KEEPING SSR COOL WITH REACT HYDRATION

MONICA POWELL

React Rally 2020

👩🏾‍💻 www.monica.dev | 🐳 @waterproofheart

HI, I'M MONICA 👋🏾

I'm a software engineer who enjoys building technology that elevates people whether that's increasing access to e-books, creating tools for Meetup's community organizers or sending curated career opportunities to diverse job-seekers at scale. I'm also passionate about making open-source more accessible and recently became an inaugural GitHub Star 🌟

👩🏾‍💻 www.monica.dev | 🐳 @waterproofheart

OVERVIEW

The purpose of this talk is to share some helpful things to keep in mind to render a seamless experience as a Server-Side Rendered (SSR) site transitions from a window-less (server) environment to a browser.

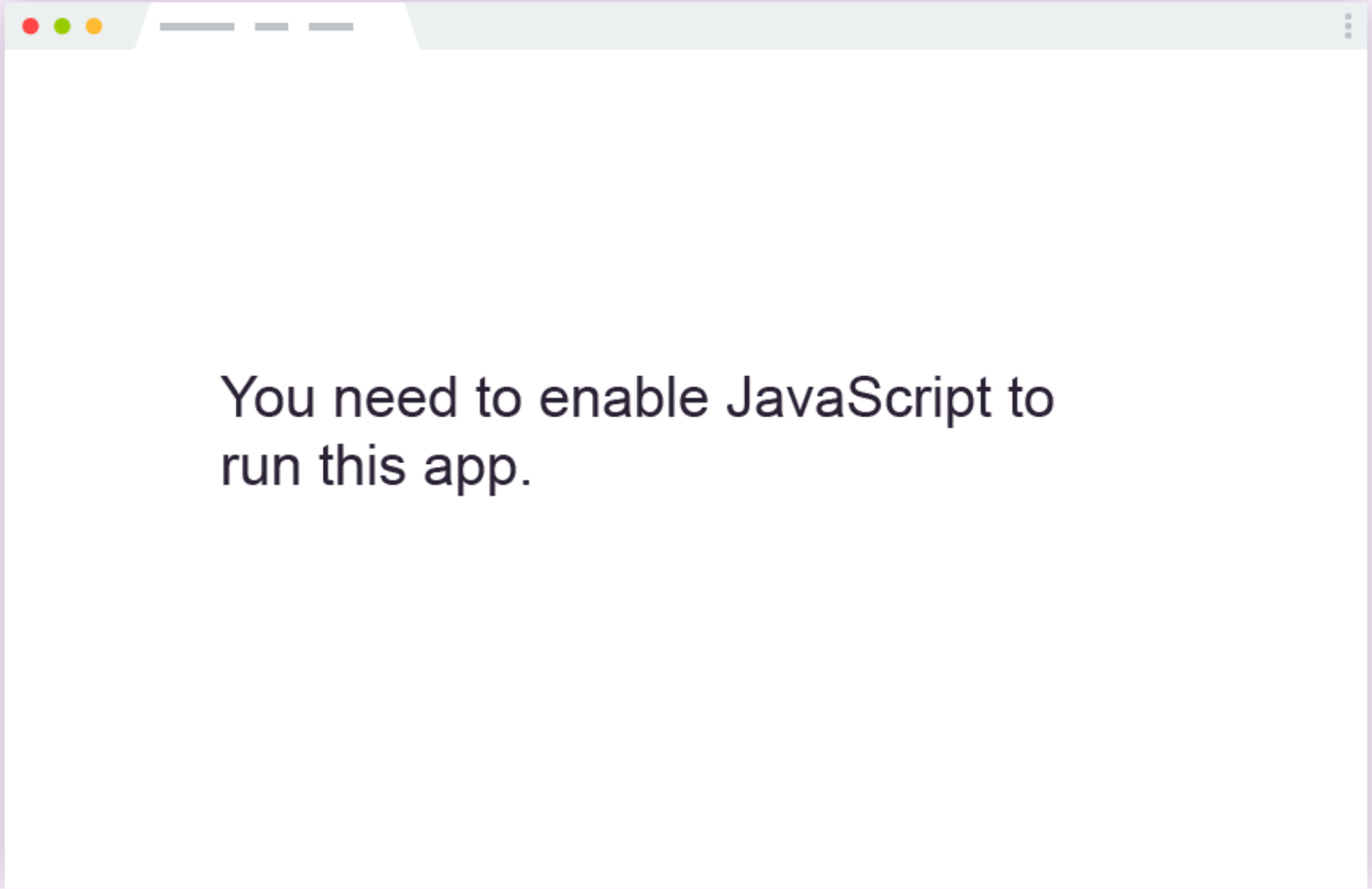
WHAT IS SERVER-SIDE RENDERING (SSR)?

A server generates the initial HTML that loads in a browser. Frameworks like NextJS and GatsbyJS support SSR out-of-the box

WHY SERVER-SIDER RENDERING (SSR)?

SSR apps tend to have faster initial loading times and better SEO than client-side only apps.

WHAT IS CLIENT-SIDE RENDERING (CSR)?



CLIENT-SIDE RENDERED INITIAL DOM

```
<html>
  <head><!-- SEO/Metadata here --></head>
  <body>
    <div>You need to enable JavaScript to run this app.</div>
    <div id="root"></div>
    <script><!-- all of the JavaScript --></script>
    <script src="/static/js/2.6158a3d8.chunk.js"></script>
    <script src="/static/js/main.ba831a9f.chunk.js"></script>
  </body>
</html>
```

OVERVIEW OF SSR (IN STATIC CONTEXT)

Write site in React

Create build for production using ReactDOMServer, a server-side React API to generate HTML from React.

On Initial Load HTML generated from the server is displayed

ReactDOM.hydrate(), hydrates the HTML page rendered from the server with JavaScript.

TOGGLING JAVASCRIPT

SSR vs CSR

ww.aboutmonica.com

NICA POWELL

are engineer, content creator &
unity organizer

Monica! I'm a software engineer who is passionate about making open-source more acces
technology to elevate people, and building community. You can find me teaching web
ment on **Egghead**, contributing to open-source projects like **NYPL-Simplified** and Ga
g to **React Ladies**, a community I founded for women and non-binary React developer

t Writing

View All Writing

ow To Create A GitHub Profile README

ploring Art Direction With Gatsby-Image

tial Thoughts On Migrating from gatsby-transformer-remark to gatsby-plugin-mdx

ve Your CSS Superpowers with CSS Variables

www.monica.dev

eysanatomyloremipsum.netlify.app

Grey's Anatomy Lorem Ipsum Generator

How many paragraphs should be generated?

4

Generate Lorem Ipsum

Grey's Anatomy Lorem Ipsum Generator

By: Monica Powell • View Code

@waterproofheart

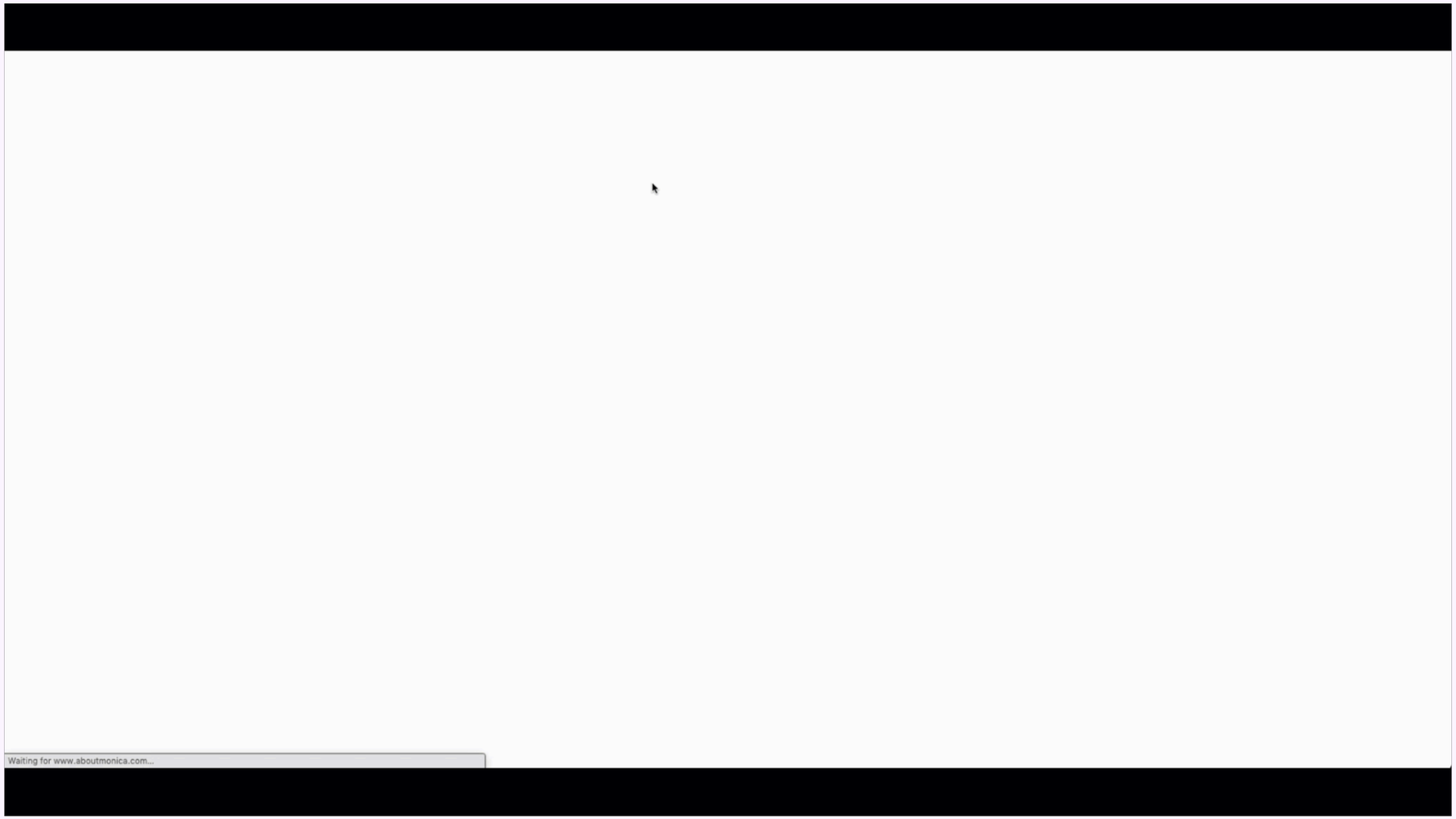
IT LOOKS GREAT IN DEVELOPMENT...WHAT COULD GO WRONG? 😅

- Layout shifts that only appear in production
- Errors that only appear at build-time

MISSING DATA ON SERVER-SIDE ⚠️

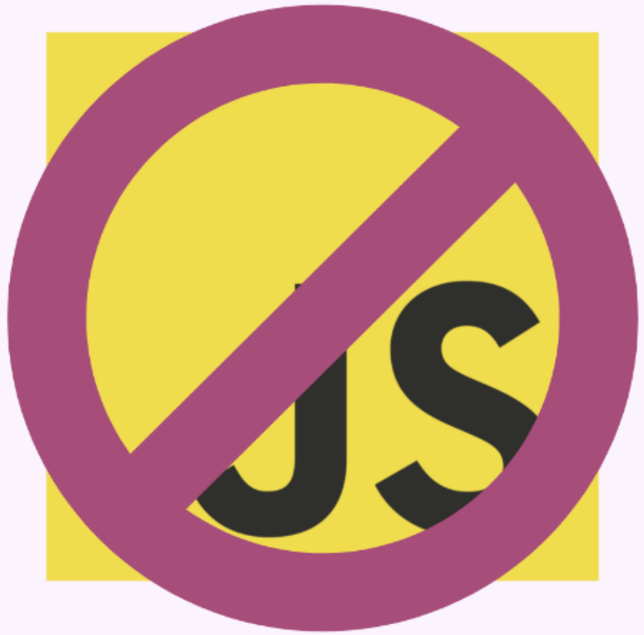
- User or browser-specific data is not available in the server when static HTML for the site is generated (i.e., window size, authentication status, local storage etc)
- Static-site-generation (SSG) is when all of the pages for a site are generated at build time

DEBUGGING SSG HYDRATION ISSUES

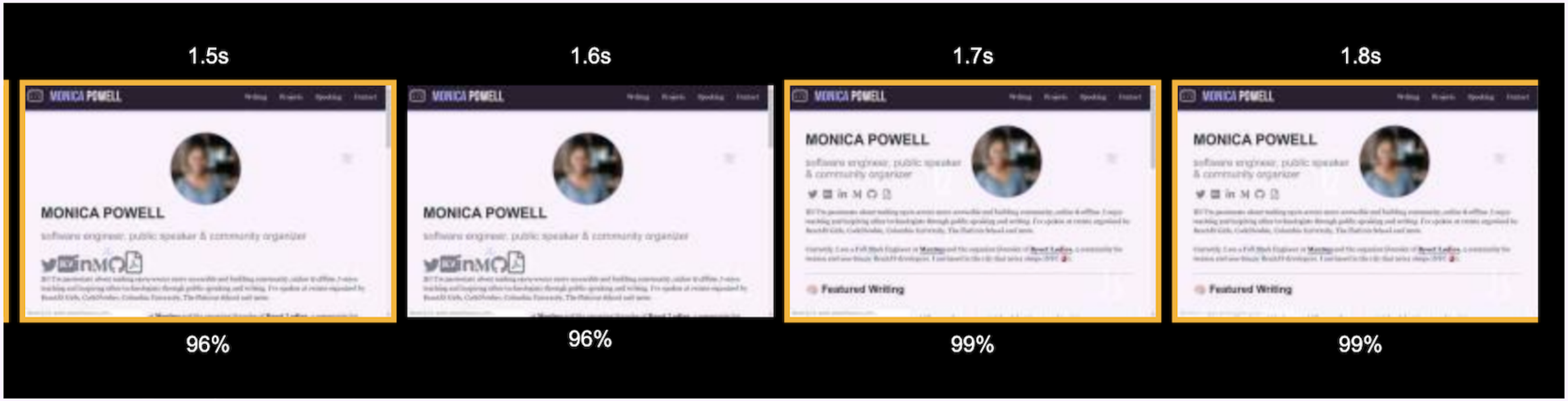


DEBUGGING SSG HYDRATION ISSUES 🐞

- Disable JavaScript
- View filmstrips to see how the website hydrates



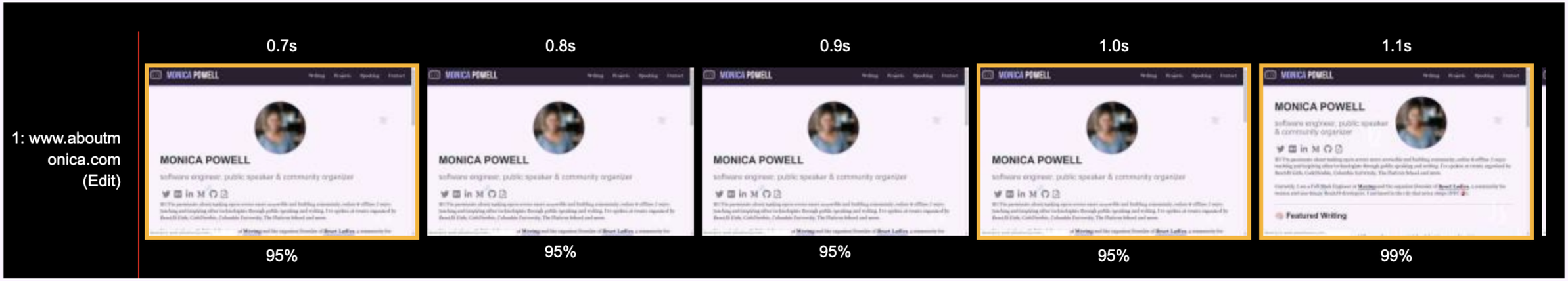
Before: Styled icons with CSS from Font Awesome NPM Package

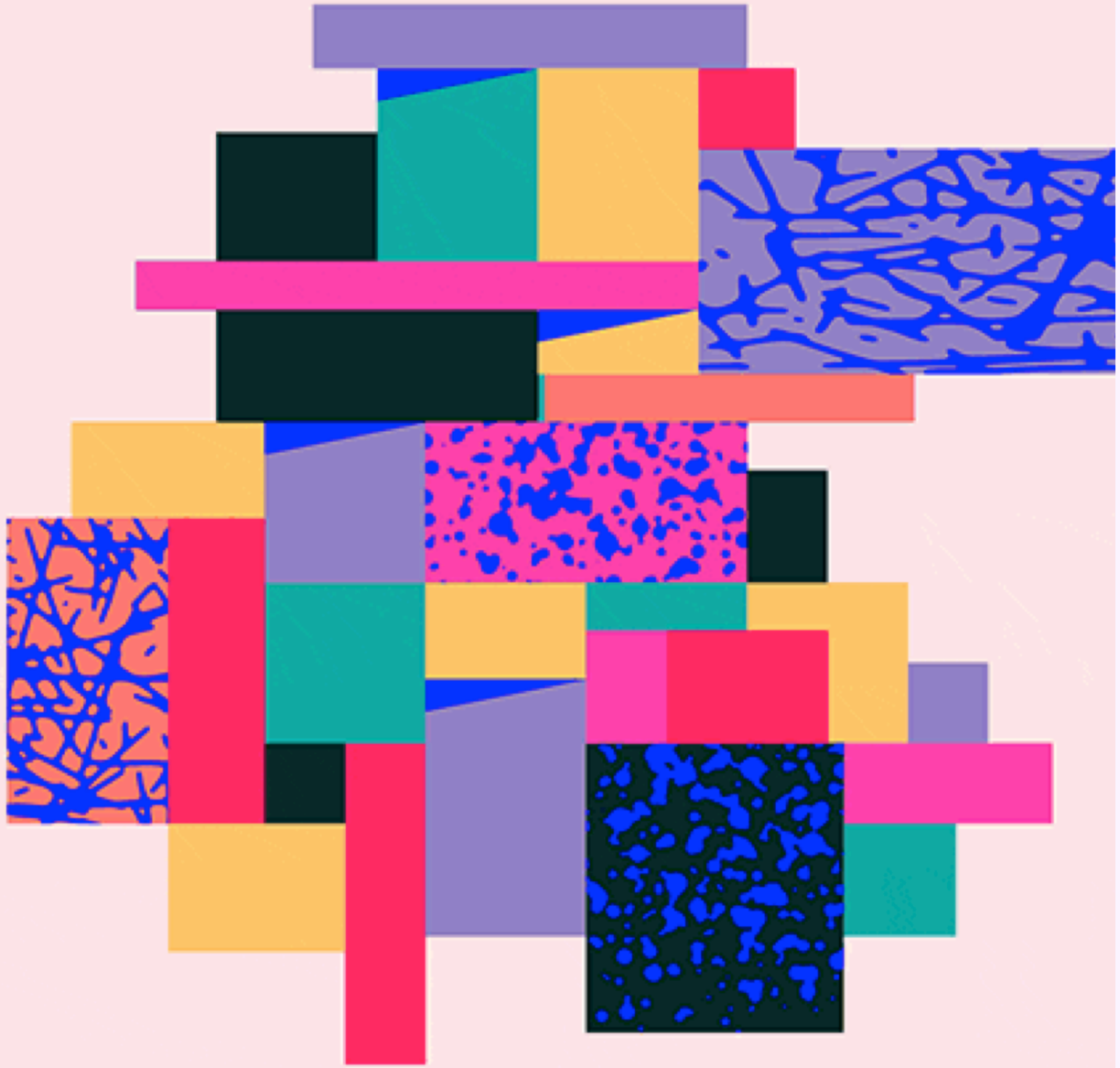


SOLUTION FOR ICONS RESIZING ON LOAD

- Replicate the final styling with local CSS without relying on FontAwesome's external CSS which required JavaScript to be applied

After: Styling icons locally and disable Font Awesome's CSS



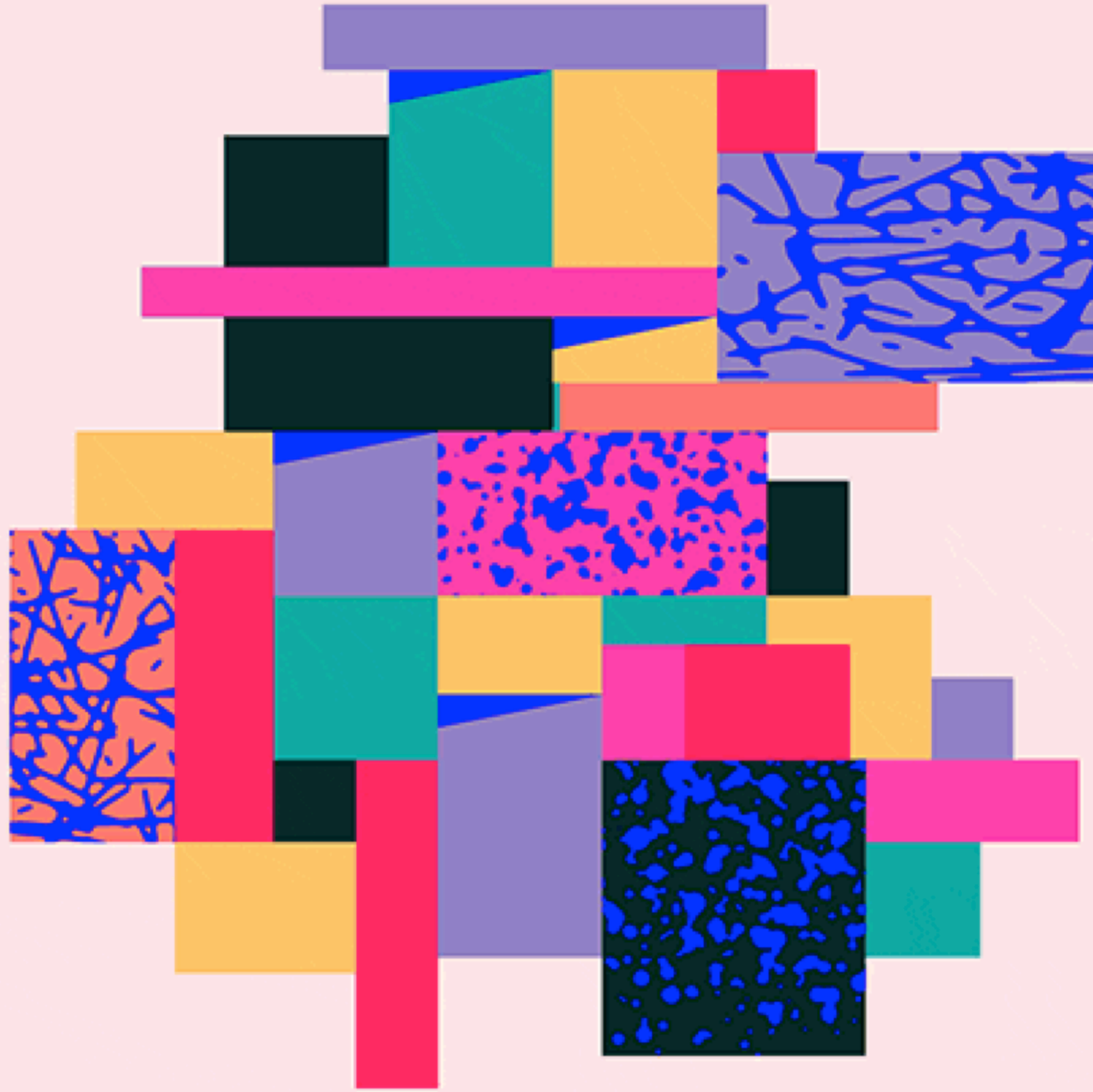


IMMUTABLE LAYOUT

- Avoid unnecessary layout shifts during page load

IMMUTABLE LAYOUT

- Implement layouts with placeholder/gap for expected client-side content
- Use CSS instead of JS to handle the layout of the page



CONDITIONAL RENDERING

```
if (small) {
  return <MobileApp />
} else {
  return <DesktopApp />
}
```

MATCHMEDIA IN SERVER CONTEXT ⚠️

- With the `matchMedia()` Web API you can't reliably detect the browser size in the server which leads to strange layout shifts when the initial positioning is incorrect.

CSS IN SERVER CONTEXT 

- Use CSS media queries directly or a library like arts/fresnel that wraps all `Media` components in css.

```jsx
import React from "react"
import ReactDOM from "react-dom"
import { createMedia } from "@artsy/fresnel"

const { MediaContextProvider, Media } = createMedia({
  breakpoints: {
    sm: 0,
    md: 768
  },
})

const App = () => (
  <MediaContextProvider>
    <Media at="sm">
      <MobileApp />
    </Media>
    <Media greaterThan="sm">
      <DesktopApp />
    </Media>
  </MediaContextProvider>
)

ReactDOM.render(<App />, document.getElementById("react"))
```

ERROR: WINDOW IS UNDEFINED

ERROR: WINDOW IS UNDEFINED

When building a site you might run into the `window is undefined` or `document is undefined` error. This happens when logic within an app assume the browser window is defined in a server

ERROR: WINDOW IS UNDEFINED

Your first inclination to resolve the undefined Window error might be to write something like:

```
typeof window !== undefined ? // render component : // return null
```

REACTDOM.HYDRATE CONSTRAINTS

ReactDOM.hydrate:

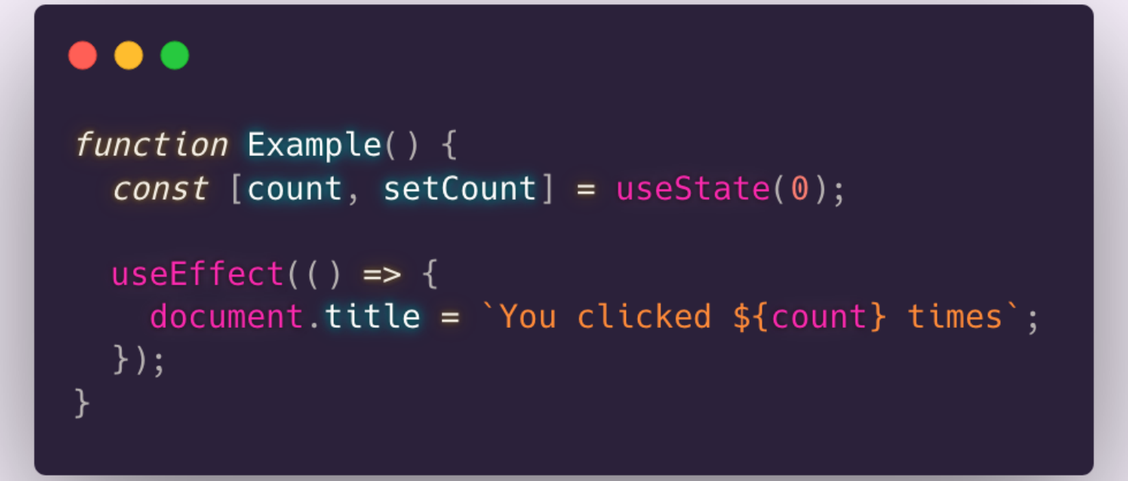
👯 expects that the rendered content is identical between the server and the client.

🙅🏾‍♀️ does not guarantee that attribute differences will be patched up in case of mismatches.

SAFELY ACCESS BROWSER ELEMENTS

- Avoid reconciliation error when ReactDOM.hydrates a site from HTML -> React.
- Wrap React that can only render when Window or document is defined in useEffect which only fires **after** the component has mounted.

USEEFFECT EXAMPLE



```
function Example() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    document.title = `You clicked ${count} times`;
  });
}
```

(example from React docs)

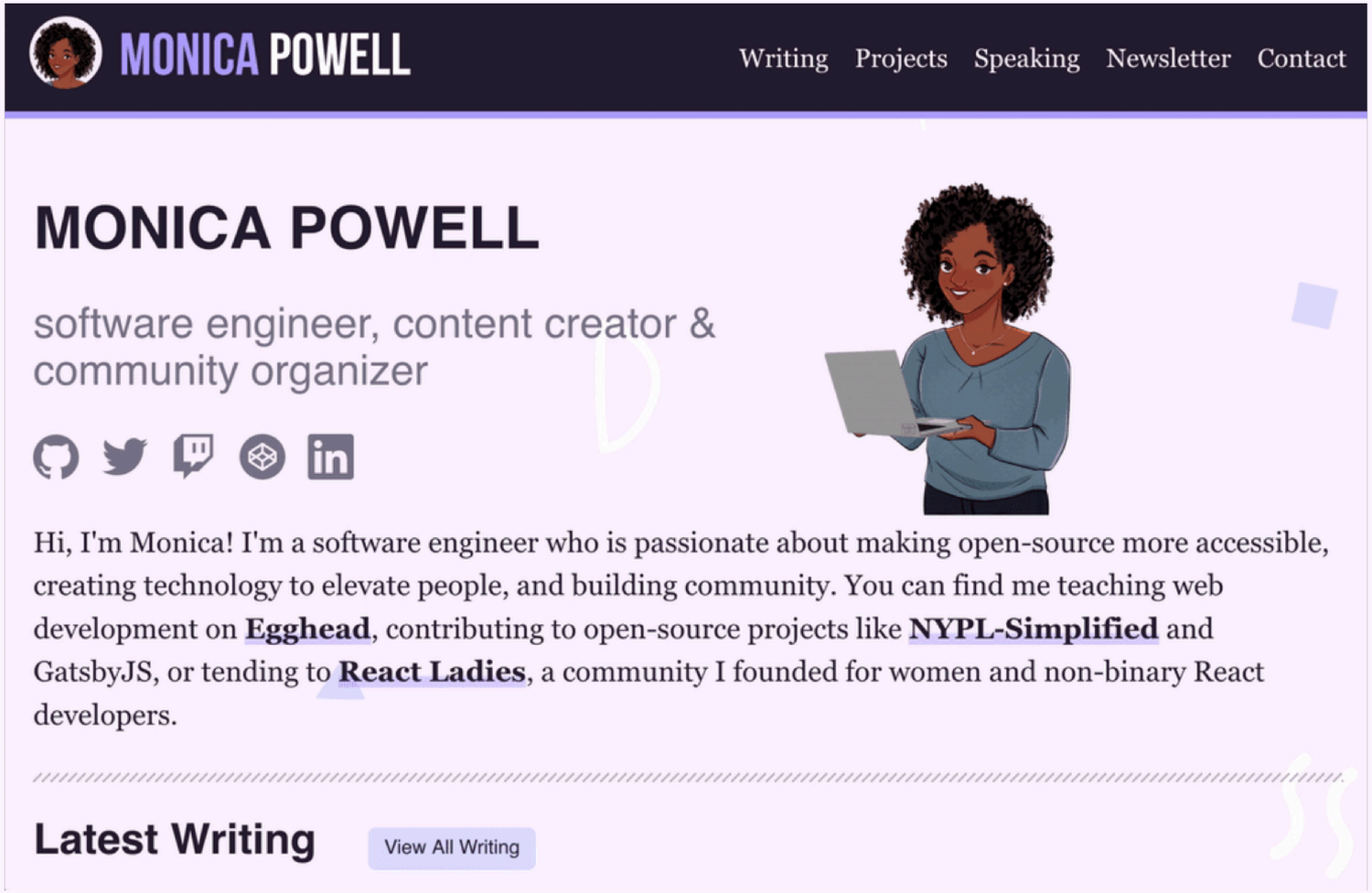
"JavaScript is a powerful language that can do some incredible things, but it's incredibly easy to jump to using it too early in development, when you could be using HTML and CSS instead...

-- Iain Bean, Your blog doesn't need a JavaScript framework

...Consider the rule of least power: Don’t use the more powerful language (JavaScript) until you’ve exhausted the capabilities of less powerful languages (HTML)."

-- Iain Bean, Your blog doesn’t need a JavaScript framework

ART DIRECTION: RULE OF LEAST POWER



ART DIRECTION: HTML OR JAVASCRIPT?

- HTML art direction can be used to dynamically load of images based on screen size using srcset attributes with `<img>` and `<source>` instead of JavaScript.

ART DIRECTION: WHY HTML?

- Swapping images at different screen sizes can be done with JavaScript or CSS instead of native HTML attributes however using HTML can improve page loading performance as it prevents unnecessarily preloading two images.

ART DIRECTION IN HTML

```
<picture>
  <source media="(min-width: 625px)" srcset="animonica-full.png" />
  <source srcset="animonica-headshot-cropped.png" />
  <img src="animonica-full.png" alt="Illustrated Monica" />
</picture>
```

```
import { useStaticQuery, graphql } from "gatsby"
import Img from "gatsby-image"

const Avatar = () => {
  const data = useStaticQuery(graphql`
    query {
     // queries to get mobileImage and desktopImage
    }
  `


const sources = [
    data.mobileImage.childImageSharp.fluid,
    {
      ...data.desktopImage.childImageSharp.fluid,
      media: `(min-width: 625px)`,
    },
  ]

  return ( <Img fluid={sources} alt="image description" />)
}

export default Avatar
```

SUMMARY

- In a Server-Side Rendered context it's important to consider **how** the page loads and when data is or is not available.
  - CSS is the right tool for handling layout.
  - Guard references to browser specific elements like `document` or `window` within `useEffect()` to avoid reconciliation error as the page hydrates

RESOURCES + FURTHER READING

- https://www.gatsbyjs.org/docs/react-hydration/
- https://joshwcomeau.com/react/the-perils-of-rehydration/
- https://reactjs.org/docs/reconciliation.html
- https://www.webpagetest.org/
- https://github.com/artsy/fresnel

THANK YOU!

SEE YOU IN CYBERSPACE 👋🏾

@waterproofheart

👩🏾‍💻 www.monica.dev/reactrally