Considering Single Page Applications

Sebastian De Deyne
Backend-turned-frontend developer at Spatie

Dealing with JavaScript

- Laravel app + some vanilla JS
- Laravel app + Vue.js
- Laravel API + Single page app

Considering Single Page Applications

- Frontend development
- JavaScript frameworks
- Web Performance
- Developer experience
- The history of the web

info.cern.ch/hypertext/WWW/TheProject.html

World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
: Pointers to the world's online information, subjects , W3 servers, etc.

Help
: on the browser you are using

Software Products
: A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)

Technical
: Details of protocols, formats, program internals etc

Bibliography
: Paper documentation on W3 and references.

People
: A list of some people involved in the project.

History
: A summary of the history of the project.

How can I help ?
: If you would like to support the web..

Getting code
: Getting the code by anonymous FTP , etc.

The project started with the philosophy that much academic information should be freely available to anyone.

Tim Berners-Lee

Information

1991

Presentation

1994

Interactivity

1996

Asynchronicity

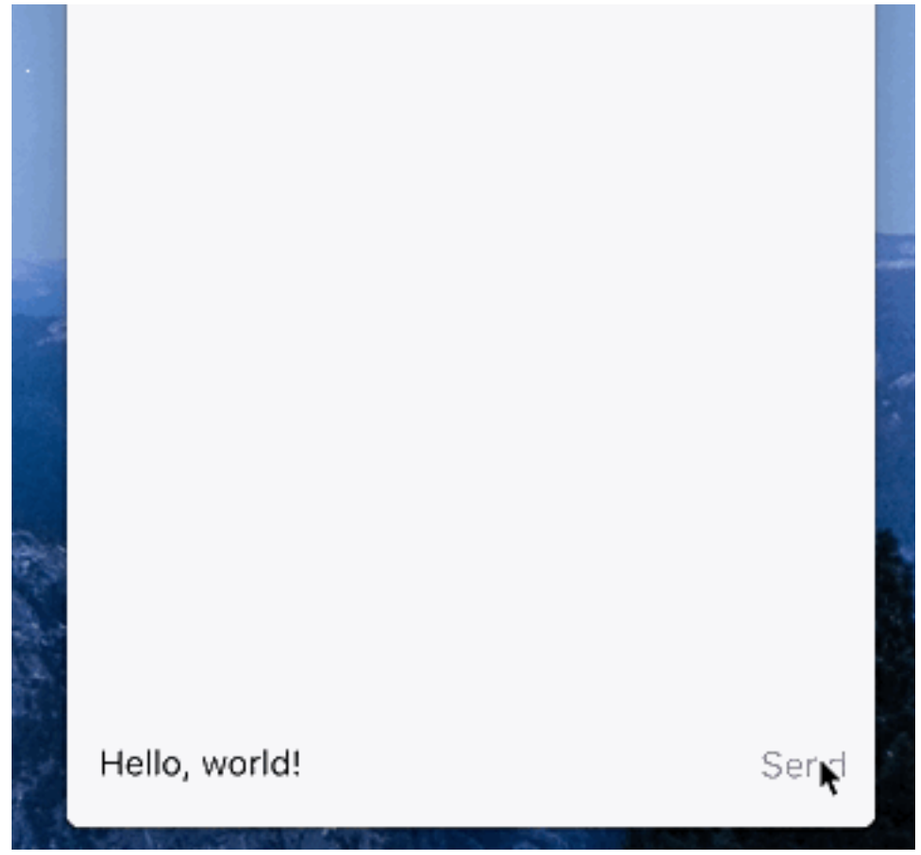
2005

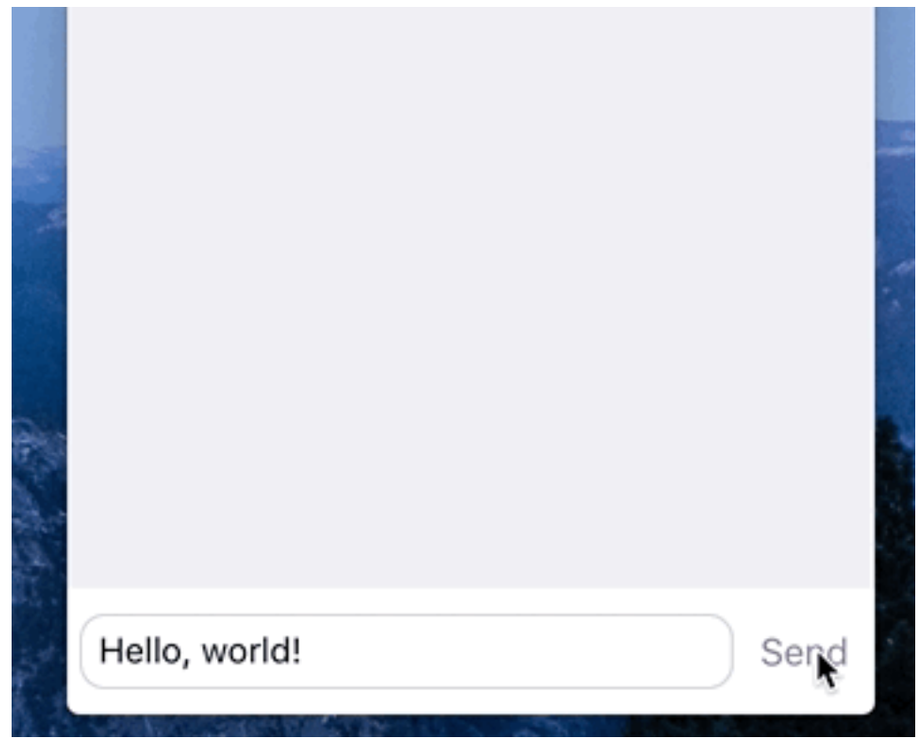
We use new rendering methods to make them [the maps] easier to read. Click and drag the map to view the adjacent area dynamically - there's no wait for a new image to download

Google Maps announcement post

JavaScript & AJAX enabled
a next level of user experience

Optimistic design


Hello, world!
Send


Hello, world!
Send

A document is a static sheet of information.

An app is a long-running process.

Yehuda Katz
@wycats

Follow

Front end software development is:

- real-time (instant load, 60fps)
- distributed, incremental (synchronize remote data as needed)
- asynchronous
- reactive (react to user actions in realtime)

Front end is the hardest kind of dev I do. The folks who do it every day are heroes.

7:53 AM - 14 Nov 2017

1,777 Retweets **4,742** Likes

91 1.8K 4.7K

Somewhere down the road, web development became hard

Interactivity and design are enhancements for *documents*

Interactivity and design are first class citizens for *apps*

We're *(ab)*using the web for two completely different paradigms

info.cern.ch/hypertext/WWW/TheProject.html

World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
: Pointers to the world's online information, subjects , W3 servers, etc.

Help
: on the browser you are using

Software Products
: A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)

Technical
: Details of protocols, formats, program internals etc

Bibliography
: Paper documentation on W3 and references.

People
: A list of some people involved in the project.

History
: A summary of the history of the project.

How can I help ?
: If you would like to support the web..

Getting code
: Getting the code by anonymous FTP , etc.

We can't afford a sudden shift in web technologies

In the web's current state

- Building something is easy
- Building something great requires effort

```html
<img src="cow.jpg" alt="Cow grazing grass">
```

```html
<img srcset="cow-320w.jpg 320w,
             cow-480w.jpg 480w,
             cow-800w.jpg 800w"
     sizes="(max-width: 320px) 280px,
            (max-width: 480px) 440px,
            800px"
     src="cow-800w.jpg"
     alt="Cow grazing grass">
```

The web in 2018

- Is complicated
- Sites resemble apps, not documents
- Things are interactive & real-time
- Apps that feel like native apps
- Apps can work offline

UI complexity means embracing JavaScript

Single Page Applications

A single page application is a JavaScript program that contains your entire webapp

Your entire application is contained in a single script

Network speed & computing power

Bottlenecks

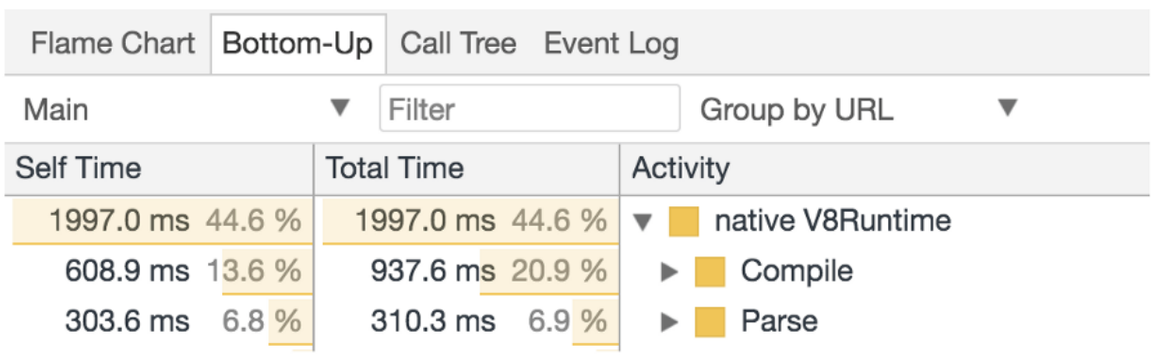
JavaScript bytes !== JPEG bytes

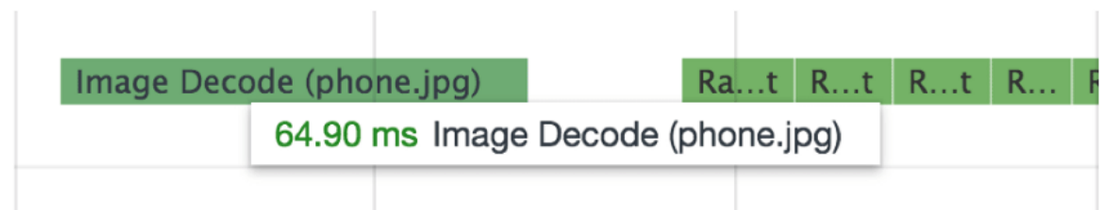
~170KB

~170KB

Network Transmission

Resource Processing

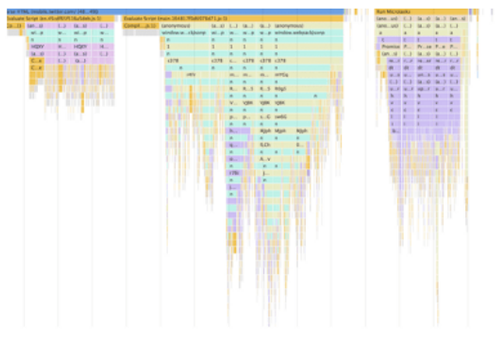




~2s in Parse/Compile

0.064s in Image Decode

~1.5s in Execution



0.028s in Rasterize Paint

@addyosmani - 170KB of (compressed) JS vs. JPEG bytes over a slow 3G network on a Moto G4. JS needing parsed is even larger once decompressed.

SPA's are very fast once loaded

What do SPA's mean for developers?

Unified language for building interfaces

- Traditional: server language + JavaScript
- SPA: All JavaScript

The component model

```html
<form class="signup-form">
  <input type="email" class="signup-form-input">
  <button type="submit" class="button button-submit">
    Submit
  </button>
</form>
```

```
.signup-form {
}

.signup-form-input {
}

.signup-form-thanks {
}

/* Somewhere else... */

.button {
}

.button-submit {
}
```

```
const formEl = document.querySelector('.signup-form');

formEl.addEventListener('submit', e => {
  // Submit with Ajax...

  const thanksEl = document.createElement('strong');
  thanksEl.classList.add('signup-form-thanks');
  thanksEl.innerText = 'Thanks!';

  formEl.appendChild(thanksEl);
});
```

```
project/
  app/
  resources/
    assets/
      css/
        signup-form.css
      js/
        signup-form.js
    views/
      signup.blade.php
```

Problems

- The template doesn't declare intent
- Styles are global
- Everything is scattered
- Everything is implicitly coupled
- Concerns aren't really separated

```
export default class SignupForm extends Component {
  state = { submitted: false }
  handleSubmit = e => { /* ... */ }

  render() {
    return <form onSubmit="handleSubmit">
      <input type="email" />
      <button
        type="submit"
        className="button button-submit"
        onClick={this.handleSubmit}
      >Submit</button>
      {this.state.submitted && (
        <strong>Thanks!</strong>
      )}
      <style jsx>{`
        form { /* ... */ }
        input { /* ... */ }
        strong { /* ... */ }
      `}</style>
    </form>;
  }
}
```

```
project/
  app/
  resources/
    assets/
      js/
        SignupForm.js
```

```
export default class SignupForm extends Component {
  state = { submitted: false }
  handleSubmit = e => { /* ... */ }

  render() {
    return <form onSubmit="handleSubmit">
      <input type="email" />
      <button
        type="submit"
        className="button button-submit"
        onClick={this.handleSubmit}
      >Submit</button>
      {this.state.submitted && (
        <strong>Thanks!</strong>
      )}
      <style jsx>{`
        form { /* ... */ }
        input { /* ... */ }
        strong { /* ... */ }
      `}</style>
    </form>;
  }
}
```

Things we fixed

- The component declares intent
- Styles are scoped
- Everything is coupled
- (okay because it's not scattered anymore)
- Separation of concerns...

😱😱😱

You're mixing logic and presentation!!!!

Do you even separation of concerns?????

```
export default class SignupForm extends Component {
  state = { submitted: false }
  handleSubmit = e => { /* ... */ }

  render() {
    return <form onSubmit="handleSubmit">
      <input type="email" />
      <button
        type="submit"
        className="button button-submit"
        onClick={this.handleSubmit}
      >Submit</button>
      {this.state.submitted && (
        <strong>Thanks!</strong>
      )}
      <style jsx>{`
        form { /* ... */ }
        input { /* ... */ }
        strong { /* ... */ }
      `}</style>
    </form>;
  }
}
```

Too many concerns

- Submit button implementation details
- Rendering a form
- Handling a form

```
export default class SignupForm extends Component {
  state = { submitted: false }
  handleSubmit = e => { /* ... */ }

  render() {
    return <form onSubmit="handleSubmit">
      <input type="email" />
      <button
        type="submit"
        className="button button-submit"
        onClick={this.handleSubmit}
      >Submit</button>
      {this.state.submitted && (
        <strong>Thanks!</strong>
      )}
      <style jsx>{`
        form { /* ... */ }
        input { /* ... */ }
        strong { /* ... */ }
      `}</style>
    </form>;
  }
}
```

```
class Button extends Component {
  render() {
    return (
      <button
        type="submit"
        onClick={this.props.onClick}
      >
        {this.props.children}
        <style jsx>{`
            button { /* ... */ }
        `}</style>
      </button>
    );
  }
}

// Usage:

<SubmitButton onClick={this.handleSubmit}>
  Submit
</SubmitButton>
```

```
export default class SignupForm extends Component {
  render() {
    return <AjaxForm>
      {({ submitted, handleSubmit }) => (
        <input type="email" />
        <SubmitButton onClick={handleSubmit}>
          Submit
        </SubmitButton>
        {submitted && (
          <strong>Thanks!</strong>
        )}
        <style jsx>{`
          form { /* ... */ }
          input { /* ... */ }
          strong { /* ... */ }
        `}</style>
      )};
    </AjaxForm>;
  }
}
```

```
export default class AjaxForm extends Component {
  state = { submitted: false }
  handleSubmit = e => { /* ... */ }

  render() {
    return <form onSubmit={handleSubmit}>
      {this.props.children({
        submitted: this.state.submitted,
        handleSubmit: this.handleSubmit,
      })}
    </form>;
  }
}
```

Unidirectional data flow

```html
<button type="submit">
  Submit
</button>
```

```javascript
function startSubmit() {
  button.disabled = true;
  button.style.opacity = 0.5;
}

function finishSubmit() {
  button.disabled = false;
  button.style.opacity = 1;
}
```

```
function SubmitButton(submitting, handleSubmit) {
  return (
    <button
      type="submit"
      disabled={submitting}
      style={{ opacity: submitting ? 0.5 : 1 }}
      onClick={handleSubmit}
    >
      Submit
    </button>
  );
}
```

Components

- Better separation of concerns
- Better developer experience by keeping code together
- Less bugs with unidirectional data flow
- Self-documentation & communication

Real-world example

Gestalt: Pinterest's UI library

Back to SPA's

- Unified programming language
- Better performance once loaded
- Components are awesome

SPA's break browsers

Things we lose

- Browser history
- Scroll position on back
- Scripts like analytics
- Loading indicators

When there's a runtime error, the whole page breaks

Infrastructure

Server side rendering

Without SSR

- Download scripts
- Parse scripts
- Run scripts
- Retrieve data
- Render the app *(first meaningful paint)*
- Ready! *(interactive)*

With SSR

- Render incoming HTML from the server *(first meaningful paint)*
- Download scripts
- Parse scripts
- Run scripts
- Retrieve data
- Make the existing HTML interactive
- Ready! *(interactive)*

What are you building?

Is interactivity a first-class citizen?

Server side app & JS sprinkles

- All about content
- It's simple
- Document is immediately available
- Complex interfaces are prone to become a ball of mud

API + Client side SPA

- API-first development
- Needs to work offline
- When backend doesn't provide much value
- Scalability
- SSR is hard

Server side app & JS framework

- Balanced complexity
- Not the same DX as a SPA
- Not the same UX as a well-built SPA

Eric Clemmons Follow
I write about tech (JavaScript, node, GraphQL, React, webpack) and Leadership. Currently building GraphQL apps @Starbucks.
Dec 27, 2015 · 4 min read

Javascript Fatigue

A few days ago, I met up with a friend & peer over coffee.

> *Saul: "How's it going?"*
> *Me: "Fatigued."*
> *Saul: "Family?"*
> *Me: "No, Javascript."*

Getting started with SPA's

- vuejs/vue-cli
- facebook/create-react-app

Thanks!

@sebdedeyne — *Twitter*
sebastiandedeyne.com — *Blog*
growingthestack.io — *Newsletter*
spatie.be — *We're hiring!*