width: 1px;
  height: 1px;
  overflow: hidden;
  clip: rect(0 0 0 0);
  white-space: nowrap;
}
```

The extra `160svh` in `.reveal-story` gives the visitor scroll distance while the hero remains sticky. The production Agenseo component gets the same behavior from GSAP ScrollTrigger with `pin: true`, `scrub: true`, and an end distance of `+=160%`.

For production, delay the video source until the hero is near the viewport and pause it when it leaves. An `IntersectionObserver` is enough. When `prefers-reduced-motion: reduce` matches, render the poster instead of playing the video.

Part 3: Paint the old layer on top

Step 1: Size the canvas correctly

Canvas has a CSS size and a drawing-buffer size. Match both, and cap the device pixel ratio at 2 to keep high-density screens sharp without wasting too much memory.

```
const dpr = Math.min(window.devicePixelRatio || 1, 2);

function sizeCanvas(canvas, context, stage) {
  const width = stage.clientWidth;
  const height = stage.clientHeight;

  canvas.width = width * dpr;
  canvas.height = height * dpr;
  canvas.style.width = `${width}px`;
  canvas.style.height = `${height}px`;
  context.setTransform(dpr, 0, 0, dpr, 0, 0);

  return { width, height };
}
```

Run it at startup, after fonts are ready, and on resize.

Step 2: Draw a believable old website

Paint the complete old page on every animation frame with the default `source-over` composite mode.

```
function paintOldWorld(context, width, height) {
  context.globalCompositeOperation = "source-over";
  context.clearRect(0, 0, width, height);

  context.fillStyle = "#ffffff";
  context.fillRect(0, 0, width, height);

  // Draw the old navigation, logo, headline, button, and feature columns.
  // Use responsive coordinates derived from width and height.
}
```

Do not make the old layer cartoonishly bad. The effect works when viewers recognize a site that once looked acceptable but now feels dated.

The current Agenseo version uses these cues:

- Flat white background
- Thin gray navigation rule
- Small blue square logo
- Uppercase navigation links
- Large "SOMETHING OLD" headline
- Rounded blue "LEARN MORE" button
- Three generic feature columns

Keep the old headline aligned with the new headline below. The transition then feels like one phrase changing material instead of two unrelated designs occupying the same space.

Step 3: Record a continuous brush trail

Store each brush mark as an object containing `x`, `y`, `radius`, and `strength`. Add intermediate points when the pointer moves quickly, or fast movement will leave dotted gaps.

```
const trail = [];
let lastPoint = null;

function spawnBrush(x, y, width, height) {
  const radius = Math.min(width, height) * 0.13;

  if (lastPoint) {
    const distance = Math.hypot(x - lastPoint.x, y - lastPoint.y);
    const steps = Math.min(20, Math.floor(distance / 9));
```

```
      for (let index = 1; index < steps; index += 1) {
        trail.push({
          x: lastPoint.x + ((x - lastPoint.x) * index) / steps,
          y: lastPoint.y + ((y - lastPoint.y) * index) / steps,
          radius,
          strength: 1,
        });
      }
    }

    trail.push({ x, y, radius, strength: 1 });
    if (trail.length > 420) trail.splice(0, trail.length - 420);
    lastPoint = { x, y };
  }
```

Listen for pointer movement. You may also support touch movement, but keep `touch-action: pan-y` so the interaction never traps vertical scrolling.

Step 4: Erase instead of painting the video

After drawing the old page, switch the canvas to `destination-out`. Anything painted in this mode removes alpha from the existing canvas. The normal HTML and video underneath become visible through those transparent pixels.

```
function eraseTrail(context, trail) {
  context.globalCompositeOperation = "destination-out";

  for (const point of trail) {
    const gradient = context.createRadialGradient(
      point.x,
      point.y,
      0,
      point.x,
      point.y,
      point.radius
    );

    gradient.addColorStop(0, `rgba(0, 0, 0, ${point.strength})`);
    gradient.addColorStop(0.72, `rgba(0, 0, 0, ${point.strength * 0.85})`);
    gradient.addColorStop(1, "rgba(0, 0, 0, 0)");

    context.fillStyle = gradient;
    context.beginPath();
    context.arc(point.x, point.y, point.radius, 0, Math.PI * 2);
    context.fill();
  }

  context.globalCompositeOperation = "source-over";
}
```

Soft radial gradients produce organic edges without an SVG goo filter. The Agenseo version tried the goo approach and removed it because it blurred the entire old page.

Step 5: Make pointer holes temporary

On every animation frame, reduce each brush point's strength. Remove it when its strength reaches zero.

```
for (let index = trail.length - 1; index >= 0; index -= 1) {
  trail[index].strength -= deltaMilliseconds / 2600;
  if (trail[index].strength <= 0) trail.splice(index, 1);
}
```

This causes the old surface to heal behind the pointer. The temporary interaction invites play without letting a random pointer path leave the page permanently half-erased.

Step 6: Add an optional automatic preview

If the visitor has not moved the pointer for about 900 milliseconds, move a smaller virtual brush through a slow looping path. Stop the automatic brush immediately when real input arrives.

The automatic preview teaches the interaction without requiring an instruction modal. Keep its radius much smaller than the user brush so it does not reveal the whole composition by itself.

Part 4: Make scroll complete the reveal

Pointer interaction is not enough. Phones, trackpads, keyboards, and impatient visitors all need a deterministic path through the hero.

Calculate progress from 0 to 1 while the sticky hero moves through its scroll story:

```
function getScrollProgress(story) {
  const rect = story.getBoundingClientRect();
  const distance = story.offsetHeight - window.innerHeight;
  return Math.min(1, Math.max(0, -rect.top / distance));
}
```

Use that progress to erase a permanent radial flood after erasing the temporary trail:

```
function eraseScrollFlood(context, progress, width, height, center) {
  if (progress <= 0.001) return;

  const maximumRadius = Math.hypot(width, height) * 1.1;
  const radius = Math.pow(progress, 1.5) * maximumRadius;
  const x = center?.x ?? width / 2;
  const y = center?.y ?? height / 2;

  const gradient = context.createRadialGradient(x, y, 0, x, y, radius);
  gradient.addColorStop(0, "rgba(0, 0, 0, 1)");
  gradient.addColorStop(0.86, "rgba(0, 0, 0, 1)");
  gradient.addColorStop(1, "rgba(0, 0, 0, 0)");

  context.globalCompositeOperation = "destination-out";
  context.fillStyle = gradient;
  context.beginPath();
  context.arc(x, y, radius, 0, Math.PI * 2);
  context.fill();
  context.globalCompositeOperation = "source-over";
}
```

Center the flood on the most recent pointer position when available. Otherwise, use the center of the hero. Ease it with `progress ** 1.5` so the reveal begins deliberately and finishes with momentum.

At about 98.5 percent progress, set the canvas opacity to zero. This avoids leaving tiny opaque corners after the flood visually finishes.

If the project already uses GSAP, the equivalent controller is:

```
ScrollTrigger.create({
  trigger: stage,
  start: "top top",
  end: "+=160%",
  pin: true,
  scrub: true,
  onUpdate: (self) => {
    progress = self.progress;
  },
});
```

In React, create a pinned ScrollTrigger in a layout effect and call `trigger.kill(true)` during cleanup. The `true` restores the DOM structure that pinning changed before React removes the component.

Part 5: Assemble the animation loop

The render order matters:

```
function frame(now) {
```

```
    const delta = now - lastFrameTime;
    lastFrameTime = now;

    updateAutomaticBrush(now, delta);
    decayTrail(delta);
    paintOldWorld(context, width, height);
    eraseTrail(context, trail);
    eraseScrollFlood(context, progress, width, height, pointer);

    canvas.style.opacity = progress > 0.985 ? "0" : "1";
    requestAnimationFrame(frame);
  }
```

Always restore `source-over` after erasing. If you forget, the next frame will erase the new drawing instead of repainting the old page.

Part 6: Accessibility and resilience

Treat these as part of the feature, not optional polish.

- Put the meaningful headline, supporting copy, and links in real HTML below the canvas.
- Mark the decorative video and canvas `aria-hidden="true"`.
- Do not put essential words only inside the generated video.
- Respect `prefers-reduced-motion`. Show the poster and skip the animated old cover.
- Preserve normal vertical touch scrolling.
- Keep CTAs keyboard accessible.
- With JavaScript disabled, let the bottom HTML and poster remain visible.
- Pause video outside the viewport.
- Recalculate canvas dimensions on resize and after fonts load.
- Remove listeners, observers, animation frames, and scroll controllers on unmount.

Part 7: Test the finished hero

Visual checks

- The old and new phrases occupy the same geometry.
- Pointer strokes have soft continuous edges with no dotted gaps.
- Temporary pointer holes heal smoothly.
- Scroll always reveals every corner.
- The video crop remains intentional on phone, tablet, laptop, and ultrawide layouts.
- The poster-to-video transition does not jump.

Interaction checks

- Mouse, trackpad, touch, keyboard scrolling, and reduced-motion mode all reach the new world.
- The page does not trap scrolling.
- Leaving and returning to the route does not produce console errors.
- Resizing does not blur or stretch the canvas.

- Hidden site navigation returns at the intended point if the hero temporarily replaces it.

Performance checks

- Cap the device pixel ratio at 2.
- Cap the trail length.
- Throttle or coalesce very frequent pointer events.
- Use passive listeners where no `preventDefault` call is required.
- Encode the film for the web and include a poster.
- Do not load or decode off-screen video unnecessarily.

How to teach it in the YouTube video

Explain the build in this order:

1. Show the finished interaction.
2. Turn off the canvas so viewers see the complete new HTML and video layer.
3. Turn the canvas back on and show the painted old website.
4. Demonstrate `destination-out` with one large circle.
5. Replace the circle with a continuous pointer trail.
6. Make trail points fade so the surface heals.
7. Connect scroll progress to the permanent radial flood.
8. Add the poster, reduced-motion behavior, and cleanup.
9. Run the mouse, mobile, resize, and reduced-motion checks.
10. Give viewers the companion agent prompt in `layered-video-reveal-hero-agent-prompt.md`.

The simplest line for the audience is: "The new website is always there. We draw the old website on a canvas above it, then erase the canvas."

PART TWO

The Copy-Paste Agent Prompt

Replace the bracketed inputs, open your coding agent at the root of your project, and paste the complete assignment.

Why this is a prompt, not a skill

A prompt is the best artifact for this YouTube tutorial. It is portable, visible, easy to customize, and works with an agent for one clearly scoped build. The viewer does not need to install anything or understand a skill registry.

Convert it into a skill only when a team will build this pattern repeatedly across multiple projects and needs shared scripts, reference assets, framework variants, or organization-specific quality gates. A reusable skill would then contain a concise `SKILL.md`, this build contract as a reference, and any tested media or validation scripts.

How to use it

1. Put `new-world.mp4` and `new-world-poster.jpg` in the target project.
2. Replace every bracketed input in the prompt.
3. Open the coding agent in the root of the project.
4. Paste the complete prompt.
5. Review the agent's implementation and verification evidence before shipping.

Copy-paste prompt

```
Build a production-ready layered video reveal hero in this existing web project.

Inputs

- Old-world phrase: [SOMETHING OLD]
- New-world phrase: [SOMETHING NEW]
- New-world video: [PATH OR URL]
- New-world poster: [PATH OR URL]
- Brand name or logo: [BRAND]
- Supporting copy: [COPY]
- Primary CTA label: [CTA LABEL]
- Primary CTA destination: [CTA URL]
- Old-world visual era: [OLD-WORLD VISUAL ERA, FOR EXAMPLE, A 2015 CORPORATE TEMPLATE]
- Preferred accent color: [COLOR]

Goal

Create a full-viewport hero with two layers in the same geometry. The lower
layer is the modern new world and contains a looping film plus real semantic
HTML. The upper layer is a dated website painted on a transparent 2D canvas.
Pointer movement temporarily erases soft holes in the canvas to reveal the
film. Scrolling permanently expands the reveal until the old canvas is gone.

Work autonomously within the project, but do not deploy, push, install paid
services, or change unrelated code.

Discovery requirements

1. Read the repository instructions and inspect the current framework,
   dependency versions, routing, styling conventions, accessibility patterns,
   test setup, and existing animation or media helpers.
2. If this is Next.js or another fast-moving framework, read the matching local
   version documentation before using framework APIs.
3. Inspect the working tree and preserve all existing user changes. Keep the
   implementation focused on the requested hero.
4. Reuse existing dependencies and utilities where practical. Do not add an
   animation package if a current dependency or a dependency-free sticky
   implementation can satisfy the contract.
5. Inspect the supplied video and poster dimensions, codec, duration, and crop.
   Report missing or unsuitable assets instead of inventing paths.

Implementation contract

1. Architecture
   - Use a full-viewport stage with the lower new-world layer and an absolute
     canvas above it.
   - Keep the new-world headline, copy, brand, and CTA in real HTML.
   - Treat the canvas and film as decorative for accessibility.
   - Do not render the film into the canvas. The canvas is only the old cover.

2. New-world layer
   - Render the supplied film as muted, looping, inline video with a poster.
   - Use object-cover behavior and intentional focal positioning.
   - Add gradients when needed to protect the readability of real HTML.
   - Delay loading or playback until the hero is near the viewport if the
     project has an established media helper or can support an
     IntersectionObserver cleanly.
   - Pause the film outside the viewport.
   - For prefers-reduced-motion, render the poster instead of animated video.

3. Old-world canvas
   - Paint a responsive, believable [OLD-WORLD VISUAL ERA] site with the
     old-world phrase, navigation, logo treatment, CTA, and era-appropriate
     supporting elements.
   - Make it recognizably dated, not broken or absurd.
   - Align the old-world phrase with the new-world phrase beneath it so the
     wipe feels like one message transforming.
   - Size the drawing buffer from the rendered stage size and cap device pixel
     ratio at 2.
   - Recalculate on resize and after required fonts are ready.
```

4. Pointer reveal
   - Repaint the old page with globalCompositeOperation set to source-over.
   - Erase with destination-out and soft radial gradients.
   - Record brush points with x, y, radius, and strength.
   - Interpolate points across fast pointer movement so the trail has no gaps.
   - Use a user-brush radius near 13 percent of the shorter viewport dimension as a starting point, then tune it visually.
   - Decay brush strength over roughly 2.6 seconds so pointer holes heal.
   - Cap the trail length near 420 points and coalesce very frequent input.
   - Optionally start a smaller automatic preview brush after about 900 ms of inactivity. Stop it immediately when real user input arrives.

5. Scroll completion
   - Provide roughly 160 viewport-percent of controlled scroll distance after the initial hero viewport.
   - Pin the hero with the project's established animation system, or use a sticky hero inside a taller scroll wrapper.
   - Convert scroll position to progress from 0 through 1.
   - After the temporary trail erasure, erase a permanent radial flood whose radius is approximately:

     ```
     progress ** 1.5 * Math.hypot(width, height) * 1.1
     ```

   - Center the flood on the most recent pointer position when available, and use the stage center otherwise.
   - Hide the canvas completely near 98.5 percent progress so no opaque corner remains.
   - Fade any scroll instruction early in the reveal.

6. React and lifecycle safety, when applicable
   - Create DOM-reparenting pin behavior in a layout-phase effect.
   - If using GSAP ScrollTrigger with pinning, revert the pin during cleanup, such as trigger.kill(true), before React mutates the DOM.
   - Cancel animation frames and remove every listener, observer, media query listener, and scroll controller on unmount.
   - Do not create hydration-dependent visible content that causes a first-paint mismatch. A poster-first render is acceptable.

7. Accessibility and resilience
   - Provide one meaningful page-level heading in semantic HTML.
   - Mark decorative canvas and video content aria-hidden.
   - Keep the CTA keyboard accessible.
   - Preserve vertical touch scrolling and do not trap the viewport.
   - When JavaScript is unavailable, leave the new-world HTML and poster usable.
   - When reduced motion is requested, skip the animated old cover and show a clear static new-world experience.
   - Do not rely on words inside the video as the only source of meaning.

8. Visual and content rules
   - Adapt the pattern to this project's brand. Do not copy Agenseo colors, claims, name, or assets unless they are explicitly supplied in the inputs.
   - Do not invent company claims, statistics, testimonials, clients, or past performance.
   - Keep the old design's content strategically dated. Avoid making it praise the same qualities the new design is trying to sell.
   - Keep text contrast readable in the lower layer.

Verification requirements

1. Run the project's relevant formatter, lint, typecheck, component tests, browser tests, and production build in proportion to the change.
2. Add or update focused tests for:
   - the hero route rendering successfully
   - semantic headline and CTA presence
   - supplied media and poster paths
   - reduced-motion behavior
   - pointer and scroll completion where the test environment supports them
   - safe route entry and exit with no client-side console error
3. Manually inspect desktop, phone, reduced-motion, and resized layouts when a browser tool is available.
4. Confirm that scrolling can always complete the reveal without a pointer.
5. Confirm that the canvas remains sharp and the loop crop stays intentional.

```
Deliverables

- The working hero implementation integrated through existing project patterns
- Any focused tests needed for the new behavior
- A concise handoff naming changed files, architectural choices, checks run,
  results, limitations, and any remaining owner decision

Do not stop after writing a plan. Implement and verify the feature. If a
required media asset is genuinely missing, finish every safe part that does
not depend on it and report the exact missing input.
```

Optional follow-up prompt

After the first build, use this shorter review request:

```
Review the layered reveal hero you just built as a critical frontend lead.
Test the visual alignment of the old and new phrases, pointer trail continuity,
scroll completion, touch scrolling, reduced-motion behavior, video loading,
resize behavior, accessibility, cleanup on route exit, and performance. Fix
validated defects within scope, rerun the relevant checks, and report evidence.
```

When to turn it into a skill

Create a dedicated skill later if at least one of these becomes true:

- The pattern is being built across several repositories.
- The same canvas engine is copied more than once.
- The team needs framework-specific variants.
- The build requires reusable video-encoding or screenshot-validation scripts.
- Brand rules and acceptance checks must load automatically whenever the task is requested.

At that point, name the skill something action-oriented such as `build-layered-reveal-hero`. Keep the core `SKILL.md` concise. Put the detailed build contract in one reference file, and put deterministic media or validation commands in tested scripts.