

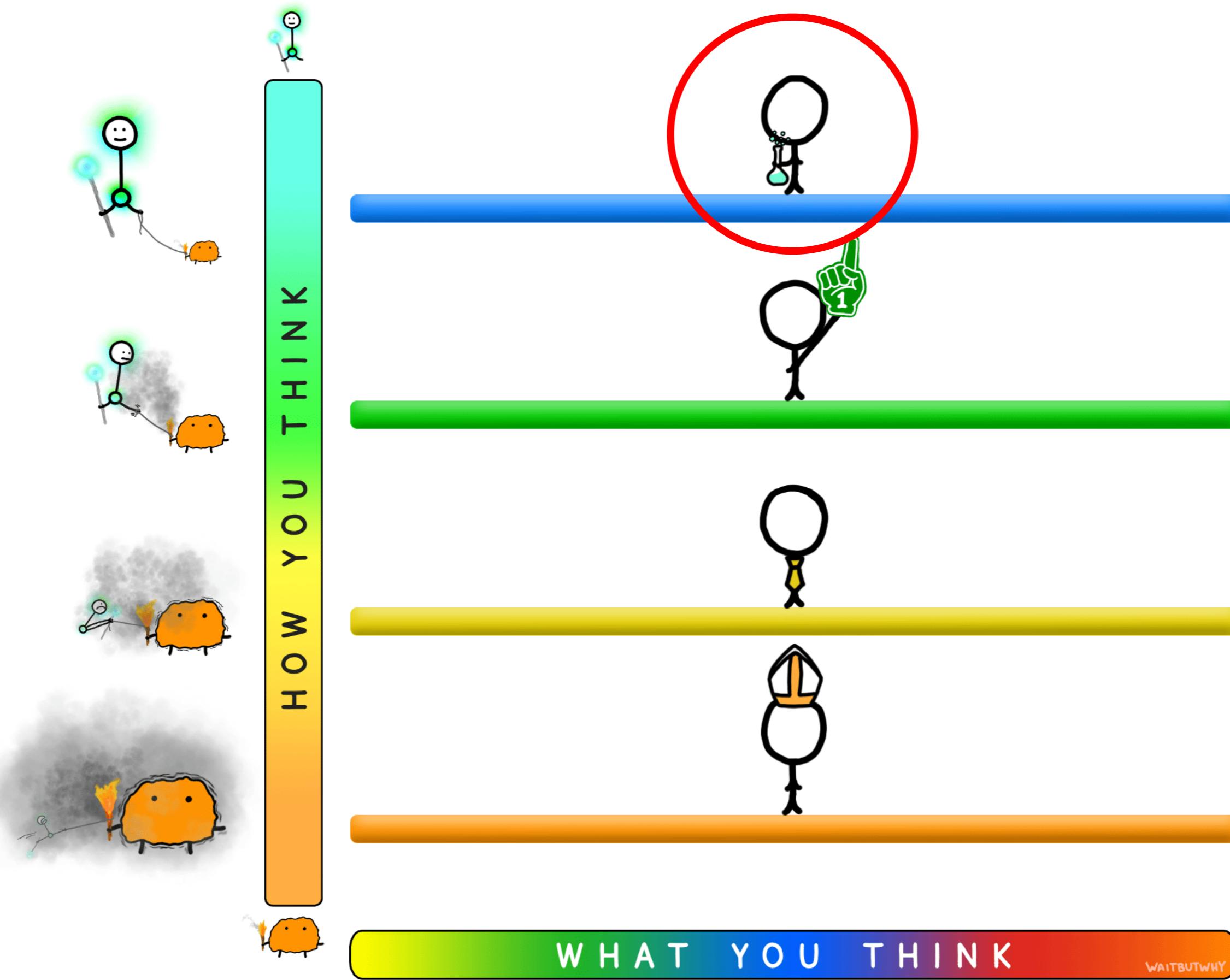
A dense cluster of colorful umbrellas, including shades of pink, red, green, yellow, orange, and blue, are scattered across a clear blue sky. The umbrellas overlap and point upwards, creating a sense of depth and variety.

I JamStack

BUT

A large, intense fire with orange and yellow flames against a black background.

Let's Get Real



The Problems with Static Architecture

1

Unbounded Page Growth

A Static Site Axiom

The more unique pages your static site has, the slower it compiles.

As your organization grows, you will continue adding more unique pages.

Therefore as your organization grows, your site's compilation time gets slower.

The more unique pages your static site has, the slower it compiles.

But, Hugo! - Hugo is great & fast but this is still true

But, Gatsby Cloud! - This is a paid, remote only workaround

As your organization grows, you will continue adding more unique pages.

But, we have no blog! - Targeted marketing, custom audiences...

But, it will be a long time! - Ok, still true

But, we won't grow! - Then you don't have scale problems

2

Dynamic Functionality

Some Examples

Forms

Email Signups

Clearbit Auto Form Fill

User Login

API routes to protect access tokens

API routes to combine different api sends & responses

IP-based location lookup on request

How To Fix This

- 1 Pay a third-party service provider
- 2 Make it yourself, run a server with an API

3

Content Preview

4

Fast-changing, Dynamically
Rendered
Public Pages

A dark, moody photograph of a man sitting at a desk, looking intensely at a laptop screen. He has his hand to his forehead in a gesture of stress or exhaustion. The laptop is covered in various developer stickers, including ones for CSS, PHP, Node.js, and Git. The background is dimly lit, suggesting a late-night work session.

The Stress of Web Architec Decisions: **The Game!**

All-in Dynamic

- ✓ no limits
- ✓ no refactors later
- ✗ complexity when not needed
- ✗ slower time to market
- ✗ more expensive

Worst Case Scenario

Competitors beat you to market with a simpler, faster option that costed them less. Your resources are drained due to the higher costs and complexity.

All-in Static

- ✓ fast, easy start
- ✓ low cost & complexity
- ✗ limitations as discussed
- ✗ getting around them is brutal

Worst Case Scenario

As your product matures, you need crazy workarounds to the limitations of static, which are more expensive and cause more velocity reduction than a dynamic build, competition crushes you over time by being more efficient.

Start Static, Refactor to Dynamic

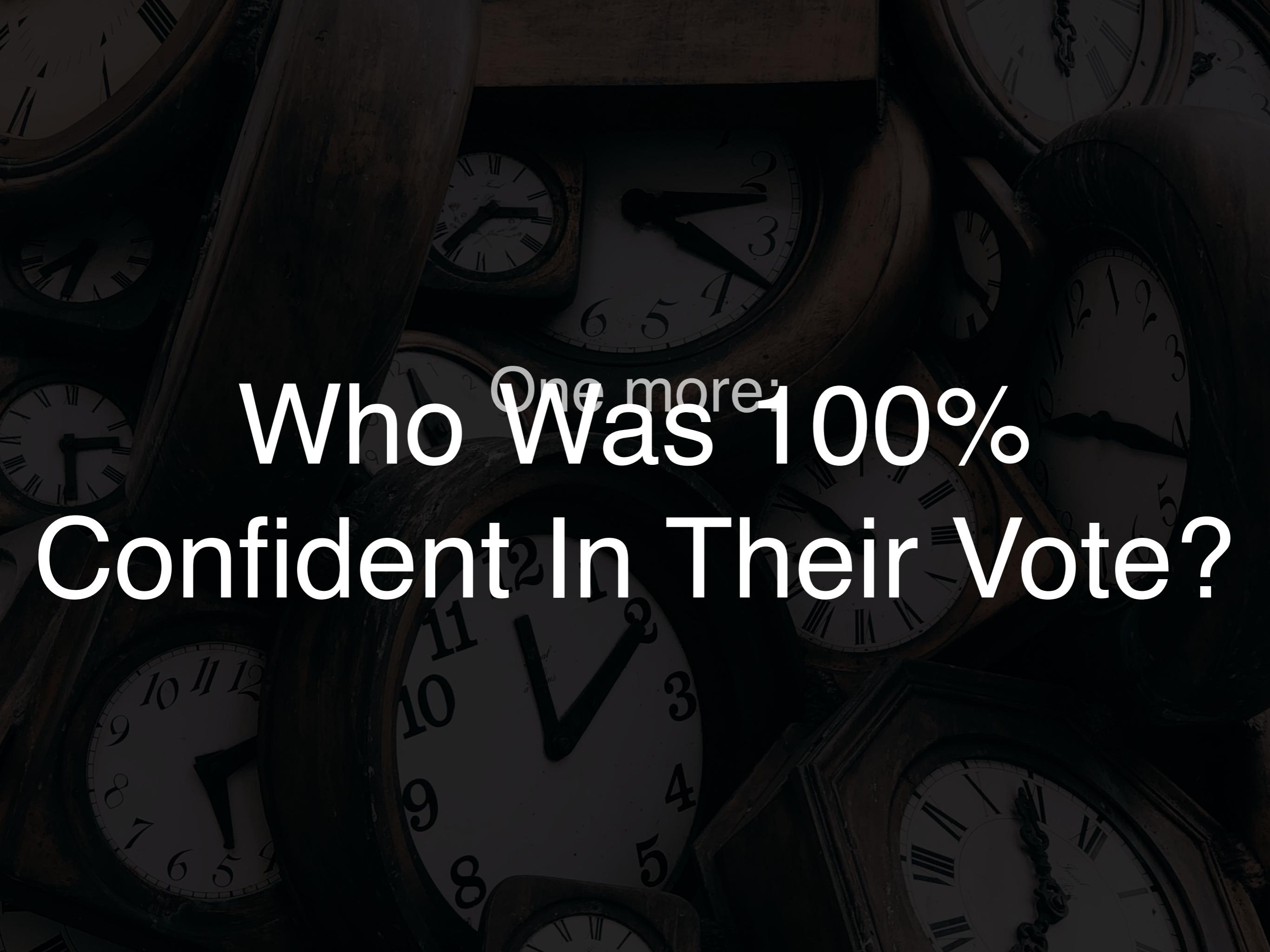
- ✓ no limits
- ✓ fast, easy start
- ✗ enormously expensive refactor

Worst Case Scenario

You make it to market and start maturing, but your growth stalls and expenses skyrocket as you begin to refactor, drains your resources and competition outpaces you drastically.



Time for a vote:
Which Do You Pick?



One more:

Who Was 100% Confident In Their Vote?



These Options Are All
Awful

Let's be real.

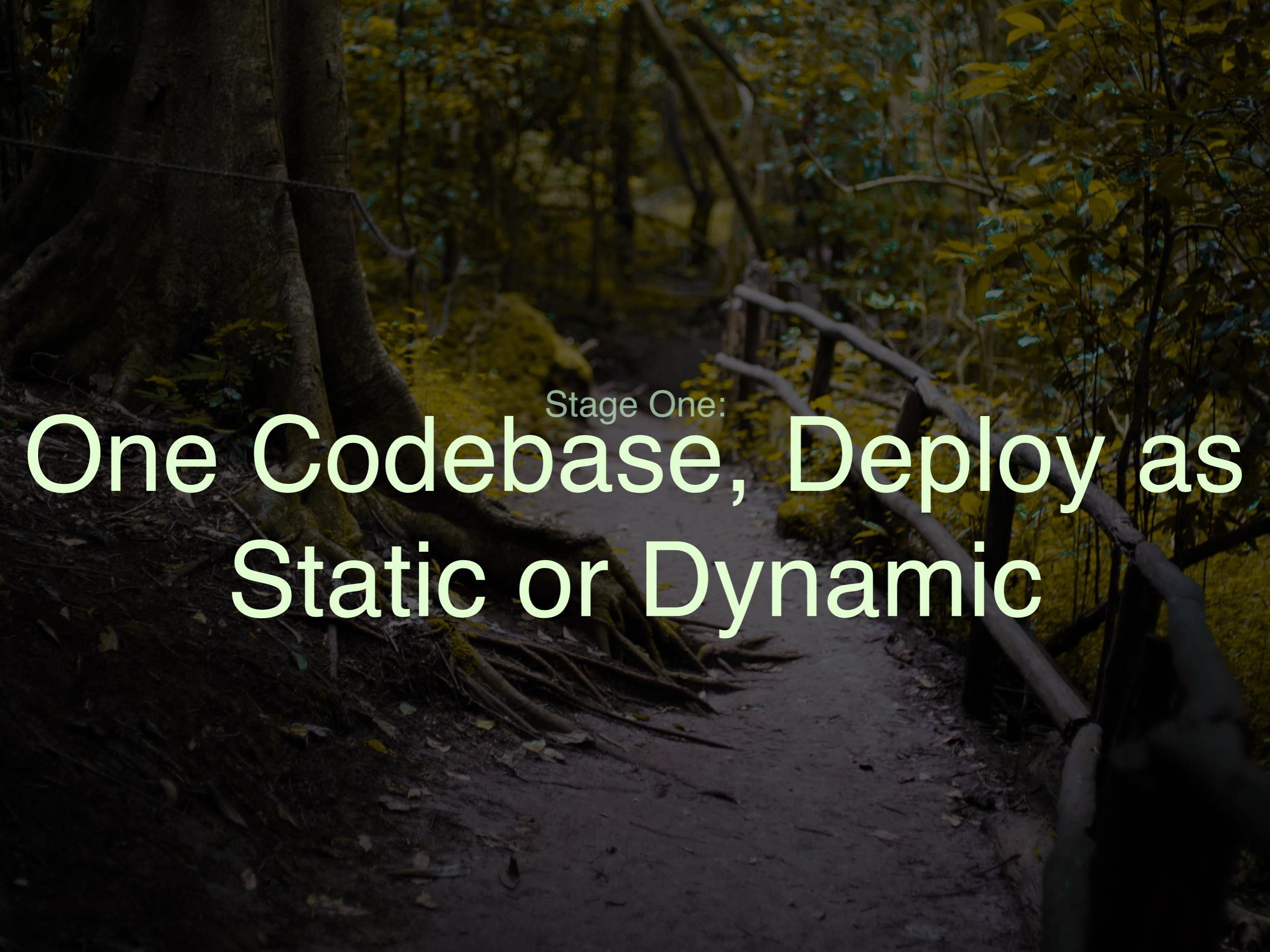
The Solution



~~NE~~X~~T~~.JS

A dark, atmospheric forest scene with a path winding through trees. The path is covered in fallen leaves and branches. The background is filled with dense foliage and tree trunks.

You Dont Need to Choose
The Plan:
Static or Dynamic



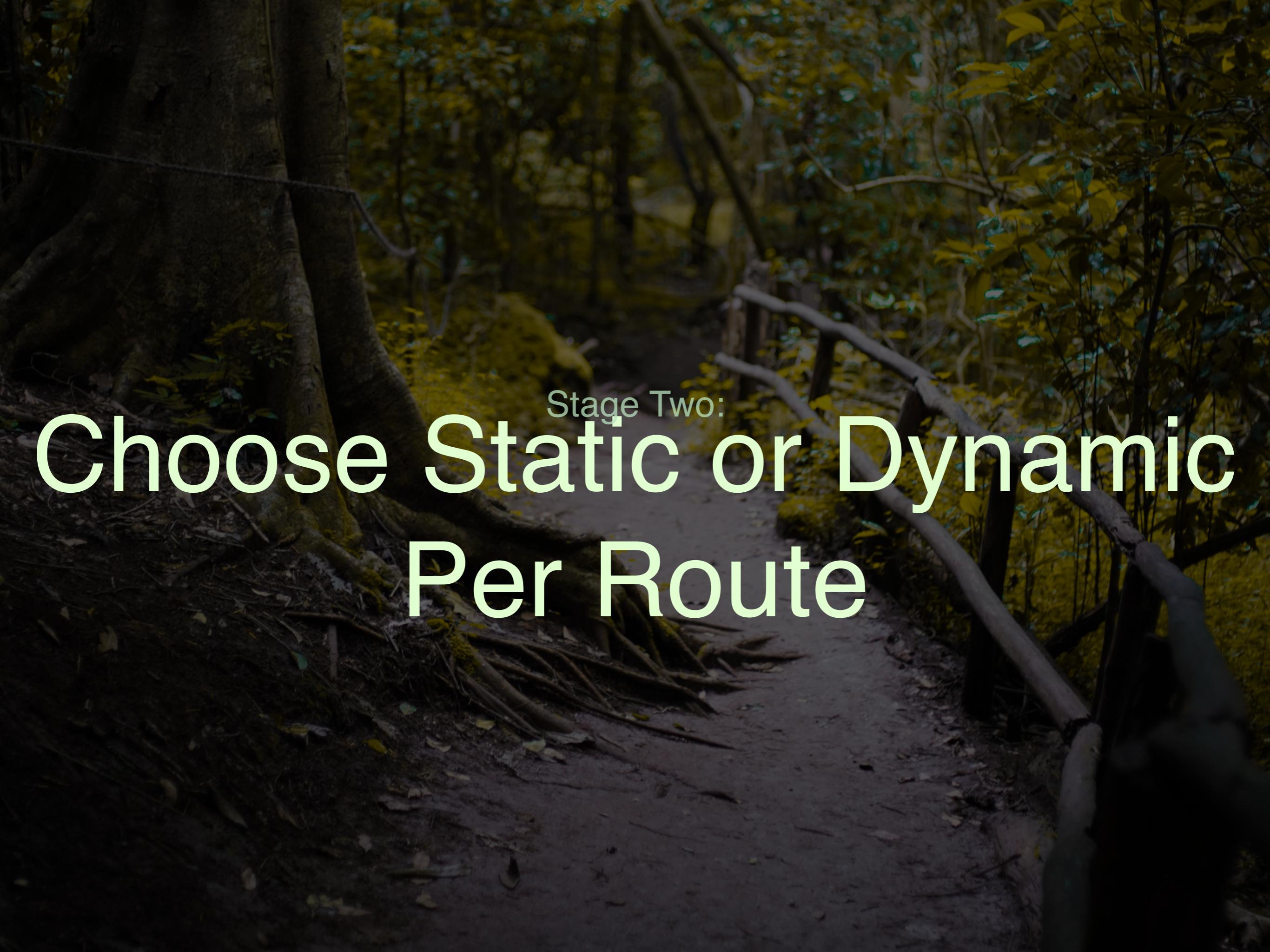
One Codebase, Deploy as Static or Dynamic

Stage One:

The background of the image is a dark, atmospheric scene of a stage performance. A large crowd of people is visible in the foreground, looking towards the stage. The stage is illuminated by several bright, glowing structures that resemble stylized letters or geometric shapes, creating a futuristic and dynamic visual effect.

Example:

 HashiCorp.com

A photograph of a dirt path winding through a dense forest. The path is covered in fallen leaves and branches. Several large, fallen trees are scattered across the ground, some leaning at angles. The surrounding trees have dense foliage, with sunlight filtering through the canopy.

Stage Two: Choose Static or Dynamic Per Route

getStaticProps

getServerProps

getStaticPaths



Static
Limitations, Be
Gone!

MULTI-THREADED EXPORT

Unbounded Page Growth

SPLIT BABEL COMPIRATION & EXPORT

1

SWAP ANY ROUTE TO DYNAMIC

“API” FOLDER



FULL SERVER RENDER

LAMBDA DEPLOYMENT

Dynamic Functionality

DEPLOY AS DYNAMIC

3

Content Preview

EVEN MORE GOODNESS SOON!

4

Fast-changing, Dynamically
Rendered
USE GETSERVERPROPS
Public Pages

Table Stakes

- ✓ Route-based code splitting, including css
- ✓ Crazy webpack bundle optimization
- ✓ SSR setup out of the box
- ✓ Load data in any way, from any source
- ✓ Client or server side routing, filesystem-based
- ✓ Automatic prefetching
- ✓ Easily customize with config & plugins
- ✓ Automatic typescript config

I



NextJS

BUT

Every Tool Has Its Weaknesses

- ✓ Growing companies ✗ Personal websites
- ✓ Venture-funded companies ✗ Small/medium blogs
- ✓ Web applications ✗ Small business website
-
- ✗ Informational website

At Least One Full Time Web De

NextJS is changing fast

It is way more complicated than
straight html/css/js

Not even close to future-proof

Thank You!

