Verbatim Extract – Preview Window (grouped)

Source file: index Hero refresh good.html

HTML

```
<div class="preview-overlay" aria-hidden="true">
  <div class="preview-card" role="dialog" aria-modal="true" aria-labelledby="preview-title">
    <button class="preview-close-btn" type="button" aria-label="Close preview">×</button>
    <div class="preview-media">
      <!-- Later: video or wide poster goes here -->
      <div class="preview-media-placeholder"></div>
      <img class="preview-media-poster" src="images/poster-wide-A1.png" alt="Wide poster for preview">
    </div>
    <div class="preview-info">
      <header class="preview-header">
        <h2 id="preview-title" class="preview-title">The Unreasonably Long and Impossible-to-Skip Title That Spills Acros
        <div class="preview-meta">
          <span class="preview-runtime">2h 14m</span>
          <span class="preview-tags">Prestige Drama • Mythic Noir • Alt■History • Neo■Western • Psychological Thriller •
        </div>
      </header>
      <div class="preview-body">
        <p class="preview-description">
          This is an intentionally overlong description designed to push the limits of the preview body container, wrappi
        </p>
      </div>
      <div class="preview-actions">
        <button class="preview-play-button" type="button">
          Play
        </button>
        <button class="preview-details-button" type="button">
          Details
        </button>
        <div class="preview-volume" aria-label="Preview volume control">
          <button class="preview-volume__btn" type="button" aria-label="Preview volume">
            <span aria-hidden="true">■</span>
          </button>
          <div class="preview-volume__sliderWrap">
            <input class="preview-volume__slider" type="range" min="0" max="100" value="100" aria-label="Preview volume s
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

CSS

```
:root { --bg:#000; --fg:#eaeaf0; --muted:#9aa0a6; --accent:#7aa2ff; --nav-height:64px; --brand:#e31b23; --brand-glow-shadow

.preview-morph-tile {
  position: fixed;
  top: 0;
  left: 0;
  transform-origin: top left;
  pointer-events: none;
  z-index: 11000;
  background: #05060a;
  overflow: hidden;
  border: var(--preview-frame-border);
  border-radius: var(--preview-frame-radius);
  box-shadow: var(--preview-frame-shadow);
  transition:
    transform var(--preview-morph-duration) var(--preview-morph-ease),
    width var(--preview-morph-duration) var(--preview-morph-ease),
    height var(--preview-morph-duration) var(--preview-morph-ease),
    opacity 0.24s ease,
    border-radius var(--preview-morph-duration) var(--preview-morph-ease),
    box-shadow var(--preview-morph-duration) var(--preview-morph-ease),
    border-color var(--preview-morph-duration) var(--preview-morph-ease);
  will-change: transform, opacity, width, height, border-radius;
  backface-visibility: hidden;
  transform: translate3d(0,0,0);
}

.preview-morph-fill {
  position: absolute;
  inset: 0;
  background: #000;
  background-size: cover;
  background-position: center;
  background-repeat: no-repeat;
  opacity: 1;
  transition: opacity var(--preview-morph-duration) var(--preview-morph-ease);
  backface-visibility: hidden;
}

.preview-morph-tile.is-fade-out {
  opacity: 0;
}

.preview-morph-fill.is-fade-out {
  opacity: 0;
}

  .preview-play-button,
  .preview-details-button {
    font-size: 0.95rem;
    padding: 0.50rem 0.80rem;
  }

  .preview-overlay {
    position: fixed;
    inset: 0;
    display: flex;
    align-items: center;
    justify-content: center;
    background: radial-gradient(circle at top, rgba(0,0,0,0.80), rgba(0,0,0,0.95));
    z-index: 11000;
    padding: 16px;
    opacity: 0;
    visibility: hidden;
    pointer-events: none;
    transition: opacity var(--preview-morph-duration) var(--preview-morph-ease), visibility 0s linear var(--preview-morph-d
    overscroll-behavior: contain;
  }

  .preview-overlay.is-visible {
    z-index: 11000;
    opacity: 1;
    visibility: visible;
    pointer-events: auto;
    transition: opacity var(--preview-morph-duration) var(--preview-morph-ease);
  }

  .preview-card {
    width: 100%;
    max-width: 960px;
```

```
  max-height: 90vh;
  background: #05060a;
  border-radius: var(--preview-frame-radius);
  border: var(--preview-frame-border);
  box-shadow: var(--preview-frame-shadow);
  overflow: hidden;
  display: grid;
  grid-template-rows: auto 1fr;
  position: relative;
  opacity: 0;
  transition:
    transform var(--preview-morph-duration) var(--preview-morph-ease),
    opacity var(--preview-morph-duration) var(--preview-morph-ease);
}

.preview-card.is-active {
  opacity: 1;
}

.preview-card.is-hidden {
  opacity: 0;
  pointer-events: none;
}

.preview-close-btn {
  position: absolute;
  top: 12px;
  right: 12px;
  width: 36px;
  height: 36px;
  display: inline-grid;
  place-items: center;
  border-radius: 50%;
  border: 1px solid rgba(255,255,255,0.18);
  background: rgba(0,0,0,0.55);
  color: #fff;
  font-size: 1.1rem;
  font-weight: 700;
  cursor: pointer;
}

.preview-close-btn:hover {
  background: rgba(255,255,255,0.10);
}

.preview-close-btn {
  touch-action: manipulation;
}

.preview-media {
  position: relative;
  background: #000;
  aspect-ratio: 16 / 9;
}

.preview-media-placeholder {
  position: absolute;
  inset: 0;
  z-index: 0;
  background:
    radial-gradient(circle at top, rgba(255,255,255,0.10), transparent 55%),
    linear-gradient(135deg, #222632, #11131a);
}

.preview-media-poster {
  position: absolute;
  inset: 0;
  width: 100%;
  height: 100%;
  object-fit: cover;
  opacity: 1;
  z-index: 1;
}

.preview-teaser-player {
  position: absolute;
  inset: 0;
  width: 100%;
  height: 100%;
  z-index: 2;
  opacity: 1;
  transition: opacity 700ms ease;
}
.preview-teaser-player iframe {
```

```
  width: 100%;
  height: 100%;
  border: 0;
}

.preview-teaser-player.teaser-fadeout {
  opacity: 0;
}

.preview-info {
  padding: 16px 18px 18px 18px;
  display: grid;
  grid-template-rows: auto 1fr auto;
  gap: 12px;
}

.preview-header {
  display: flex;
  flex-direction: column;
  gap: 4px;
}

.preview-title {
  font-family: var(--font-display);
  font-size: 1.2rem;
  font-weight: 800;
  letter-spacing: 0.03em;
  text-transform: uppercase;
  margin: 0;
}

.preview-meta {
  display: flex;
  flex-wrap: wrap;
  gap: 8px;
  font-size: 0.78rem;
  color: var(--muted);
}

.preview-runtime {
  padding: 2px 8px;
  border-radius: 999px;
  background: rgba(255,255,255,0.08);
  border: 1px solid rgba(255,255,255,0.18);
  font-weight: 600;
}

.preview-tags {
  opacity: 0.9;
}

.preview-body {
  font-size: 0.88rem;
  color: var(--muted);
  overflow: auto;
}

.preview-description {
  margin: 0;
}

.preview-actions {
  display: flex;
  flex-wrap: wrap;
  gap: 10px;
  justify-content: flex-start;
  padding: 0 6px;
}

.preview-play-button,
.preview-details-button {
  appearance: none;
  border-radius: 999px;
  padding: 0.42rem 0.56rem;
  font: 700 0.8rem/1 var(--font-ui);
  cursor: pointer;
  border: 0;
  transition: transform 0.12s ease, opacity 0.12s ease, box-shadow 0.12s ease, background 0.12s ease;
  touch-action: manipulation;
  box-shadow: 0 6px 14px rgba(0,0,0,.18);
}

.preview-play-button {
  background: #fff;
```

```
  color: #000;
}

.preview-play-button:hover {
  background: #e6e6e6;
}

.preview-details-button {
  background: rgba(255,255,255,0.10);
  color: #eaeaf0;
  box-shadow: inset 0 0 0 1.25px rgba(255,255,255,0.28);
}

.preview-details-button:hover {
  box-shadow: inset 0 0 0 1.25px rgba(255,255,255,0.5);
}

.preview-volume {
  --volume-pill-grow: 118px;
  display: inline-flex;
  align-items: center;
  gap: 6px;
  color: #eaeaf0;
  position: relative;
  z-index: 2;
  isolation: isolate;
  flex: 0 0 auto;
}

.preview-volume::before {
  content: "";
  position: absolute;
  left: 0;
  top: 0;
  height: 100%;
  width: 100%;
  background: #121212;
  border-radius: 999px;
  transition: width 0.2s ease, box-shadow 0.2s ease;
  pointer-events: none;
  z-index: 0;
}

.preview-volume__btn {
  appearance: none;
  width: auto;
  padding: 0.42rem 0.56rem;
  border-radius: 999px;
  border: none;
  background: transparent;
  color: #eaeaf0;
  display: inline-flex;
  align-items: center;
  justify-content: center;
  font: 700 0.8rem/1 var(--font-ui);
  cursor: pointer;
  touch-action: manipulation;
  position: relative;
  z-index: 1;
  isolation: isolate;
  overflow: visible;
}

.preview-volume__btn:focus {
  outline: none;
}

.preview-volume__btn:focus-visible {
  outline: none;
}

.preview-volume.is-muted::before {
  box-shadow: 0 0 0 2px rgba(255, 42, 42, 0.85), 0 0 12px rgba(255, 42, 42, 0.9);
}

.preview-volume.is-muted {
  --saber-color: 255 42 42;
}

.preview-volume.is-muted:focus-within::before {
  box-shadow: 0 0 0 3px rgba(255, 42, 42, 0.85), 0 0 16px rgba(255, 42, 42, 0.95);
}

.preview-volume.is-unmuted::before {
```

```
  box-shadow: 0 0 0 2px rgba(57, 255, 20, 0.9), 0 0 14px rgba(57, 255, 20, 0.95);
}

.preview-volume.is-unmuted {
  --saber-color: 57 255 20;
}

.preview-volume.is-unmuted:focus-within::before {
  box-shadow: 0 0 0 3px rgba(57, 255, 20, 0.9), 0 0 18px rgba(57, 255, 20, 0.95);
}

.preview-volume__sliderWrap {
  position: absolute;
  top: 50%;
  left: 100%;
  margin-left: 0;
  padding-left: 8px;
  transform: translateY(-50%) scale(0.96);
  display: flex;
  align-items: center;
  width: 100px;
  opacity: 0;
  overflow: visible;
  pointer-events: none;
  transition: opacity 0.2s ease, transform 0.2s ease;
  z-index: 1;
}

.preview-volume__slider {
  width: 90px;
  appearance: none;
  background: transparent;
}

.preview-volume__slider {
  height: 16px;
}

.preview-volume__slider::-webkit-slider-runnable-track {
  height: 6px;
  border-radius: 999px;
  background: rgb(var(--saber-color, 57 255 20));
  box-shadow:
    0 0 6px rgb(var(--saber-color, 57 255 20) / 0.8),
    0 0 12px rgb(var(--saber-color, 57 255 20) / 0.7);
}

.preview-volume__slider::-webkit-slider-thumb {
  -webkit-appearance: none;
  width: 14px;
  height: 14px;
  border-radius: 999px;
  background: #0d0d0d;
  border: 2px solid rgb(var(--saber-color, 57 255 20));
  box-shadow:
    0 0 6px rgb(var(--saber-color, 57 255 20) / 0.8),
    0 0 12px rgb(var(--saber-color, 57 255 20) / 0.6);
  margin-top: -4px;
}

.preview-volume__slider::-moz-range-track {
  height: 6px;
  border-radius: 999px;
  background: rgb(var(--saber-color, 57 255 20));
  box-shadow:
    0 0 6px rgb(var(--saber-color, 57 255 20) / 0.8),
    0 0 12px rgb(var(--saber-color, 57 255 20) / 0.7);
}

.preview-volume__slider::-moz-range-thumb {
  width: 14px;
  height: 14px;
  border-radius: 999px;
  background: #0d0d0d;
  border: 2px solid rgb(var(--saber-color, 57 255 20));
  box-shadow:
    0 0 6px rgb(var(--saber-color, 57 255 20) / 0.8),
    0 0 12px rgb(var(--saber-color, 57 255 20) / 0.6);
}

  .preview-volume:hover::before {
    width: calc(100% + var(--volume-pill-grow));
  }
  .preview-volume:hover .preview-volume__sliderWrap {
```

```
    opacity: 1;
    pointer-events: auto;
    transform: translateY(-50%) scale(1);
  }

.preview-volume.is-open::before,
.preview-volume:focus-within::before {
  width: calc(100% + var(--volume-pill-grow));
}

.preview-volume.is-open .preview-volume__sliderWrap,
.preview-volume:focus-within .preview-volume__sliderWrap {
  opacity: 1;
  pointer-events: auto;
  transform: translateY(-50%) scale(1);
}

  .preview-volume__sliderWrap {
    display: none;
  }

  .preview-volume.is-open::before,
  .preview-volume:focus-within::before {
    width: 100%;
  }

  .preview-card {
    max-width: 100%;
    border-radius: var(--preview-frame-radius);
    max-height: 100vh;
  }

  .preview-info {
    padding: 14px 14px 16px 14px;
  }

  .preview-card {
    max-height: calc(100vh - 32px);
  }

.poster-preview-button {
  position: absolute;
  top: 10px;
  left: 50%;
  transform: translateX(-50%);
  padding: 6px 14px;
  font-size: 0.75rem;
  font-family: var(--font-ui);
  font-weight: 700;
  text-transform: uppercase;
  border-radius: 999px;
  background: rgba(255,255,255,0.12);
  border: 1px solid rgba(255,255,255,0.25);
  color: #fff;
  backdrop-filter: blur(6px);
  cursor: pointer;
  transition: opacity 0.25s ease, transform 0.25s ease;
  opacity: 1;            /* always visible for now */
  pointer-events: auto;  /* clickable */
}

.poster-preview-button:hover {
  background: rgba(255,255,255,0.20);
  transform: translateX(-50%) scale(1.05);
}
```

JS

```
const previewOverlay = document.querySelector('.preview-overlay');
    if (previewOverlay && previewOverlay.classList.contains('is-visible')) {
      previewOverlay.setAttribute('aria-hidden', 'false');
      const previewPlayButton = previewOverlay.querySelector('.preview-play-button');
      const previewCloseButton = previewOverlay.querySelector('.preview-close-btn');
      const focusTarget = previewPlayButton || previewCloseButton;
      if (focusTarget && typeof focusTarget.focus === 'function') {
        focusTarget.focus();
      }
    }


const previewOverlay = document.querySelector('.preview-overlay');
      const previewCloseButton = document.querySelector('.preview-close-btn');

      const previewTitleEl = document.querySelector('.preview-title');
      const previewRuntimeEl = document.querySelector('.preview-runtime');

      const previewTagsEl = document.querySelector('.preview-tags');
      const previewDescriptionEl = document.querySelector('.preview-description');

        !previewOverlay ||
        !previewCloseButton ||
        !previewTitleEl ||
        !previewRuntimeEl ||
        !previewTagsEl ||
        !previewDescriptionEl
      ) {
        return;
      }

      const previewMedia = previewOverlay.querySelector('.preview-media');
      const previewPlaceholder = previewMedia ? previewMedia.querySelector('.preview-media-placeholder') : null;

      const previewPosterImage = previewMedia ? previewMedia.querySelector('.preview-media-poster') : null;
      const A1_PREVIEW_LABEL = 'A1';

      let previewOpening = false;

      function isPreviewOpen() {
        return previewOpening || previewOverlay.classList.contains('is-visible');
      }

      let previewTeaserPlayer = null;
      let previewTeaserTimer = null;

      let previewTeaserFadeTimeout = null;
      let previewTeaserFadeInterval = null;

      let previewTeaserContainer = null;
      let previewAutounmuteHandler = null;

      let previewAutounmuteRoot = null;
      let previewVolume = 100;

      let previewMuted = false;
      let initialPreviewVolume = previewVolume;

      let desiredPreviewVolume = previewVolume;
      let previewLastNonZeroVolume = previewVolume || 100;

      const previewVolumeControl = previewOverlay.querySelector('.preview-volume');
      const previewVolumeButton = previewOverlay.querySelector('.preview-volume__btn');

      const previewVolumeSlider = previewOverlay.querySelector('.preview-volume__slider');
      const previewVolumeIdleCloseDelay = 1500;

      let previewVolumeCloseTimeout = null;
      let previewVolumeCloseReason = null;

      let previewVolumeInitialTimeout = null;

      function pvLog(message, details = {}) {
        console.log('[PREVIEW-VOL]', message, details);
      }

      function openPreviewVolume() {
        if (!previewVolumeControl) return;
        previewVolumeControl.classList.add('is-open');
      }
```

```
  if (previewVolumeInitialTimeout) {
    window.clearTimeout(previewVolumeInitialTimeout);
    previewVolumeInitialTimeout = null;
  }

  if (!previewVolumeControl) return;
  pvLog('idle-close:call', { reason });

  if (previewVolumeCloseTimeout) {
    pvLog('idle-close:clear', { id: previewVolumeCloseTimeout, reason });
    window.clearTimeout(previewVolumeCloseTimeout);
  }

  previewVolumeCloseReason = reason;
  previewVolumeCloseTimeout = window.setTimeout(() => {
    const activeElement = document.activeElement;
    const containsActive = previewVolumeControl.contains(activeElement);
    pvLog('idle-close:fire', {
      reason,
      id: previewVolumeCloseTimeout,
      activeElement: activeElement
        ? `${activeElement.tagName.toLowerCase()}${activeElement.className ? `.${String(activeElement.className).trim
        : null,
      containsActive
    });
    if (containsActive && activeElement && activeElement.matches('.preview-volume__slider[type="range"]')) {
      activeElement.blur();
      pvLog('idle-close:blur', { target: 'preview-volume__slider' });
    }
    if (containsActive && activeElement && activeElement.matches('.preview-volume__btn')) {
      activeElement.blur();
      pvLog('idle-close:blur', { target: 'preview-volume__btn' });
    }
    previewVolumeControl.classList.remove('is-open');
    pvLog('idle-close:remove-open', { reason });
    previewVolumeCloseReason = null;
  }, previewVolumeIdleCloseDelay);

  pvLog('idle-close:set', { delay: previewVolumeIdleCloseDelay, id: previewVolumeCloseTimeout, reason });
}

  if (previewVolumeCloseTimeout) {
    window.clearTimeout(previewVolumeCloseTimeout);
    previewVolumeCloseTimeout = null;
  }

  previewVolumeCloseReason = null;
}

function clampPreviewVolume(value) {
  return Math.max(0, Math.min(100, value));
}

      rootEl: previewOverlay,
      playerGetter: () => previewTeaserPlayer,
      desiredVolume
    });

  if (!previewAutounmuteRoot || !previewAutounmuteHandler) return;
  previewAutounmuteRoot.removeEventListener('pointerdown', previewAutounmuteHandler);

  previewAutounmuteRoot = null;
  previewAutounmuteHandler = null;

  previewAutounmuteRoot = rootEl;
  previewAutounmuteHandler = () => {
    const player = playerGetter();
    attemptPreviewUnmute(player, desiredVolume);
    removePreviewAutounmuteListener();
  };

  previewAutounmuteRoot.addEventListener('pointerdown', previewAutounmuteHandler, { once: true });
}

  if (previewVolumeControl) {
    previewVolumeControl.classList.toggle('is-muted', isMuted);
    previewVolumeControl.classList.toggle('is-unmuted', !isMuted);
  }

  if (previewVolumeButton) {
    const iconTarget = previewVolumeButton.querySelector('span') || previewVolumeButton;
    iconTarget.textContent = isMuted ? '■' : '■';
    previewVolumeButton.setAttribute('aria-pressed', isMuted ? 'true' : 'false');
  }
```

```
  previewVolume = clampPreviewVolume(nextVolume);
  if (previewVolume > 0) {
    previewLastNonZeroVolume = previewVolume;
    previewMuted = false;
  } else {
    previewMuted = true;
  }

  if (previewVolumeSlider) {
    previewVolumeSlider.value = String(previewVolume);
  }

  if (previewTeaserPlayer) {
    applyPreviewVolumeToPlayer(previewTeaserPlayer, previewVolume, previewMuted);
  }

  updatePreviewVolumeUI({ muted: previewMuted, volume: previewVolume });
}

  const playerMuted = previewTeaserPlayer && typeof previewTeaserPlayer.isMuted === 'function'
    ? previewTeaserPlayer.isMuted()
    : previewMuted;

  if (playerMuted || previewVolume === 0) {
    previewMuted = false;
    if (previewVolume === 0) {
      previewVolume = previewLastNonZeroVolume || 50;
    }
    if (previewVolumeSlider) {
      previewVolumeSlider.value = String(previewVolume);
    }
    if (previewTeaserPlayer && typeof previewTeaserPlayer.unMute === 'function') {
      previewTeaserPlayer.unMute();
    }
    if (previewTeaserPlayer && typeof previewTeaserPlayer.setVolume === 'function') {
      previewTeaserPlayer.setVolume(previewVolume);
    }
  } else {
    previewMuted = true;
    if (previewTeaserPlayer && typeof previewTeaserPlayer.mute === 'function') {
      previewTeaserPlayer.mute();
    }
  }

  updatePreviewVolumeUI({ muted: previewMuted, volume: previewVolume });
}

if (previewVolumeSlider) {
  let previewSliderPointerActive = false;
  previewVolumeSlider.value = String(previewVolume);
  previewVolumeSlider.addEventListener('input', (event) => {
    const value = Number(event.target.value);
    if (!Number.isNaN(value)) {
      setPreviewVolume(value);
    }
    clearPreviewVolumeInitialClose();
    openPreviewVolume();
    resetPreviewVolumeIdleClose('slider');
  });
  previewVolumeSlider.addEventListener('change', (event) => {
    const value = Number(event.target.value);
    if (!Number.isNaN(value)) {
      setPreviewVolume(value);
    }
    if (previewSliderPointerActive) {
      const slider = previewVolumeSlider;
      window.setTimeout(() => {
        slider.blur();
      }, 0);
    }
    clearPreviewVolumeInitialClose();
    openPreviewVolume();
    resetPreviewVolumeIdleClose('slider');
  });
  previewVolumeSlider.addEventListener('pointerdown', () => {
    previewSliderPointerActive = true;
  });
  previewVolumeSlider.addEventListener('pointerup', () => {
    const slider = previewVolumeSlider;
    window.setTimeout(() => {
      slider.blur();
    }, 0);
    previewSliderPointerActive = false;
  });
```

```
  previewVolumeSlider.addEventListener('pointercancel', () => {
    const slider = previewVolumeSlider;
    window.setTimeout(() => {
      slider.blur();
    }, 0);
    previewSliderPointerActive = false;
  });
}

updatePreviewVolumeUI({ muted: previewMuted, volume: previewVolume });

if (previewVolumeControl) {
  previewVolumeControl.addEventListener('wheel', (event) => {
    event.preventDefault();
    const delta = event.deltaY || event.deltaX || 0;
    if (delta === 0) return;
    const step = delta > 0 ? -5 : 5;
    setPreviewVolume(previewVolume + step);
    clearPreviewVolumeInitialClose();
    openPreviewVolume();
    resetPreviewVolumeIdleClose('slider');
  }, { passive: false });
  previewVolumeControl.addEventListener('pointerenter', () => {
    clearPreviewVolumeInitialClose();
    openPreviewVolume();
    if (previewVolumeCloseTimeout && previewVolumeCloseReason === 'leave') {
      window.clearTimeout(previewVolumeCloseTimeout);
      previewVolumeCloseTimeout = null;
      previewVolumeCloseReason = null;
    }
  });
  previewVolumeControl.addEventListener('pointerleave', () => {
    resetPreviewVolumeIdleClose('leave');
  });
}

if (previewVolumeButton) {
  previewVolumeButton.addEventListener('click', () => {
    togglePreviewMute();
    clearPreviewVolumeInitialClose();
    resetPreviewVolumeIdleClose('button');
  });
}

  if (!previewPlaceholder) return;
  previewPlaceholder.style.backgroundImage = '';

  previewPlaceholder.style.backgroundSize = '';
  previewPlaceholder.style.backgroundPosition = '';

  previewPlaceholder.style.opacity = '';
  previewPlaceholder.style.transition = '';

  if (!previewPosterImage) return;
  previewPosterImage.src = '';

  previewPosterImage.alt = '';
  previewPosterImage.style.opacity = '0';

  if (!previewPosterImage || !card) return;
  const posterSrc = card.getAttribute('data-poster-vert') || '';

  previewPosterImage.src = posterSrc;
  previewPosterImage.alt = `${card.getAttribute('data-label') || 'Preview'} poster`;

  previewPosterImage.style.opacity = '1';
}

function isA1PreviewCard(card) {
  return !!(card && card.getAttribute('data-label') === A1_PREVIEW_LABEL);
}

  if (!previewPlaceholder) return;
  previewPlaceholder.style.transition = '';

  previewPlaceholder.style.opacity = '0';
}

function clearPreviewTeaserTimer() {
  if (previewTeaserTimer !== null) {
    clearTimeout(previewTeaserTimer);
    previewTeaserTimer = null;
  }
}
```

```
  if (previewTeaserFadeTimeout !== null) {
    clearTimeout(previewTeaserFadeTimeout);
    previewTeaserFadeTimeout = null;
  }

  if (previewTeaserFadeInterval !== null) {
    clearInterval(previewTeaserFadeInterval);
    previewTeaserFadeInterval = null;
  }

  const current = typeof player.getVolume === 'function' ? player.getVolume() : previewVolume;
  const startVolume = clampPreviewVolume(Number.isFinite(current) ? current : previewVolume);

  previewTeaserFadeInterval = setInterval(() => {
    step += 1;
    const nextVolume = clampPreviewVolume(Math.round(startVolume - (volumeStep * step)));
    player.setVolume(nextVolume);
    if (nextVolume <= 0 || step >= totalSteps) {
      clearPreviewTeaserFadeInterval();
    }
  }, PREVIEW_TEASER_AUDIO_STEP_MS);

  if (previewTeaserPlayer) {
    try {
      if (typeof previewTeaserPlayer.stopVideo === 'function') {
        previewTeaserPlayer.stopVideo();
      }
    } catch (_) {}
    try {
      if (typeof previewTeaserPlayer.destroy === 'function') {
        previewTeaserPlayer.destroy();
      }
    } catch (_) {}
    previewTeaserPlayer = null;
  }

  if (previewTeaserContainer && previewTeaserContainer.parentNode) {
    previewTeaserContainer.classList.remove('teaser-fadeout');
    previewTeaserContainer.parentNode.removeChild(previewTeaserContainer);
  }

  previewTeaserContainer = null;
}

function startHeroPreviewTeaser() {
  if (!isPreviewOpen()) return;
  if (!previewMedia || !window.YT || typeof window.YT.Player !== 'function') return;
  stopPreviewTeaser();
  previewTeaserContainer = document.createElement('div');
  previewTeaserContainer.className = 'preview-teaser-player';
  previewMedia.appendChild(previewTeaserContainer);
  previewTeaserPlayer = new YT.Player(previewTeaserContainer, {
    playerVars: {
      autoplay: 1,
      mute: isMobile ? 1 : 0,
      playsinline: 1,
      controls: 0,
      modestbranding: 1,
      rel: 0,
      start: HERO_PREVIEW_START_SECONDS
    },
    events: {
      onReady: e => {
        applyPreviewVolumeToPlayer(e.target, desiredPreviewVolume, isMobile);
        if (isMobile && typeof e.target.mute === 'function') {
          e.target.mute();
        }
        if (!isMobile && typeof e.target.unMute === 'function') {
          e.target.unMute();
        }
        previewMuted = isMobile;
        previewVolume = desiredPreviewVolume;
        if (typeof e.target.loadVideoById === 'function') {
          e.target.loadVideoById({ videoId: HERO_VIDEO_ID, startSeconds: HERO_PREVIEW_START_SECONDS });
        } else if (typeof e.target.seekTo === 'function') {
          e.target.seekTo(HERO_PREVIEW_START_SECONDS, true);
        }
        updatePreviewVolumeUI({ muted: previewMuted, volume: previewVolume });
        const attemptPlay = () => {
          if (typeof e.target.playVideo === 'function') {
            e.target.playVideo();
          }
          setTimeout(() => {
            attemptPreviewAutounmuteIfPlaying(e.target);
```

```
          }, 0);
        };
        setTimeout(attemptPlay, 0);
        setTimeout(() => {
          if (!e.target || typeof e.target.getPlayerState !== 'function') return;
          const playingState = (window.YT && window.YT.PlayerState) ? window.YT.PlayerState.PLAYING : 1;
          if (e.target.getPlayerState() !== playingState) {
            attemptPlay();
            return;
          }
          attemptPreviewAutounmuteIfPlaying(e.target);
        }, 600);
        previewTeaserTimer = setTimeout(() => {
          stopPreviewTeaser();
        }, HERO_PREVIEW_DURATION_MS);
      }
    }
  });
}

  if (!previewMedia || !window.YT || typeof window.YT.Player !== 'function') return;
  const teaserUrl = currentPreviewCardEl ? currentPreviewCardEl.getAttribute('data-youtube-url') : '';

  previewTeaserContainer = document.createElement('div');
  previewTeaserContainer.className = 'preview-teaser-player';

  previewMedia.appendChild(previewTeaserContainer);
  previewTeaserPlayer = new YT.Player(previewTeaserContainer, {
    playerVars: {
      autoplay: 1,
      mute: isMobile ? 1 : 0,
      playsinline: 1,
      controls: 0,
      modestbranding: 1,
      rel: 0,
      start: teaserData.startSeconds
    },
    events: {
      onReady: e => {
        applyPreviewVolumeToPlayer(e.target, desiredPreviewVolume, isMobile);
        if (isMobile && typeof e.target.mute === 'function') {
          e.target.mute();
        }
        if (!isMobile && typeof e.target.unMute === 'function') {
          e.target.unMute();
        }
        previewMuted = isMobile;
        previewVolume = desiredPreviewVolume;
        if (typeof e.target.loadVideoById === 'function') {
          e.target.loadVideoById({ videoId: teaserData.videoId, startSeconds: teaserData.startSeconds });
        } else if (typeof e.target.seekTo === 'function') {
          e.target.seekTo(teaserData.startSeconds, true);
        }
        updatePreviewVolumeUI({ muted: previewMuted, volume: previewVolume });
        const attemptPlay = () => {
          if (typeof e.target.playVideo === 'function') {
            e.target.playVideo();
          }
          setTimeout(() => {
            attemptPreviewAutounmuteIfPlaying(e.target);
          }, 0);
        };
        setTimeout(attemptPlay, 0);
        setTimeout(() => {
          if (!e.target || typeof e.target.getPlayerState !== 'function') return;
          const playingState = (window.YT && window.YT.PlayerState) ? window.YT.PlayerState.PLAYING : 1;
          if (e.target.getPlayerState() !== playingState) {
            attemptPlay();
            return;
          }
          attemptPreviewAutounmuteIfPlaying(e.target);
        }, 600);
        previewTeaserTimer = setTimeout(() => {
          if (isA1PreviewCard(card)) {
            if (previewTeaserContainer) {
              previewTeaserContainer.classList.add('teaser-fadeout');
            }
            fadeOutPreviewTeaserAudio(previewTeaserPlayer);
            clearPreviewTeaserFadeTimeout();
            previewTeaserFadeTimeout = setTimeout(() => {
              stopPreviewTeaser();
            }, PREVIEW_TEASER_FADE_MS);
            return;
          }
```

```
          stopPreviewTeaser();
          showPreviewPoster(card);
        }, HERO_PREVIEW_DURATION_MS);
      }
    }
  });

let previewSwipeHandlers = null;

function isScrollableElement(element) {
  if (!element) return false;
  const style = getComputedStyle(element);
  const overflowY = style.overflowY;
  return (overflowY === 'auto' || overflowY === 'scroll') && element.scrollHeight > element.clientHeight;
}

  const candidate = target.closest('.preview-body');
  if (candidate && isScrollableElement(candidate)) {
    return candidate;
  }

  if (previewSwipeHandlers) return;
  const previewCard = previewOverlay.querySelector('.preview-card');

  if (!previewCard) return;

  let startY = 0;

    if (event.target && event.target.closest('button, input, a, .preview-actions, .preview-volume, .preview-play-butt
      swipeEligible = false;
      scrollableStartEl = null;
      return;
    }

      closePreview();
    }
    startY = 0;

  previewSwipeHandlers = {
    previewCard,
    onTouchStart,
    onTouchMove,
    onTouchEnd
  };

  previewCard.addEventListener('touchstart', onTouchStart, { passive: true });
  previewCard.addEventListener('touchmove', onTouchMove, { passive: false });

  previewCard.addEventListener('touchend', onTouchEnd);
  previewCard.addEventListener('touchcancel', onTouchEnd);

  if (!previewSwipeHandlers) return;
  const { previewCard, onTouchStart, onTouchMove, onTouchEnd } = previewSwipeHandlers;

  previewCard.removeEventListener('touchstart', onTouchStart);
  previewCard.removeEventListener('touchmove', onTouchMove);

  previewCard.removeEventListener('touchend', onTouchEnd);
  previewCard.removeEventListener('touchcancel', onTouchEnd);

  previewSwipeHandlers = null;
}

function captureHeroPausedTime() {
  try {
    if (typeof ytPlayer !== 'undefined' && ytPlayer && typeof ytPlayer.getCurrentTime === 'function') {
      const t = ytPlayer.getCurrentTime();
      if (typeof t === 'number' && isFinite(t)) {
        heroPausedTime = t;
      }
    }
  } catch (_) {}
}

  if (!previewPlaceholder || !card) return;
  const posterWide = card.getAttribute('data-poster-wide');

  previewPlaceholder.style.backgroundImage = `url(${posterWide})`;
  previewPlaceholder.style.backgroundSize = 'cover';
  previewPlaceholder.style.backgroundPosition = 'center';
  previewPlaceholder.style.transition = `opacity ${FILL_FADE_DURATION}ms ${currentMorphEasing}`;

  previewPlaceholder.style.opacity = '1';
}
```

```
let scrollLockState = null;

function animateOpenTile(tile, startRect, endRect, cardRadius, targetRadius, previewShadow, previewBorderColor, profi
  const fill = tile.querySelector('.preview-morph-fill');

  const startTileShadow = getComputedStyle(tile).boxShadow;
  const startTileBorderColor = getComputedStyle(tile).borderColor;

  const startX = startRect.left;
  const startY = startRect.top;
  const endX = endRect.left;
  const endY = endRect.top;

  const startW = startRect.width;
  const startH = startRect.height;
  const endW = endRect.width;
  const endH = endRect.height;

  const dX = endX - startX;
  const dY = endY - startY;

  const baseStart = {
    transform: `translate(${startX}px, ${startY}px)`,
    width: `${startW}px`,
    height: `${startH}px`,
    borderRadius: cardRadius,
    boxShadow: startTileShadow,
    borderColor: startTileBorderColor,
    opacity: 1,
    filter: 'none'
  };

  const baseEnd = {
    transform: `translate(${endX}px, ${endY}px)`,
    width: `${endW}px`,
    height: `${endH}px`,
    borderRadius: targetRadius,
    boxShadow: previewShadow,
    borderColor: previewBorderColor,
    opacity: 1,
    filter: 'none'
  };

  // Ensure a clean baseline (no residual transforms / filters).
  tile.style.transition = 'none';
  tile.style.transformOrigin = 'top left';
  tile.style.filter = 'none';
  tile.style.opacity = '1';
  tile.style.clipPath = '';
  tile.style.borderRadius = cardRadius;
  tile.style.boxShadow = startTileShadow;
  tile.style.borderColor = startTileBorderColor;

  if (fill) {
    fill.style.filter = 'none';
    fill.style.opacity = '1';
    fill.style.clipPath = '';
    fill.style.transform = 'none';
    fill.style.backgroundPosition = 'center';
  }

  const kf = (profile && profile.kf) ? profile.kf : 'straight';
  const dur = (profile && profile.duration) ? profile.duration : MORPH_DURATION;
  const ease = (profile && profile.easing) ? profile.easing : PREVIEW_EASING;

  const kfs = [];
  let options = { duration: dur, easing: ease, fill: 'forwards' };

  // Some concepts benefit from per-segment easing (keyframe easings).
  const midShadowStrong = buildBoostedShadow(previewShadow, 0.70);
  const midShadowSoft = buildBoostedShadow(previewShadow, 0.35);

  const add = (obj) => kfs.push(obj);

  if (kf === 'straight') {
    add({ offset: 0, ...baseStart });
    add({ offset: 1, ...baseEnd });
  }

  else if (kf === 'arcSweep') {
    const ax = (profile && profile.arcX != null) ? profile.arcX : 40;
    const ay = (profile && profile.arcY != null) ? profile.arcY : -28;

    add({ offset: 0, ...baseStart });
```

```
  add({
    offset: 0.55,
    transform: `translate(${startX + dX * 0.55 + ax}px, ${startY + dY * 0.55 + ay}px) rotate(0deg)`,
    width: `${startW + (endW - startW) * 0.55}px`,
    height: `${startH + (endH - startH) * 0.55}px`,
    borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.70)`,
    boxShadow: midShadowSoft,
    borderColor: previewBorderColor
  });
  add({ offset: 1, ...baseEnd });
}

else if (kf === 'overshootSnap') {
  const os = (profile && profile.overshoot != null) ? profile.overshoot : 16;

  add({ offset: 0, ...baseStart });
  add({
    offset: 0.78,
    transform: `translate(${endX + os}px, ${endY - os * 0.45}px)`,
    width: `${endW * 1.02}px`,
    height: `${endH * 1.02}px`,
    borderRadius: `calc(${targetRadius} * 0.92)`,
    boxShadow: midShadowStrong,
    borderColor: previewBorderColor,
    easing: 'cubic-bezier(0.16, 1, 0.3, 1)'
  });
  add({
    offset: 1,
    ...baseEnd,
    easing: 'cubic-bezier(0.20, 0.80, 0.20, 1)'
  });
}

else if (kf === 'springElastic') {
  const b = (profile && profile.bounce != null) ? profile.bounce : 3;

  add({ offset: 0, ...baseStart });
  add({
    offset: 0.58,
    transform: `translate(${endX + 18}px, ${endY - 10}px)`,
    width: `${endW * 1.04}px`,
    height: `${endH * 1.04}px`,
    borderRadius: `calc(${targetRadius} * 0.88)`,
    boxShadow: midShadowStrong,
    borderColor: previewBorderColor,
    easing: 'cubic-bezier(0.16, 1, 0.3, 1)'
  });
  add({
    offset: 0.74,
    transform: `translate(${endX - 10}px, ${endY + 6}px)`,
    width: `${endW * 0.99}px`,
    height: `${endH * 0.99}px`,
    borderRadius: `calc(${targetRadius} * 1.02)`,
    boxShadow: midShadowSoft,
    borderColor: previewBorderColor,
    easing: 'cubic-bezier(0.34, 1.56, 0.64, 1)'
  });
  if (b >= 3) {
    add({
      offset: 0.86,
      transform: `translate(${endX + 4}px, ${endY - 3}px)`,
      width: `${endW * 1.01}px`,
      height: `${endH * 1.01}px`,
      borderRadius: targetRadius,
      boxShadow: previewShadow,
      borderColor: previewBorderColor,
      easing: 'cubic-bezier(0.20, 0.80, 0.20, 1)'
    });
  }
  add({ offset: 1, ...baseEnd });
}

else if (kf === 'hingeFlip') {
  const tilt = (profile && profile.tilt != null) ? profile.tilt : 10;

  add({ offset: 0, ...baseStart });
  add({
    offset: 0.45,
    transform: `translate(${startX + dX * 0.45}px, ${startY + dY * 0.45}px) perspective(900px) rotateX(${tilt}deg)
    width: `${startW + (endW - startW) * 0.45}px`,
    height: `${startH + (endH - startH) * 0.45}px`,
    borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.55)`,
    boxShadow: midShadowSoft,
    borderColor: previewBorderColor
```

```
  });
  add({
    offset: 0.78,
    transform: `translate(${endX}px, ${endY}px) perspective(900px) rotateX(-4deg) rotateZ(2deg)`,
    width: `${endW}px`,
    height: `${endH}px`,
    borderRadius: targetRadius,
    boxShadow: previewShadow,
    borderColor: previewBorderColor
  });
  add({ offset: 1, ...baseEnd });
}

else if (kf === 'yawRotate') {
  const yaw = (profile && profile.yaw != null) ? profile.yaw : 14;

  add({ offset: 0, ...baseStart });
  add({
    offset: 0.52,
    transform: `translate(${startX + dX * 0.52}px, ${startY + dY * 0.52}px) perspective(900px) rotateY(${yaw}deg)`,
    width: `${startW + (endW - startW) * 0.52}px`,
    height: `${startH + (endH - startH) * 0.52}px`,
    borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.60)`,
    boxShadow: midShadowSoft,
    borderColor: previewBorderColor
  });
  add({
    offset: 0.78,
    transform: `translate(${endX}px, ${endY}px) perspective(900px) rotateY(-6deg)`,
    width: `${endW}px`,
    height: `${endH}px`,
    borderRadius: targetRadius,
    boxShadow: previewShadow,
    borderColor: previewBorderColor
  });
  add({ offset: 1, ...baseEnd });
}

else if (kf === 'popInPlace') {
  // Lingers near start, then "pops" and travels late.
  add({ offset: 0, ...baseStart });
  add({
    offset: 0.40,
    transform: `translate(${startX + dX * 0.12}px, ${startY + dY * 0.12}px) scale(1.08)`,
    width: `${startW + (endW - startW) * 0.18}px`,
    height: `${startH + (endH - startH) * 0.18}px`,
    borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.30)`,
    boxShadow: midShadowSoft,
    borderColor: previewBorderColor,
    opacity: 1
  });
  add({
    offset: 0.78,
    transform: `translate(${endX}px, ${endY}px) scale(1.02)`,
    width: `${endW}px`,
    height: `${endH}px`,
    borderRadius: targetRadius,
    boxShadow: previewShadow,
    borderColor: previewBorderColor
  });
  add({ offset: 1, ...baseEnd });
}

else if (kf === 'slideFadeCross') {
  const slideX = (profile && profile.slideX != null) ? profile.slideX : 80;

  add({ offset: 0, ...baseStart, opacity: 1 });
  add({
    offset: 0.45,
    transform: `translate(${startX + dX * 0.45 + slideX}px, ${startY + dY * 0.45}px)`,
    width: `${startW + (endW - startW) * 0.45}px`,
    height: `${startH + (endH - startH) * 0.45}px`,
    borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.50)`,
    boxShadow: midShadowSoft,
    borderColor: previewBorderColor,
    opacity: 0.45
  });
  add({
    offset: 0.78,
    transform: `translate(${endX}px, ${endY}px)`,
    width: `${endW}px`,
    height: `${endH}px`,
    borderRadius: targetRadius,
    boxShadow: previewShadow,
```

```
    borderColor: previewBorderColor,
    opacity: 1
  });
  add({ offset: 1, ...baseEnd, opacity: 1 });
}

else if (kf === 'verticalDrop') {
  const drop = (profile && profile.dropY != null) ? profile.dropY : 64;

  add({ offset: 0, ...baseStart });
  add({
    offset: 0.60,
    transform: `translate(${startX + dX * 0.60}px, ${startY + dY * 0.60 + drop}px)`,
    width: `${startW + (endW - startW) * 0.60}px`,
    height: `${startH + (endH - startH) * 0.60}px`,
    borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.68)`,
    boxShadow: midShadowStrong,
    borderColor: previewBorderColor
  });
  add({
    offset: 0.86,
    transform: `translate(${endX}px, ${endY}px)`,
    width: `${endW}px`,
    height: `${endH}px`,
    borderRadius: targetRadius,
    boxShadow: previewShadow,
    borderColor: previewBorderColor
  });
  add({ offset: 1, ...baseEnd });
}

else if (kf === 'diagonalSlash') {
  const sx = (profile && profile.slashX != null) ? profile.slashX : 72;
  const sy = (profile && profile.slashY != null) ? profile.slashY : -72;

  add({ offset: 0, ...baseStart });
  add({
    offset: 0.52,
    transform: `translate(${startX + dX * 0.52 + sx}px, ${startY + dY * 0.52 + sy}px) rotate(-6deg)`,
    width: `${startW + (endW - startW) * 0.52}px`,
    height: `${startH + (endH - startH) * 0.52}px`,
    borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.62)`,
    boxShadow: midShadowSoft,
    borderColor: previewBorderColor
  });
  add({
    offset: 0.80,
    transform: `translate(${endX}px, ${endY}px) rotate(2deg)`,
    width: `${endW}px`,
    height: `${endH}px`,
    borderRadius: targetRadius,
    boxShadow: previewShadow,
    borderColor: previewBorderColor
  });
  add({ offset: 1, ...baseEnd });
}

else if (kf === 'squashStretch') {
  add({ offset: 0, ...baseStart });
  add({
    offset: 0.28,
    transform: `translate(${startX + dX * 0.28}px, ${startY + dY * 0.28}px) scaleX(0.92) scaleY(1.08)`,
    width: `${startW + (endW - startW) * 0.28}px`,
    height: `${startH + (endH - startH) * 0.28}px`,
    borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.42)`,
    boxShadow: midShadowSoft,
    borderColor: previewBorderColor
  });
  add({
    offset: 0.58,
    transform: `translate(${startX + dX * 0.58}px, ${startY + dY * 0.58}px) scaleX(1.08) scaleY(0.96)`,
    width: `${startW + (endW - startW) * 0.58}px`,
    height: `${startH + (endH - startH) * 0.58}px`,
    borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.70)`,
    boxShadow: midShadowStrong,
    borderColor: previewBorderColor
  });
  add({ offset: 1, ...baseEnd });
}

else if (kf === 'blurBurst') {
  const blurPx = (profile && profile.blurPx != null) ? profile.blurPx : 10;

  add({ offset: 0, ...baseStart, filter: 'blur(0px)' });
```

```
  add({
    offset: 0.48,
    transform: `translate(${startX + dX * 0.48}px, ${startY + dY * 0.48}px)`,
    width: `${startW + (endW - startW) * 0.48}px`,
    height: `${startH + (endH - startH) * 0.48}px`,
    borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.60)`,
    boxShadow: midShadowSoft,
    borderColor: previewBorderColor,
    filter: `blur(${blurPx}px)`
  });
  add({ offset: 0.72, filter: 'blur(0px)' });
  add({ offset: 1, ...baseEnd, filter: 'blur(0px)' });
}

else if (kf === 'shutterWipe') {
  // Tile moves normally; the fill reveals like a shutter wipe.
  add({ offset: 0, ...baseStart });
  add({
    offset: 0.55,
    transform: `translate(${startX + dX * 0.55}px, ${startY + dY * 0.55}px)`,
    width: `${startW + (endW - startW) * 0.55}px`,
    height: `${startH + (endH - startH) * 0.55}px`,
    borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.70)`,
    boxShadow: midShadowSoft,
    borderColor: previewBorderColor
  });
  add({ offset: 1, ...baseEnd });

  if (fill) {
    try {
      fill.animate([
        { offset: 0, clipPath: 'inset(0 0 0 0)' },
        { offset: 0.18, clipPath: 'inset(0 0 0 0)' },
        { offset: 0.55, clipPath: 'inset(0 0 0 100%)' },
        { offset: 0.74, clipPath: 'inset(0 0 0 0)' },
        { offset: 1, clipPath: 'inset(0 0 0 0)' }
      ], { duration: dur, easing: 'linear', fill: 'forwards' });
    } catch (e) { /* no-op */ }
  }
}

else if (kf === 'portalZoom') {
  const portal = (profile && profile.portal != null) ? profile.portal : 1.22;

  add({ offset: 0, ...baseStart });
  add({
    offset: 0.72,
    transform: `translate(${endX}px, ${endY}px) scale(${portal})`,
    width: `${endW}px`,
    height: `${endH}px`,
    borderRadius: targetRadius,
    boxShadow: midShadowStrong,
    borderColor: previewBorderColor
  });
  add({
    offset: 0.88,
    transform: `translate(${endX}px, ${endY}px) scale(0.98)`,
    width: `${endW}px`,
    height: `${endH}px`,
    borderRadius: targetRadius,
    boxShadow: previewShadow,
    borderColor: previewBorderColor
  });
  add({ offset: 1, ...baseEnd });
}

else if (kf === 'stackLift') {
  const lift = (profile && profile.lift != null) ? profile.lift : 18;

  add({ offset: 0, ...baseStart });
  add({
    offset: 0.45,
    transform: `translate(${startX + dX * 0.45}px, ${startY + dY * 0.45 - lift}px) perspective(900px) translateZ(24
    width: `${startW + (endW - startW) * 0.45}px`,
    height: `${startH + (endH - startH) * 0.45}px`,
    borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.55)`,
    boxShadow: midShadowStrong,
    borderColor: previewBorderColor
  });
  add({
    offset: 0.78,
    transform: `translate(${endX}px, ${endY}px) perspective(900px) translateZ(0px)`,
    width: `${endW}px`,
    height: `${endH}px`,
```

```
    borderRadius: targetRadius,
    boxShadow: previewShadow,
    borderColor: previewBorderColor
  });
  add({ offset: 1, ...baseEnd });
}

else if (kf === 'parallaxDrift') {
  const driftX = (profile && profile.driftX != null) ? profile.driftX : 18;
  const driftY = (profile && profile.driftY != null) ? profile.driftY : -12;

  add({ offset: 0, ...baseStart });
  add({
    offset: 0.62,
    transform: `translate(${startX + dX * 0.62}px, ${startY + dY * 0.62}px)`,
    width: `${startW + (endW - startW) * 0.62}px`,
    height: `${startH + (endH - startH) * 0.62}px`,
    borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.74)`,
    boxShadow: midShadowSoft,
    borderColor: previewBorderColor
  });
  add({ offset: 1, ...baseEnd });

  if (fill) {
    try {
      fill.animate([
        { offset: 0, transform: 'translate(0px, 0px) scale(1.00)', backgroundPosition: 'center' },
        { offset: 0.55, transform: `translate(${driftX}px, ${driftY}px) scale(1.06)`, backgroundPosition: '60% 40%'
        { offset: 1, transform: 'translate(0px, 0px) scale(1.00)', backgroundPosition: 'center' }
      ], { duration: dur, easing: ease, fill: 'forwards' });
    } catch (e) { /* no-op */ }
  }
}

else if (kf === 'skewShear') {
  const skewDeg = (profile && profile.skewDeg != null) ? profile.skewDeg : 10;

  add({ offset: 0, ...baseStart });
  add({
    offset: 0.52,
    transform: `translate(${startX + dX * 0.52}px, ${startY + dY * 0.52}px) skewX(${skewDeg}deg)`,
    width: `${startW + (endW - startW) * 0.52}px`,
    height: `${startH + (endH - startH) * 0.52}px`,
    borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.64)`,
    boxShadow: midShadowSoft,
    borderColor: previewBorderColor
  });
  add({
    offset: 0.80,
    transform: `translate(${endX}px, ${endY}px) skewX(-4deg)`,
    width: `${endW}px`,
    height: `${endH}px`,
    borderRadius: targetRadius,
    boxShadow: previewShadow,
    borderColor: previewBorderColor
  });
  add({ offset: 1, ...baseEnd });
}

else if (kf === 'glitchJitter') {
  // Brief jitter micro-movements mid-flight.
  add({ offset: 0, ...baseStart });
  add({ offset: 0.40, transform: `translate(${startX + dX * 0.40}px, ${startY + dY * 0.40}px)`, width: `${startW +
  add({ offset: 0.52, transform: `translate(${startX + dX * 0.52 + 6}px, ${startY + dY * 0.52 - 4}px)`, width: `${s
  add({ offset: 0.56, transform: `translate(${startX + dX * 0.56 - 5}px, ${startY + dY * 0.56 + 3}px)`, width: `${s
  add({ offset: 0.60, transform: `translate(${startX + dX * 0.60 + 4}px, ${startY + dY * 0.60 + 1}px)`, width: `${s
  add({ offset: 0.68, transform: `translate(${startX + dX * 0.68}px, ${startY + dY * 0.68}px)`, width: `${startW +
  add({ offset: 1, ...baseEnd });
}

else if (kf === 'rippleExpand') {
  add({ offset: 0, ...baseStart });
  add({
    offset: 0.52,
    transform: `translate(${startX + dX * 0.52}px, ${startY + dY * 0.52}px) scale(1.08)`,
    width: `${startW + (endW - startW) * 0.52}px`,
    height: `${startH + (endH - startH) * 0.52}px`,
    borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.66)`,
    boxShadow: midShadowStrong,
    borderColor: previewBorderColor,
    opacity: 0.85
  });
  add({
    offset: 0.72,
```

```
      transform: `translate(${startX + dX * 0.72}px, ${startY + dY * 0.72}px) scale(0.98)`,
      width: `${startW + (endW - startW) * 0.72}px`,
      height: `${startH + (endH - startH) * 0.72}px`,
      borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.82)`,
      boxShadow: previewShadow,
      borderColor: previewBorderColor,
      opacity: 1
    });
    add({ offset: 1, ...baseEnd, opacity: 1 });
  }

  else if (kf === 'cinematicEase') {
    // Slow start, fast mid, slow end + subtle radius timing offset.
    options = { duration: dur, easing: 'linear', fill: 'forwards' };

    add({ offset: 0, ...baseStart, easing: 'cubic-bezier(0.40, 0.00, 0.60, 1)' });
    add({
      offset: 0.22,
      transform: `translate(${startX + dX * 0.18}px, ${startY + dY * 0.18}px)`,
      width: `${startW + (endW - startW) * 0.18}px`,
      height: `${startH + (endH - startH) * 0.18}px`,
      borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.22)`,
      boxShadow: 'none',
      borderColor: 'rgba(255,255,255,0.10)',
      easing: 'cubic-bezier(0.40, 0.00, 0.60, 1)'
    });
    add({
      offset: 0.62,
      transform: `translate(${startX + dX * 0.74}px, ${startY + dY * 0.74}px)`,
      width: `${startW + (endW - startW) * 0.74}px`,
      height: `${startH + (endH - startH) * 0.74}px`,
      borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.55)`, /* radius lags */
      boxShadow: midShadowSoft,
      borderColor: previewBorderColor,
      easing: 'cubic-bezier(0.16, 1, 0.3, 1)'
    });
    add({
      offset: 0.86,
      transform: `translate(${endX}px, ${endY}px)`,
      width: `${endW}px`,
      height: `${endH}px`,
      borderRadius: `calc(${cardRadius} + (${targetRadius} - ${cardRadius}) * 0.86)`,
      boxShadow: previewShadow,
      borderColor: previewBorderColor,
      easing: 'cubic-bezier(0.40, 0.00, 0.60, 1)'
    });
    add({ offset: 1, ...baseEnd });
  }

  // Safety fallback
  if (!kfs.length) {
    kfs.push({ offset: 0, ...baseStart });
    kfs.push({ offset: 1, ...baseEnd });
  }

  const anim = tile.animate(kfs, options);

  // Hard constraint: end clean (no rotation, skew, blur, opacity issues).
  anim.onfinish = () => {
    tile.style.transform = baseEnd.transform;
    tile.style.width = baseEnd.width;
    tile.style.height = baseEnd.height;
    tile.style.borderRadius = baseEnd.borderRadius;
    tile.style.boxShadow = baseEnd.boxShadow;
    tile.style.borderColor = baseEnd.borderColor;
    tile.style.opacity = '1';
    tile.style.filter = 'none';
    tile.style.clipPath = '';

    if (fill) {
      fill.style.filter = 'none';
      fill.style.clipPath = '';
      fill.style.transform = 'none';
      fill.style.backgroundPosition = 'center';
    }
  };

  return anim;
}

  const previewCard = previewOverlay.querySelector('.preview-card');
  if (!previewCard) return null;

  const wasVisible = previewOverlay.classList.contains('is-visible');
```

```
        const wasActive = previewCard.classList.contains('is-active');

        const previousOverlayTransition = previewOverlay.style.transition;
        const previousCardTransition = previewCard.style.transition;

        const previousOverlayOpacity = previewOverlay.style.opacity;
        const previousOverlayVisibility = previewOverlay.style.visibility;

        const previousOverlayPointerEvents = previewOverlay.style.pointerEvents;
        const previousTransform = previewCard.style.transform;

        const previousOpacity = previewCard.style.opacity;

        previewOverlay.style.transition = 'none';

        previewCard.style.transition = 'none';

        previewOverlay.classList.add('is-visible');

        previewOverlay.setAttribute('aria-hidden', 'false');
        previewOverlay.style.opacity = '0';

        previewOverlay.style.visibility = 'hidden';
        previewOverlay.style.pointerEvents = 'none';

        previewCard.classList.add('is-active');
        previewCard.style.transform = '';

        previewCard.style.opacity = '';

        const rect = previewCard.getBoundingClientRect();

          previewCard.classList.remove('is-active');
        }

        previewOverlay.style.transition = previousOverlayTransition;

        previewCard.style.transition = previousCardTransition;
        previewOverlay.style.opacity = previousOverlayOpacity;

        previewOverlay.style.visibility = previousOverlayVisibility;
        previewOverlay.style.pointerEvents = previousOverlayPointerEvents;

        previewCard.style.transform = previousTransform;
        previewCard.style.opacity = previousOpacity;

        tile.className = 'preview-morph-tile';

        const fill = document.createElement('div');

        fill.className = 'preview-morph-fill';

        tile.classList.remove('is-fade-out');

      function openPreviewFromCard(card, triggerElement = card) {
        // ZAQ TASK 1.8: Pause hero video when preview begins opening.
        previewOpening = true;
        const heroAudioState = getHeroAudioState();
        const syncedVolume = clampPreviewVolume(heroAudioState.volume);
        initialPreviewVolume = syncedVolume === 0 ? 50 : syncedVolume;
        desiredPreviewVolume = initialPreviewVolume;
        previewVolume = desiredPreviewVolume;
        previewMuted = isMobile;
        previewLastNonZeroVolume = previewVolume || 50;
        if (previewVolumeSlider) {
          previewVolumeSlider.value = String(previewVolume);
        }
        updatePreviewVolumeUI({ muted: previewMuted, volume: previewVolume });
        clearPreviewVolumeInitialClose();
        openPreviewVolume();
        previewVolumeInitialTimeout = window.setTimeout(() => {
          if (previewVolumeCloseReason !== 'slider' && previewVolumeControl) {
            previewVolumeControl.classList.remove('is-open');
            previewVolumeCloseReason = null;
          }
          previewVolumeInitialTimeout = null;
        }, initialPreviewVolumeOpenDelay);
        triggerVolumeSweep(previewVolumeControl);
        const title = 'Preview Placeholder Title';
        const runtime = '1h 52m';
        const tags = 'Thriller • Sci■Fi • Action';
        const description = 'This is a static placeholder description used for layout validation only. It is intentionally
previewTitleEl.textContent = title;
        previewRuntimeEl.textContent = runtime;
```

```
  previewTagsEl.textContent = tags;
  previewDescriptionEl.textContent = description;

  lastOpenedCard = triggerElement;
  currentPreviewCardEl = card;
  if (isA1PreviewCard(card)) {
    setPreviewPosterImage(card);
  } else {
    resetPreviewPosterImage();
  }
  lockScroll();
  previewOverlay.setAttribute('aria-hidden', 'false');
  const profile = getOpenMorphProfileForCard(card);
  currentMorphDuration = profile.duration;
  currentMorphEasing = profile.easing;
  // Sync overlay/card CSS variable timings to this open.
  previewOverlay.style.setProperty('--preview-morph-duration', `${profile.duration}ms`);
  previewOverlay.style.setProperty('--preview-morph-ease', profile.easing);
  const previewCard = previewOverlay.querySelector('.preview-card');
  if (previewCard) {
    previewCard.style.setProperty('--preview-morph-duration', `${profile.duration}ms`);
    previewCard.style.setProperty('--preview-morph-ease', profile.easing);
  }
  attachPreviewSwipeListeners();
  const isHeroPreview = !!(heroPreviewSource && (card === heroPreviewSource || card === heroSection || card === heroV
  runOpenMorph(card, profile).then(() => {
    previewOpening = false;
    if (isHeroPreview) {
      startHeroPreviewTeaser();
    } else {
      startCardPreviewTeaser(card);
    }
  });
}

const previewPlayButton = document.querySelector('.preview-play-button');

if (heroDetailsButton && heroPreviewSource) {
  heroDetailsButton.addEventListener('click', (event) => {
    if (previewOverlay.classList.contains('is-visible')) return;
    event.preventDefault();
    event.stopPropagation();
    openPreviewFromCard(heroPreviewSource, heroDetailsButton);
  });
}

if (previewPlayButton) {
  previewPlayButton.addEventListener('click', (event) => {
    event.preventDefault();
    event.stopPropagation();
    stopPreviewTeaser();
    if (typeof openFullscreenPlayback === 'function') {
      const playUrl = currentPreviewCardEl
        ? (currentPreviewCardEl.getAttribute('data-youtube-play-url') || currentPreviewCardEl.getAttribute('data-yout
        : '';
      const playVideoId = getYouTubeVideoId(playUrl);
      const resolvedPlayUrl = playVideoId ? `https://youtu.be/${playVideoId}` : playUrl;
      openFullscreenPlayback(resolvedPlayUrl);
    }
  });
}

    const previewCard = previewOverlay.querySelector('.preview-card');
    if (previewCard) {
      previewCard.style.opacity = '0';
      previewCard.style.transition = '';
      previewCard.classList.remove('is-hidden');
    }

    if (!previewCard || !endRect) {
      previewOverlay.classList.add('is-visible');
      previewCard.classList.add('is-active');
      resolve();
      return;
    }

    previewOverlay.style.opacity = '';
    previewOverlay.style.pointerEvents = '';

    previewOverlay.classList.add('is-visible');
    previewOverlay.setAttribute('aria-hidden', 'false');

    previewCard.classList.add('is-active');
    previewCard.style.opacity = '0';
```

```
    previewCard.style.transition = '';
    previewCard.style.transform = '';

    const previewShadow = getComputedStyle(previewCard).boxShadow;
    const previewBorderColor = getComputedStyle(previewCard).borderColor;

    const originalPreviewParent = previewCard.parentElement;
    const originalPreviewNextSibling = previewCard.nextSibling;

    tile.appendChild(previewCard);

    previewCard.style.position = 'absolute';

    previewCard.style.inset = '0';
    previewCard.style.width = '100%';

    previewCard.style.height = '100%';
    previewCard.style.margin = '0';

    previewCard.style.border = '0';
    previewCard.style.boxShadow = 'none';

    previewCard.style.borderRadius = 'inherit';
    previewCard.style.overflow = 'hidden';

    const targetRadius = getComputedStyle(previewCard).borderRadius;

    fill.style.transition = '';

          originalPreviewParent.insertBefore(previewCard, originalPreviewNextSibling);
        } else {

          originalPreviewParent.appendChild(previewCard);
        }
      }

      previewCard.style.position = '';

      previewCard.style.inset = '';
      previewCard.style.width = '';

      previewCard.style.height = '';
      previewCard.style.margin = '';

      previewCard.style.border = '';
      previewCard.style.boxShadow = '';

      previewCard.style.borderRadius = '';
      previewCard.style.overflow = '';

      previewCard.style.transition = '';
      previewCard.style.opacity = '';

      previewCard.removeEventListener('transitionend', handleCardFadeInEnd);
      finishCrossfade();

      previewCard.addEventListener('transitionend', handleCardFadeInEnd);
      previewCard.style.transition = `opacity ${FILL_FADE_DURATION}ms ${currentMorphEasing}`;

      previewCard.style.opacity = '1';
      fill.style.opacity = '0';

        previewCard.removeEventListener('transitionend', handleCardFadeInEnd);
        finishCrossfade();

      previewShadow,
      previewBorderColor,
      resolvedProfile
    );

function closePreview() {
  if (!previewOverlay || !previewOverlay.classList.contains('is-visible')) return;
  previewOpening = false;
  stopPreviewTeaser();
  resetPreviewPlaceholder();
  resetPreviewPosterImage();
  removePreviewAutounmuteListener();
  clearPreviewVolumeTimers();

  const previewCard = previewOverlay.querySelector('.preview-card');
  if (!previewCard) return;
  detachPreviewSwipeListeners();

  const previewCardRect = previewCard.getBoundingClientRect();
```

```
if (!lastOpenedCard) {
  previewOverlay.classList.remove('is-visible');
  previewOverlay.setAttribute('aria-hidden', 'true');
  previewCard.classList.remove('is-active');
  previewCard.style.opacity = '';
  previewCard.style.pointerEvents = '';
  if (activeMorphTile && activeMorphTile.parentNode) {
    activeMorphTile.parentNode.removeChild(activeMorphTile);
  }
  activeMorphTile = null;
  unlockScroll();
  lastOpenedCard = null;
  currentPreviewCardEl = null;
  if (typeof window.updateActiveTrackFromWindowScroll === 'function') {
    window.updateActiveTrackFromWindowScroll();
  }
  return;
}
const cardRect = lastOpenedCard.getBoundingClientRect();

let tile = activeMorphTile;
let fill = null;

if (!tile) {
  const created = createGlowTileFromCard(lastOpenedCard, previewCardRect);
  tile = created.tile;
  fill = created.fill;
  activeMorphTile = tile;
} else {
  tile.style.borderRadius = getComputedStyle(previewCard).borderRadius;
  tile.style.width = `${previewCardRect.width}px`;
  tile.style.height = `${previewCardRect.height}px`;
  tile.style.transform = `translate(${previewCardRect.left}px, ${previewCardRect.top}px)`;
  tile.classList.remove('is-fade-out');
  tile.style.opacity = '';
  fill = tile.querySelector('.preview-morph-fill');
  if (fill) {
    fill.classList.remove('is-fade-out');
    fill.style.opacity = '1';
    const posterImg = lastOpenedCard.querySelector('img');
    if (posterImg && posterImg.src) {
      fill.style.backgroundImage = `url(${posterImg.src})`;
    }
  }
}

previewCard.style.opacity = '0';
previewCard.style.pointerEvents = 'none';

previewOverlay.classList.remove('is-visible');
previewOverlay.setAttribute('aria-hidden', 'true');

const targetRadius = getComputedStyle(lastOpenedCard).borderRadius;

// Keep close timing compatible with the last open profile.
tile.style.setProperty('--preview-morph-duration', `${currentMorphDuration}ms`);
tile.style.setProperty('--preview-morph-ease', currentMorphEasing);

requestAnimationFrame(() => {
  tile.offsetHeight;
  moveTile(tile, previewCardRect, cardRect, targetRadius);
});

setTimeout(() => {
  previewCard.classList.remove('is-active');
  previewCard.style.opacity = '';
  previewCard.style.pointerEvents = '';

  if (tile && tile.parentNode) {
    tile.parentNode.removeChild(tile);
  }
  if (activeMorphTile === tile) {
    activeMorphTile = null;
  }
  activeMorphTile = null;

  unlockScroll();

  if (lastOpenedCard && document.body.contains(lastOpenedCard)) {
    lastOpenedCard.focus();
  }
  lastOpenedCard = null;
  currentPreviewCardEl = null;
  if (typeof window.updateActiveTrackFromWindowScroll === 'function') {
```

```
            window.updateActiveTrackFromWindowScroll();
          }
        }, currentMorphDuration);
      }

      previewCloseButton.addEventListener('click', () => {
        closePreview();
      });

      previewOverlay.addEventListener('click', (event) => {
        const previewCard = previewOverlay.querySelector('.preview-card');
        if (!previewCard) return;
        if (previewCard.contains(event.target)) return;
        closePreview();
      });

        const previewButton = event.target.closest('.poster-preview-button');
        if (!previewButton) return;

        const card = previewButton.closest('.poster');
        if (!card) return;

        openPreviewFromCard(card, previewButton);
      }, true);

        // Open preview from focused poster using Enter/Space.
        if ((event.key === 'Enter' || event.key === ' ') && !previewOverlay.classList.contains('is-visible')) {
          const active = document.activeElement;
          const card = active ? active.closest('.poster[data-morph-id]') : null;
          if (card) {
            event.preventDefault();
            openPreviewFromCard(card, card);
          }
        }

        if ((event.key === 'Escape' || event.key === 'Esc') && previewOverlay.classList.contains('is-visible')) {
          event.preventDefault();
          closePreview();
        }


/* ZAQ TASK: Hero glow truth → YouTube playback (preview override wins) */
(function(){
  var heroEl = document.querySelector('.hero');
  var previewEl = document.querySelector('.preview-overlay');

  var pauseTimer = null;          // single shared delayed-pause timer
  var pendingIntent = null;       // 'play' | 'pause' | null
  var flushTimer = null;          // polls until yt player is ready

  var lastHeroActive = null;
  var lastPreviewOpen = null;

  function isHeroActive(){
    return !!(heroEl && heroEl.classList.contains('hero--active'));
  }

  function isPreviewOpen(){
    return !!(previewEl && previewEl.classList.contains('is-visible'));
  }

  function clearPauseTimer(){
    if (pauseTimer){
      clearTimeout(pauseTimer);
      pauseTimer = null;
    }
  }

  function getPlayer(){
    try{
      if (typeof ytPlayer !== 'undefined' && ytPlayer) return ytPlayer;
    }catch(e){
      /* ignore */
    }
    try{
      if (window.ytPlayer) return window.ytPlayer;
    }catch(e){
      /* ignore */
    }
    return null;
  }

  function canControl(p){
    return !!(p && typeof p.playVideo === 'function' && typeof p.pauseVideo === 'function');
```

```
}

function stopFlushTimer(){
  if (flushTimer){
    clearInterval(flushTimer);
    flushTimer = null;
  }
}

function ensureFlushTimer(){
  if (flushTimer) return;
  flushTimer = setInterval(function(){
    var p = getPlayer();
    if (!canControl(p)) return;
    stopFlushTimer();
    // apply current truth state as soon as the player becomes controllable
    syncToTruth({ initial:true, force:true });
  }, 200);
}

function applyIntent(intent){
  pendingIntent = intent;
  var p = getPlayer();

  if (!canControl(p)){
    ensureFlushTimer();
    return;
  }

  try{
    if (pendingIntent === 'play'){
      p.playVideo();
    } else if (pendingIntent === 'pause'){
      p.pauseVideo();
    }
  }catch(e){
    /* no-op */
  }
}

function scheduleDelayedPause(){
  clearPauseTimer();
  pauseTimer = setTimeout(function(){
    pauseTimer = null;
    // only pause if preview still closed AND glow still inactive
    if (!isPreviewOpen() && !isHeroActive()){
      applyIntent('pause');
    }
  }, 1000);
}

function syncToTruth(opts){
  opts = opts || {};
  var previewOpen = isPreviewOpen();
  var heroActive = isHeroActive();

  // Preview override (always wins)
  if (previewOpen){
    clearPauseTimer();
    applyIntent('pause');
    lastHeroActive = heroActive;
    lastPreviewOpen = previewOpen;
    return;
  }

  // Page load / pageshow rule
  if (opts.initial){
    clearPauseTimer();
    if (heroActive){
      applyIntent('play');
    } else {
      applyIntent('pause');
    }
    lastHeroActive = heroActive;
    lastPreviewOpen = previewOpen;
    return;
  }

  // Scroll back rule (returning to hero): active => play immediately
  if (heroActive){
    clearPauseTimer();
    applyIntent('play');
    lastHeroActive = heroActive;
    lastPreviewOpen = previewOpen;
```

```
      return;
    }

    // Scroll down rule (leaving hero): active -> inactive triggers ~1s delayed pause
    if (lastHeroActive === true && heroActive === false){
      scheduleDelayedPause();
    } else if (opts.force && heroActive === false){
      // safety: if player becomes ready while inactive, do not let it keep playing
      scheduleDelayedPause();
    }

    lastHeroActive = heroActive;
    lastPreviewOpen = previewOpen;
  }

  function installObservers(){
    if (heroEl){
      new MutationObserver(function(){
        syncToTruth({ initial:false });
      }).observe(heroEl, { attributes:true, attributeFilter:['class'] });
    }

    if (previewEl){
      new MutationObserver(function(){
        syncToTruth({ initial:false });
      }).observe(previewEl, { attributes:true, attributeFilter:['class'] });
    }
  }

  function requeryElements(){
    heroEl = document.querySelector('.hero');
    previewEl = document.querySelector('.preview-overlay');
  }

  // Initial wiring
  installObservers();

  // Initial load rule (run now; script is placed at end of body)
  syncToTruth({ initial:true });

  // BFCache restore / refresh consistency
  window.addEventListener('pageshow', function(){
    requeryElements();
    installObservers();
    syncToTruth({ initial:true });
  });
})();
```