The slide framework built for agents.

Every page is a React component on a 1920 × 1080 canvas. This deck walks the loop: init, prompt, edit, comment, present.

Press → to begin

```
$ npx @open-slide/cli init
```

The idea

A slide is a file.

No DSL, no template language. Each page is a React component, and the default export is the deck.

slides/hello/index.tsx — TSX · 14 LINES

```tsx
import type { Page } from '@open-slide/core';

const WORD = 'deck';

const Cover: Page = () => (
  <div style={{ padding: 160, background: '#fff' }}>
    <h1 style={{ fontSize: 188, letterSpacing: '-0.04em' }}>
      Hello, <em style={{ color: '#de3b3d' }}>{WORD}</em>.
    </h1>
    <p>A React slide, rendered live.</p>
  </div>
);

export default [Cover] satisfies Page[];
```

rendered output — LIVE



O1 · Init

One command.

Scaffolds the workspace, installs dependencies, and initializes git. No global installs, no config to touch.

~/code — zsh

```
$ npx @open-slide/cli init my-deck

✓ Created open-slide workspace in ~/code/my-deck
✓ Installed dependencies with pnpm
✓ Initialized git repository with first commit

Next steps:
  cd my-deck
  pnpm dev

Then open the dev server and start authoring in slides/<your-slide>/.
$
```

02 · Prompt

Ask your agent.

One /create-slide prompt drafts every page as a real component. Refine with follow-ups, not templates.

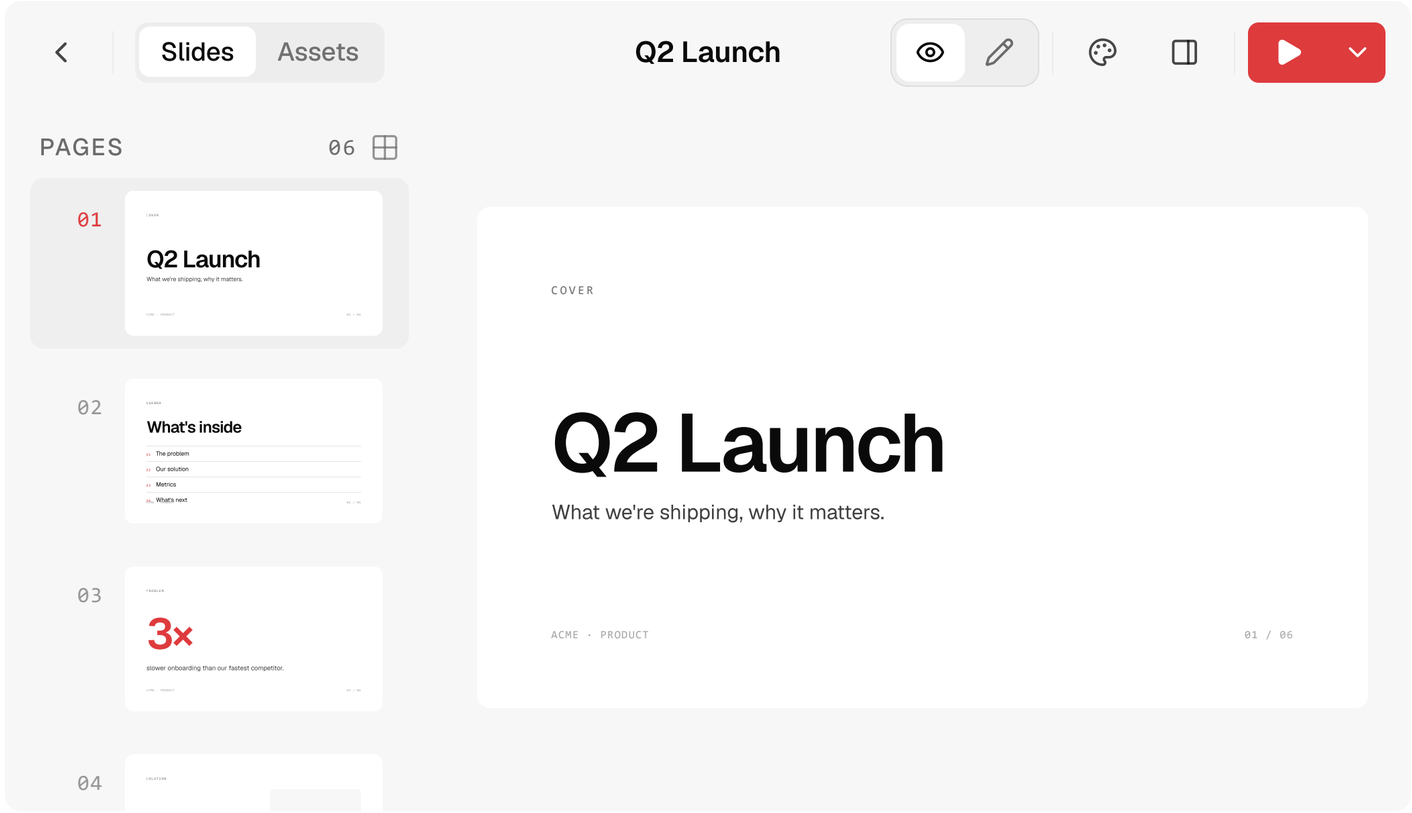
claude · ~/my-deck

```
› /create-slide the Q2 launch deck

Drafting 6 pages…
write slides/q2-launch/index.tsx
hmr localhost:5173/s/q2-launch ✓
Done. Cover, Agenda, Problem,
Solution, Metrics, Next.
›
```

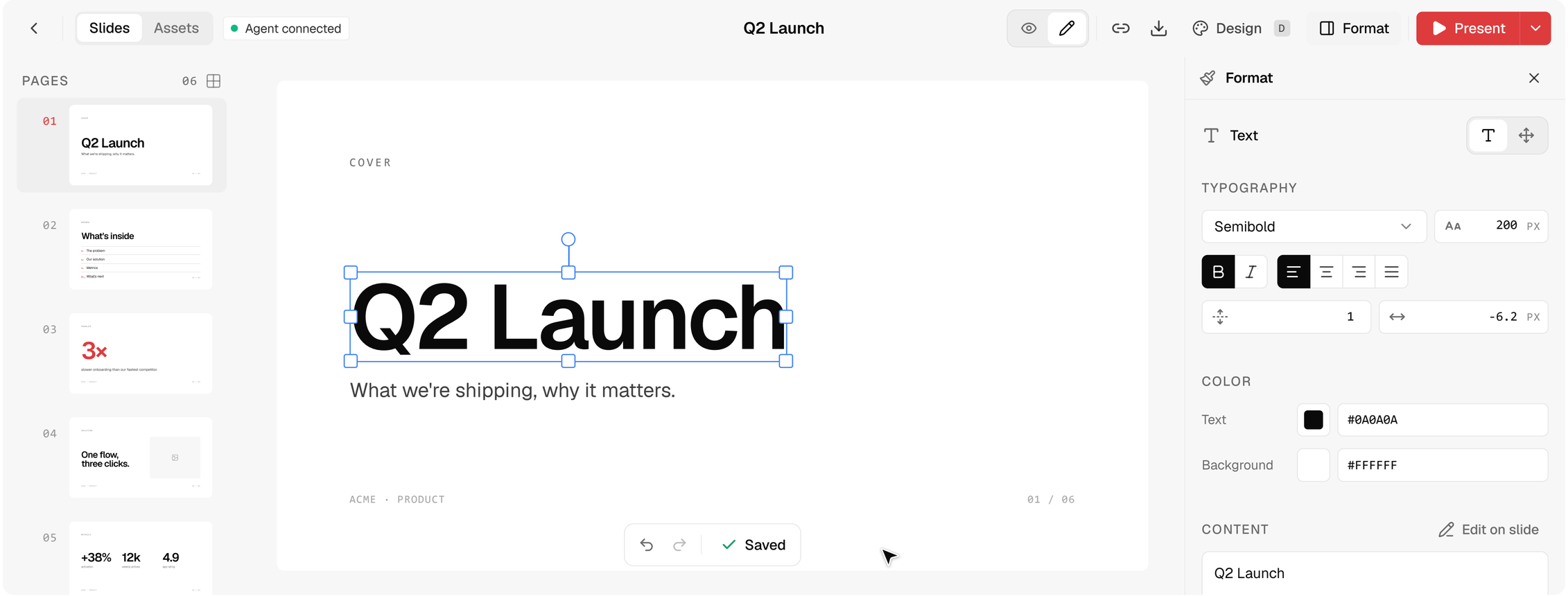
Claude Code · Codex · Cursor · Gemini …

ANY AGENT



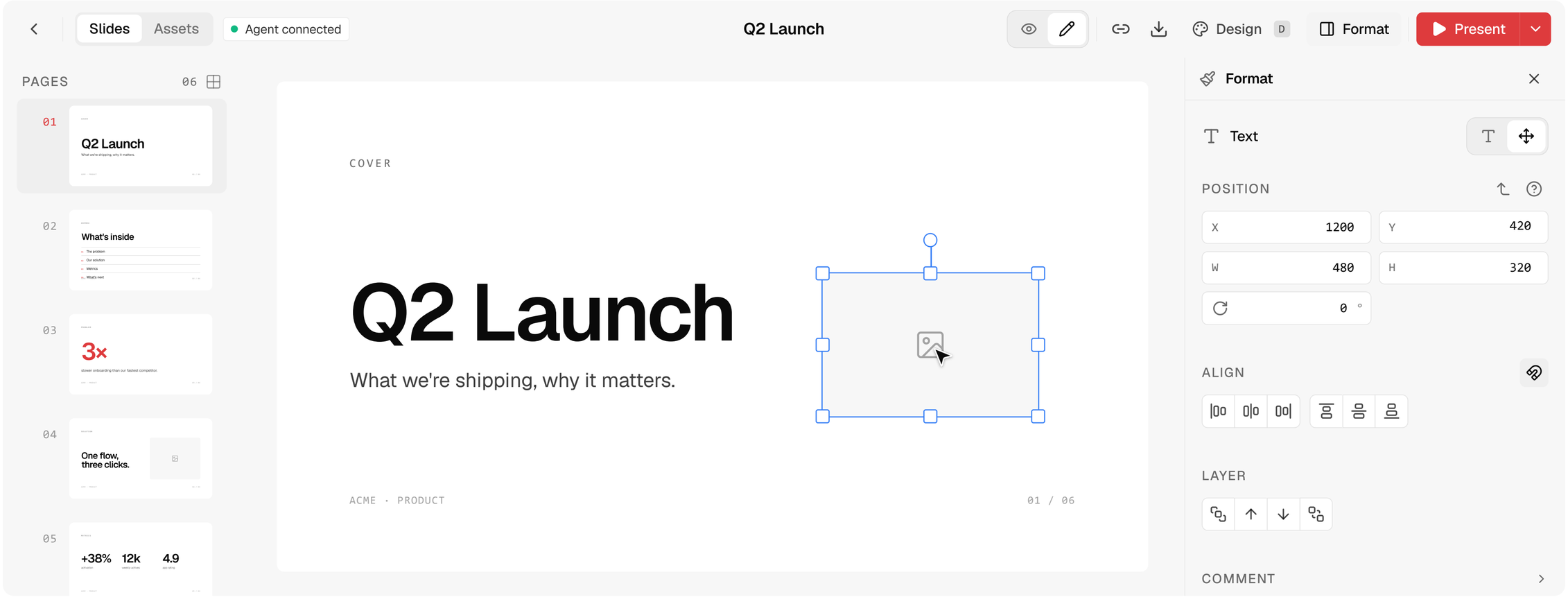
03 · Edit

Click any element. Tweak it. Save once.



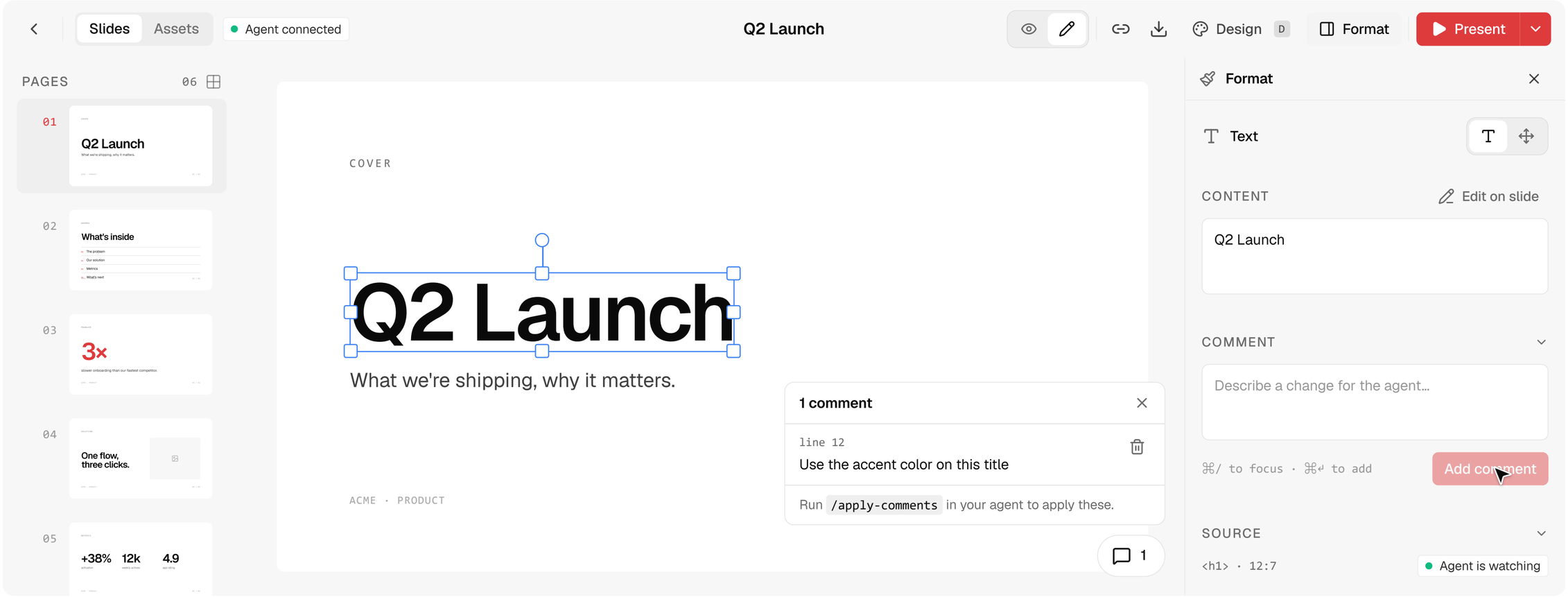
03 · Edit

Drag. Align. Snap.



04 · Comment

Point at what's wrong.



04 · Apply

The agent applies it.

Run /apply-comments. The agent reads each marker, edits exactly what you flagged, and removes the marker.

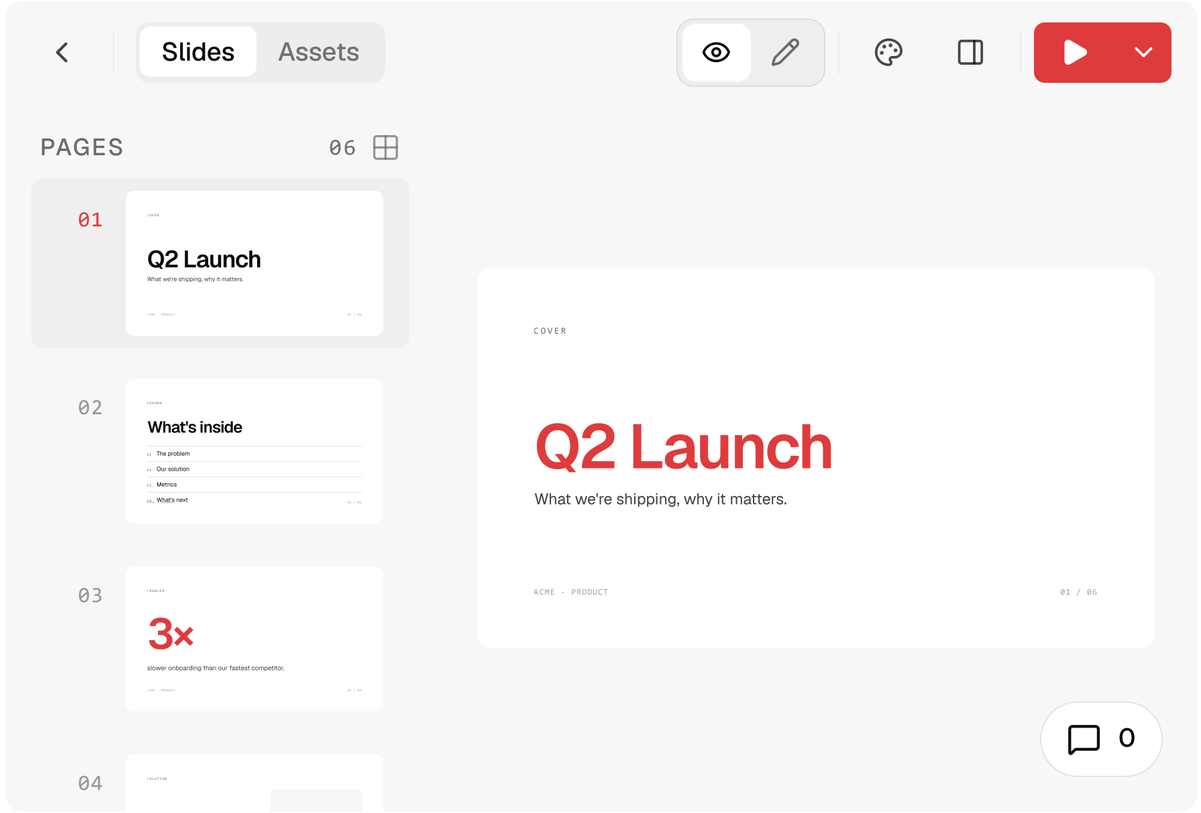
claude · ~/my-deck

```
› /apply-comments

1 comment found. Applying…

11  <section>
12  {/* @slide-comment id="c-a1b2c3d4" text="Use the accent co…
13    <h1 style={{ color: '#de3b3d' }}>Q2 Launch</h1>
14  </section>

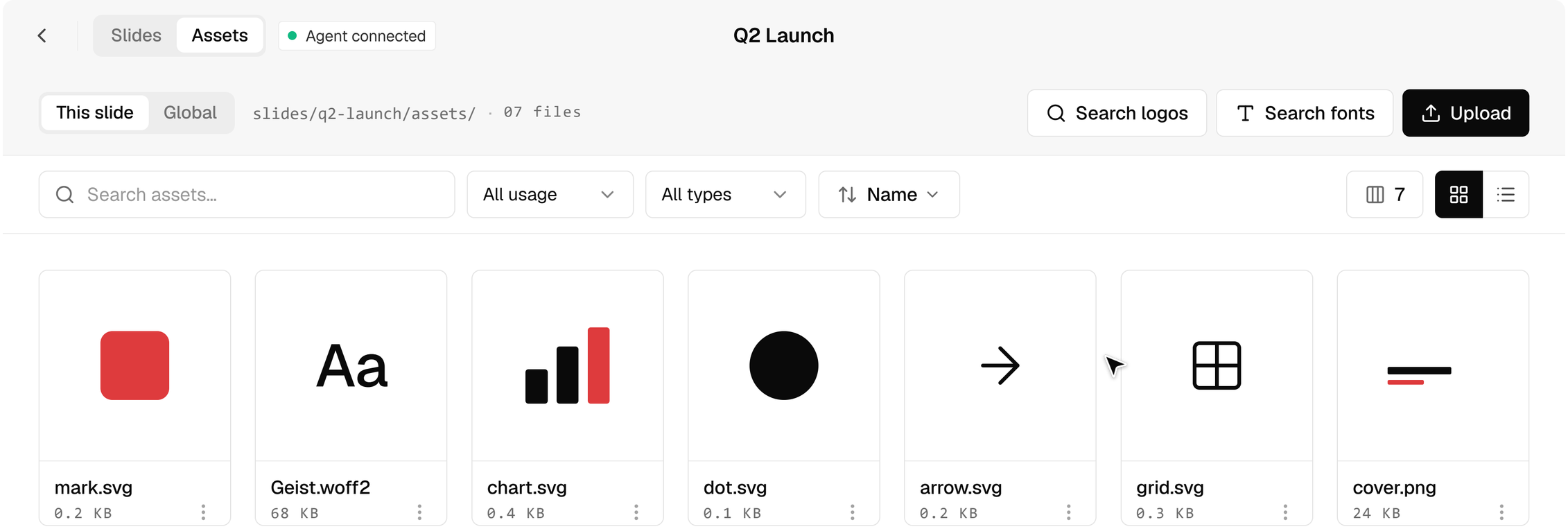
edit slides/q2-launch/index.tsx
✓ 1 comment applied
```



Toolbox

Assets, in one place.

Drop files, search logos on svgl, pull Google Fonts. Everything lands in slides/<id>/assets/, ready to import.



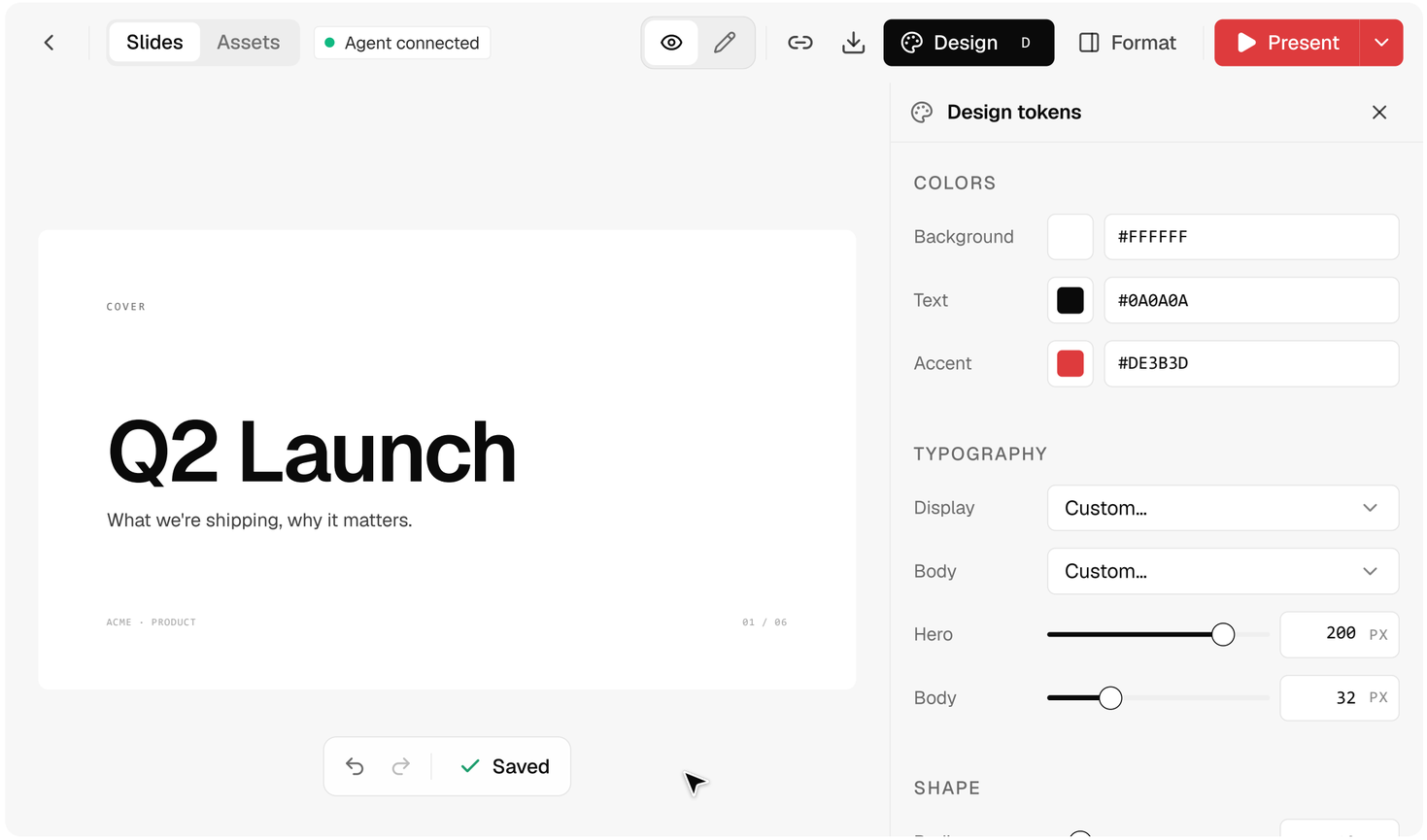
Toolbox

Design tokens, live.

slides/q2-launch/index.tsx DESIGN

```ts
export const design: DesignSystem = {
  palette: {
    bg: '#ffffff',
    text: '#0a0a0a',
    accent: '#de3b3d',
  },
  fonts: {
    display: 'Geist, sans-serif',
    body: 'Geist, sans-serif',
  },
  typeScale: { hero: 200, body: 32 },
  radius: 12,
};

// read as var(--osd-size-hero)
```



Toolbox

Pin a look. Reuse it.

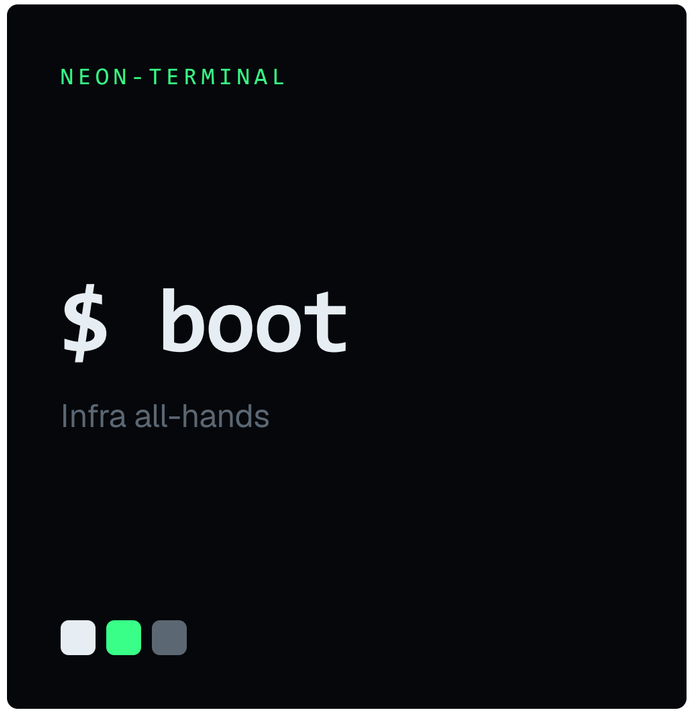
/create-theme writes themes/<id>.md: palette, type, and paste-ready Title and Footer components.
/create-slide reads it before drafting.



themes/editorial-noir.md — DARK · SERIF



themes/paper-press.md — LIGHT · SERIF



themes/neon-terminal.md — DARK · MONO

Toolbox

Motion, in three primitives.

Stepped reveals, one house transition, and shared-element morphs. All declared in the file.

01 Set the stage.

02 Layer the consequence.

03 Land the turn.

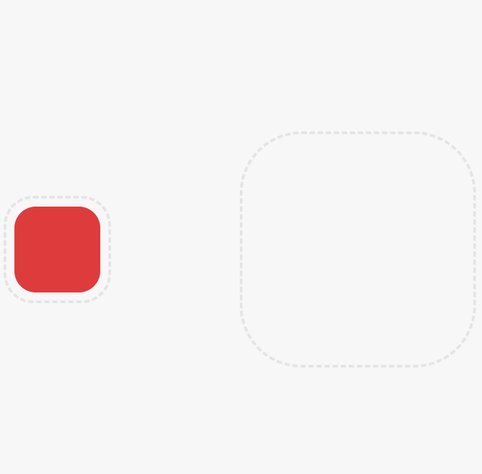
`<Steps> / <Step>`

Each → reveals the next beat. Jumping in shows the page complete.



`SlideTransition`

One DNA per deck: 6 px rise, 200 ms, ease-out. This deck uses it.



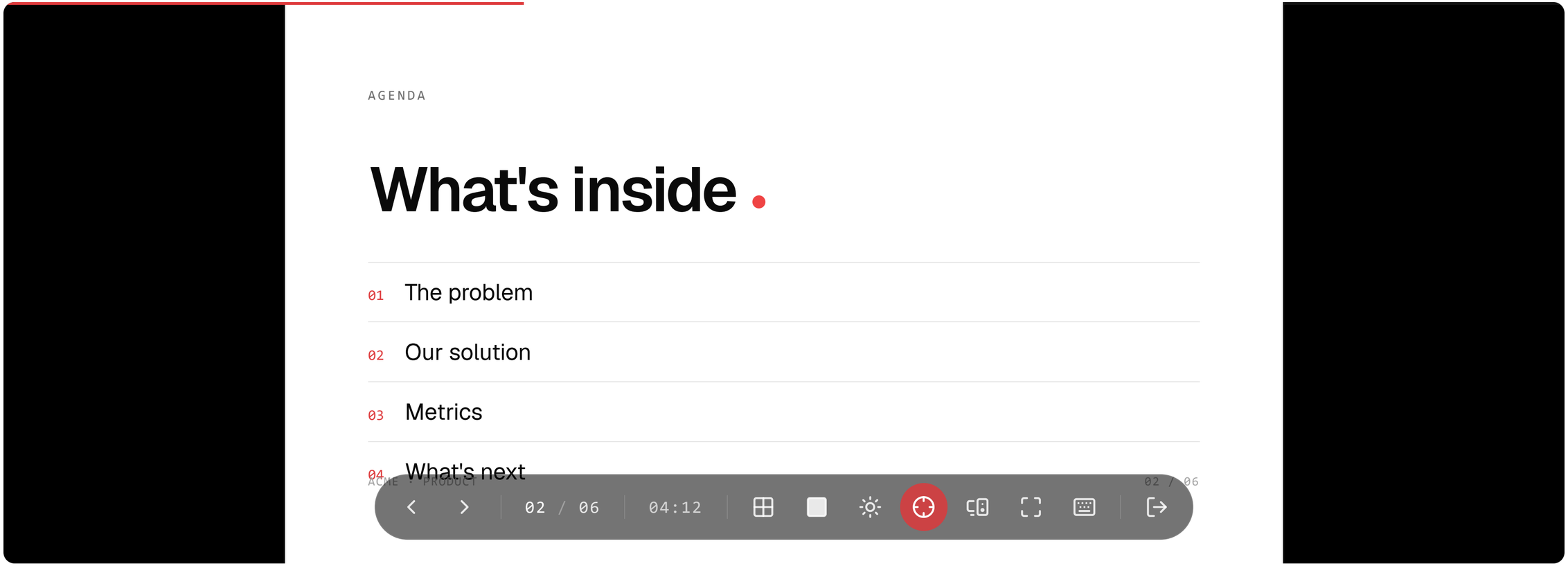
`MorphElement`

Same id on two pages, and the object glides between them. The eyebrow mark just did.

05 · Present

Present from the browser.

F for fullscreen, Enter for a window. Overview, laser, blackout, and the keyboard map are one key away.



05 · Present

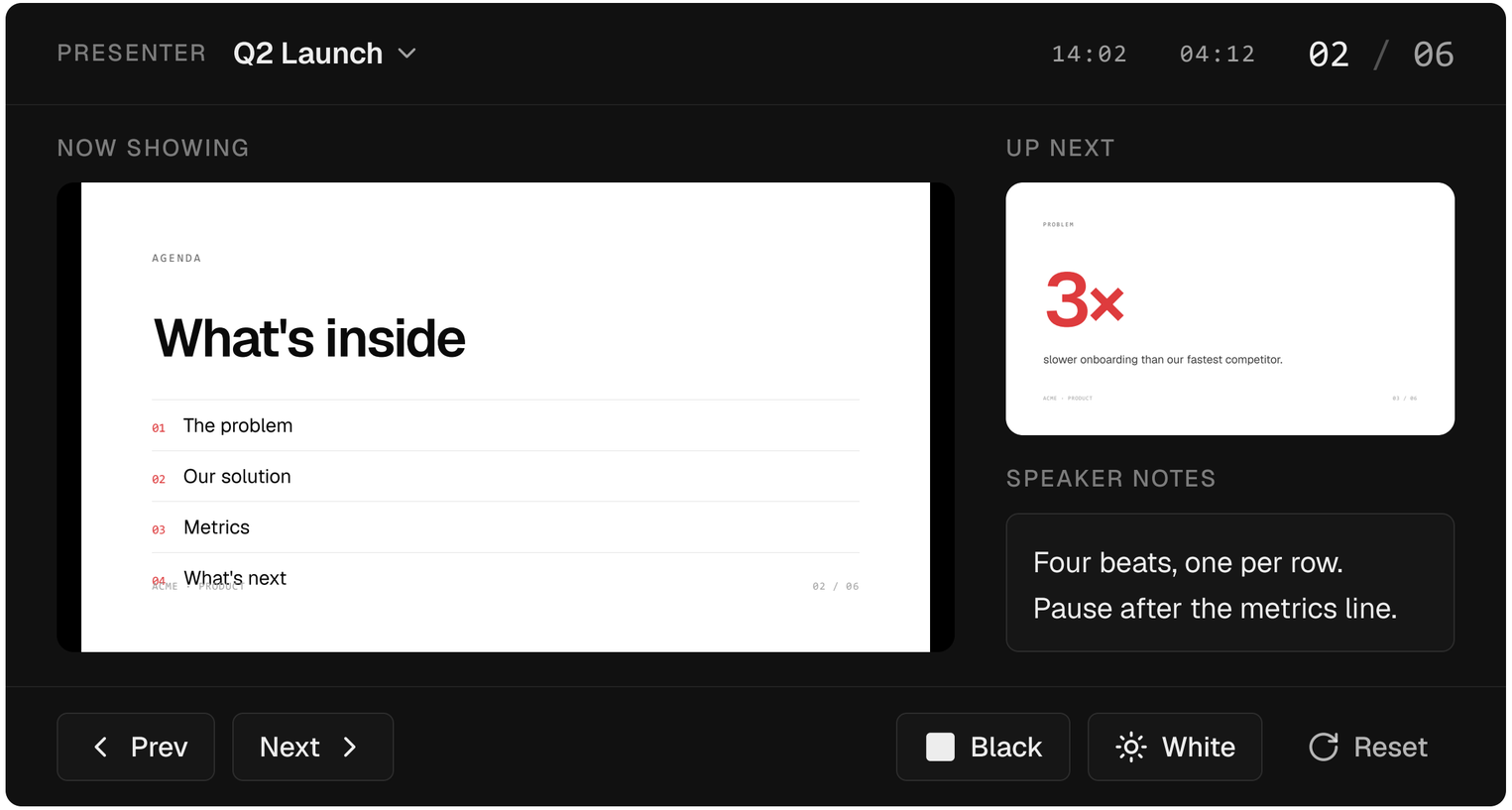
Notes travel with the deck.

Export notes next to your pages. Presenter view shows the current slide, what's next, and your script with a timer.

slides/q2-launch/index.tsx NOTES

```
export const notes = [
  'Open with the customer quote.',
  'Four beats, one per row.',
  'Let the 3× land. Then breathe.',
  undefined,
  'Cite the Q1 baseline first.',
  'End on the waitlist link.',
];

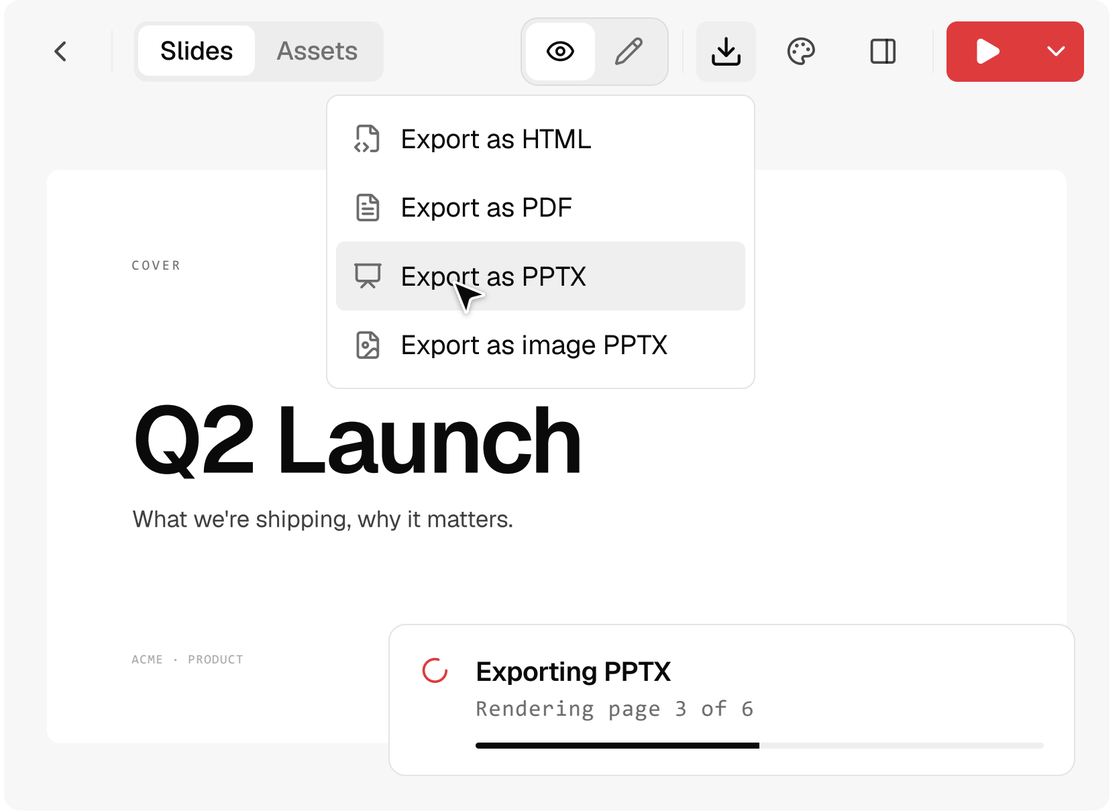
// index-aligned with the pages
```



Ship

Export it. Or deploy it.

Download HTML, PDF, or an editable PPTX from the editor. Or build once and drop dist/ on any static host.



~/my-deck — zsh

```
$ pnpm build
vite v7 building for production…
✓ built in 1.2s → dist/
$
```

Why open-slide

Bring your own agent.

Every workspace ships with agent rules and skills. Slides are plain .tsx, so no SDK, no lock-in.

~/my-deck

```
AGENTS.md  house rules
CLAUDE.md
.agents/skills/
├─ create-slide
├─ slide-authoring
├─ apply-comments
└─ create-theme
.claude/skills/  → .agents/skills
slides/
themes/
```

AGENTS.md

Rules any agent picks up.

Built-in skills

Draft, edit, theme, apply comments.

Plain React

If it edits code, it edits slides.

Why open-slide

Yours. Forever.

Every slide is a file in your repo. Diff it, review it, branch it. No proprietary database, no export-and-pray.

~/my-deck — git log

```
$ git log --oneline slides/q2-launch/

* a1b2c3d (HEAD -> main) refine cover typography
* 9f8e7d6 revise Q2 metrics after finance review
* 6a5b4c3 apply comments from design crit
* 3c2d1e0 initial draft of the Q2 launch deck

$
```

Plain .tsx files

No proprietary format.

Diffable in any tool

Review it like any other PR.

Branch · merge · revert

The tools you already know.

Recap

That's the whole loop.

01

Init

`npx @open-slide/cli init`

02

Prompt

`/create-slide`

03

Edit

`click · tweak · save`

04

Comment

`/apply-comments`

05

Present

`F · fullscreen`

Edit `slides/<your-slide>/index.tsx` and watch it hot-reload.