

Pointless — Study Guide

Overview

Pointless is a presentation platform designed around a simple observation: your audience is holding more computing power than they know what to do with, and traditional slides waste that fact. This guide expands on the philosophical and technical reasoning behind Pointless, examines the specific design decisions that separate it from existing tools, and points you toward further reading on the craft of presentations and the technology that makes them work.

If you attended the talk and want to dig deeper, start with the section that interested you most. If you're reading this without having seen the presentation, the sections are structured so you can follow the argument from start to finish.

The Problem: Passive Audiences

For decades, the slide deck has been the standard delivery mechanism for talks, lectures, and pitches. PowerPoint, Google Slides, Keynote—they all work on the same fundamental model: one person stands in front, controls the slides, and the audience watches.

This model has consequences.

First, the audience is passive. They can't navigate ahead or backward. They can't skip the boring part. They're locked to the presenter's pace, which may be too fast if they're still processing the last slide, or too slow if they got it immediately. The bandwidth is one-way.

Second, the room's attention fractures. If you're sitting in row three and you've already understood the point of the current slide, your phone becomes irresistible. Email, Slack, Twitter—all of it is more interactive than what's on the screen. So you check. By the time the presenter moves on to the next slide, you've lost thread.

Third, the presentation is tethered to the presenter's laptop. The presenter must stand at the podium with a clicker, or pace back and forth with a remote. Move to the back of the room? The slides are too small to read. The technology, which should enhance the message, becomes a constraint on the delivery.

Finally, retention is poor. Studies on learning show that multimedia learning (combining audio, visuals, and text) is more effective than text alone—but only when the audience actively engages with the material. A slide deck presented in real time is multimedia, but passive

consumption of it is not engagement. The audience absorbs what sticks and forgets the rest. If they want to revisit the material, they have nothing but a video of the talk, or a PDF of slides with no speaker notes.

Pointless starts from the premise that this problem is solvable, and that the solution doesn't require asking the audience to install an app or create an account.

The Second-Screen Model

The insight underlying Pointless comes from the gaming platform Jackbox Games. Jackbox released a series of party games—Quiplash, Fibbage, Trivia Murder Party—that operate on a second-screen model: one person (or a small group) plays on a TV or projector; everyone else plays on their phones.

Why does this work so well?

First, the barrier to entry is near-zero. No app install required. You join a game by navigating to a URL or scanning a QR code. By the time you've raised your phone, you're in.

Second, the second screen creates engagement. People aren't passive observers. They're actively participating, even if only in their heads (in Jackbox games, out loud).

Third, the second screen is *already available*. Your audience has brought it to the room. It's in their pocket. Asking them to use it is not friction; it's frictionless.

Pointless borrows this model for presentations. Instead of one person controlling passive viewers, you have a presenter and an audience, and both are equipped. The presenter drives from their phone. The audience sees the same deck on theirs. There's no app. There's no account. There's no friction.

The philosophical shift is important: your audience is not a mass to be lectured to. They're individuals with agency and attention. The second screen acknowledges that.

How Sync Actually Works

The technical implementation is deliberately simple.

The presenter's phone is the source of truth. It holds the current slide number, the slide content, metadata. When the presenter taps "next," that state updates.

Each audience member's phone polls the presenter's phone at roughly 2 Hz (twice per second).

The request is a simple HTTP GET: "What's the current state?" The response is JSON: slide number, title, body, any other metadata the renderer needs.

500ms latency is imperceptible to humans. By the time you've processed the new slide visually, the network has already synced.

There's no WebSocket. No persistent connection. No Firebase or cloud backend. The polling mechanism is dumb, bulletproof, and scales horizontally: a thousand phones polling a single presenter is just a thousand GET requests per second, which any bare-bones server or CDN can handle.

Why polling instead of push notifications? Simplicity. You don't need a service that remembers state across sessions. You don't need to manage subscriptions. You don't need to handle offline scenarios gracefully. Every request is independent; the server is stateless.

Why local sync instead of a cloud backend? Privacy, reliability, and latency. A Cloudflare Worker running at an edge location can serve a request to a phone in the same building faster than a request to a regional data center across the country. And if your internet connection drops, the audience's phones still see the current slide—they just stop updating.

Content as Markdown in Git

Pointless treats a presentation as source code, not a file.

When you write a talk, you create a Markdown file. That file lives in a git repository. It has version history. It can be branched, merged, reviewed, diffed.

Why Markdown? Because it's human-readable and version-control-friendly. A diff of two versions of your talk is legible. You can see what you changed, why, and when. Undo is trivial: revert the commit.

Why git? Because your team already uses it. You know how to collaborate in git. You know how to manage versions. You know how to code review. Treating presentations like code means they inherit all the workflow advantages of code.

There's also a philosophical element. A presentation in PowerPoint is a black box—a binary file that git can't diff, that requires the proprietary software to edit, that lives in OneDrive or Google Drive. A presentation in Markdown is text. Text is portable. Text is transparent. Text can be version-controlled, diffed, and merged.

This also makes it possible to audit a presentation's history. When did you add that example? What was the context? Why did you remove that slide? Git blame and git log answer these questions. For a talk you give multiple times, being able to see your own edits over time is

invaluable.

Static HTML and CDN Deployment

When you finish writing a presentation, Pointless compiles it to static HTML and CSS.

No server-side rendering. No JavaScript runtime. No database queries. Just files.

This has several benefits.

Performance: Static files are fast. A CDN serves them from a location close to the audience. Latency is measured in milliseconds.

Reliability: There's no server to crash. There's no runtime to fail. The files either exist or they don't. A thousand people watching the same deck costs nothing extra—the CDN handles the load.

Portability: You can serve the deck from anywhere. A Cloudflare Pages site. An S3 bucket. A folder on a web server. The built output is just files; it doesn't care where it lives.

Simplicity: No deployment pipeline, no CI/CD, no secrets to manage. Build the files, upload them, done.

This is boring infrastructure. Boring is good. It means you can focus on the presentation, not on ops.

Speaker Notes and Privacy

When you write a Pointless presentation, you can include speaker notes in the source file itself. They're formatted as HTML comments:

```
<!-- speaker-notes
This is a note to yourself.
The audience will never see this.
-->
```

When you build the presentation, these notes are stripped. They never make it to the compiled HTML. They never leave your machine. Your private thoughts stay private.

This is a deliberate design choice. A speaker should be able to write for themselves—reminders, timing cues, the joke that might not land, what to emphasize if time is running short—without worrying about it appearing in the built deck.

It also decouples your rehearsal from the audience's experience. You can develop your talk with extensive notes, then strip them down to nothing before presenting. Or you can keep them minimal throughout. The choice is yours.

The Study Guide

Most presentations are one-shot events. You give the talk once, maybe twice, and then it's done. But what if you want to preserve the idea for people who couldn't attend? What if you want to deepen the learning for people who did?

A study guide does both.

A study guide is a companion document to the talk. It expands on each slide with prose, examples, citations, code, further explanation. It's what you wished you could say in the talk but didn't have time for. It's the reading you did while preparing, curated for someone who wants to understand the topic more deeply.

The study guide is separate from the slides. Slides are constrained: a few bullets, a chart, a code snippet. A study guide is unconstrained. You can go deep. You can explore tangents. You can cite papers and books. You can include long code examples or transcripts.

During a presentation, people can open the study guide on their phone at any time. They don't have to choose between listening to the presenter and taking notes—the study guide is the notes. After the talk, they can export it as PDF or Word, reread it, share it with colleagues.

This transforms the presentation from a one-way broadcast into the beginning of a conversation.

Out of Scope Decisions

Pointless deliberately does not include several features you might expect from a modern presentation tool.

Audience polling: Asking the audience a multiple-choice question and seeing real-time results is appealing. It's not in scope for the initial version. Polling requires more state management, more infrastructure, more complexity. Version 2.0 might include it.

Live interaction: Slide elements that respond to audience interaction (e.g., buttons that only certain people can click) are out of scope. Again, complexity.

Animations and transitions: Pointless slides are static. No slide animations, no text reveals,

no fancy transitions. This sounds like a limitation, but it's a choice. Animations in slides often distract from the content. Static slides are faster to author, faster to compile, faster to load.

Themes and customization: The initial version ships with one theme (overwhelm) and limited customization. This will expand, but initially, the constraint forces good defaults.

These decisions reflect a philosophy: do one thing well. Pointless syncs a slide deck across devices in the same room. It does that elegantly. Everything else is a distraction.

Further Reading

If you want to go deeper on the topics covered in this guide, here are some starting points.

On Presentations and Speaking

- Edward Tufte's essay "[The Cognitive Style of PowerPoint](#)" is a foundational critique of how PowerPoint encourages bad presentations. His argument is that slides shouldn't be reduced outlines but should contain the full argument—something Pointless enables by keeping speaker notes separate.
- David Larson's [Presentation Zen](#) teaches design principles for slides that apply whether you're using Pointless or not. Emphasis on simplicity, storytelling, and visual clarity.

On Second Screens and Engagement

- Jackbox Games' [design philosophy](#) for party games offers insight into why second screens work. Their games require no installation and minimal friction to join, principles Pointless borrows.

On Web Standards and Static Sites

- [Reveal.js](#) is an open-source HTML presentation framework. Unlike Pointless, it renders in the browser with JavaScript, but it demonstrates the viability of web-native presentations.
- The [JAMstack philosophy](#) (JavaScript, APIs, Markup) emphasizes static site generation and content delivery from CDNs. Pointless aligns with these principles.

On Cloudflare and Edge Computing

- [Cloudflare Pages](#) is a platform for deploying static sites with automatic builds and global CDN distribution. This is one option for hosting a Pointless deck.
- [Cloudflare Workers](#) enables serverless functions at edge locations worldwide. If Pointless needs a backend (e.g., for presenter state management), Workers is a lightweight option.

On Version Control and Workflows

- [Pro Git](#) by Scott Chacon and Ben Straub is the definitive guide to git. Understanding git deeply makes the "content as code" approach to presentations much more powerful.
-

Closing: Why Build This

Pointless is built on a bet: that presentations are better when the audience is equipped to engage, when the content is portable and version-controlled, and when the technology gets out of the way.

The slide deck won't disappear. But the next generation of slide decks might work differently—starting with the second screen everyone already has in their pocket.