

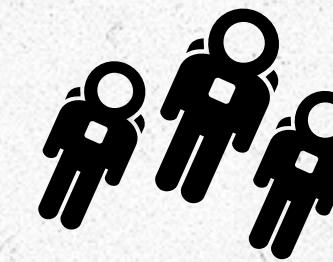
# From MVC to React

[@ericclemmons](#)

# History

How this all started

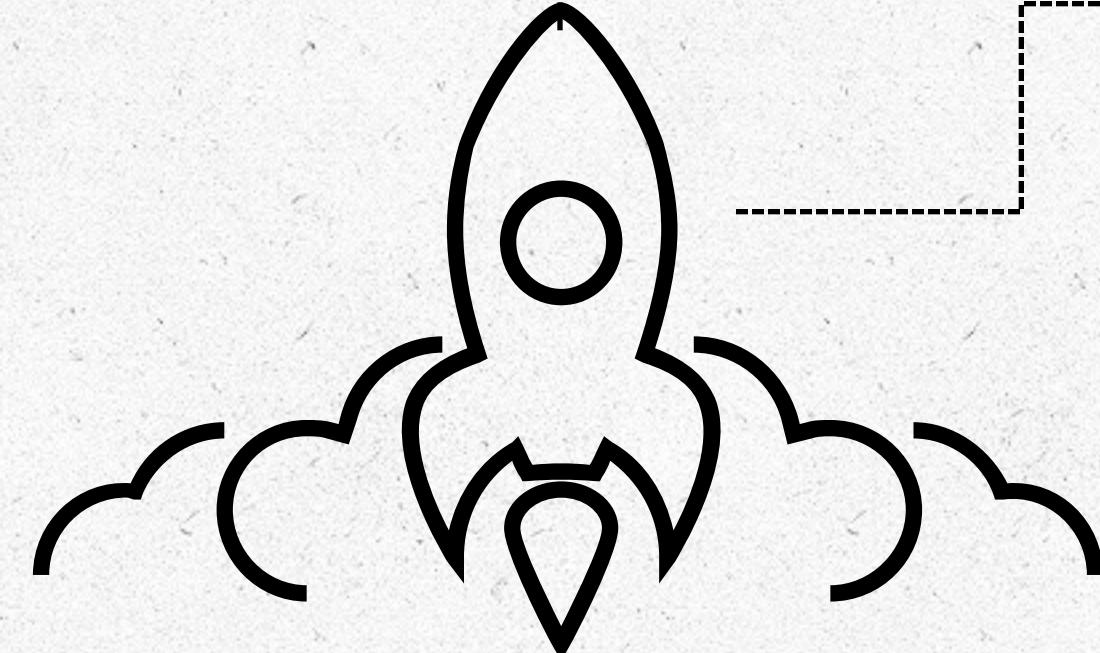
**2011**  
Grew team & developed prototype.



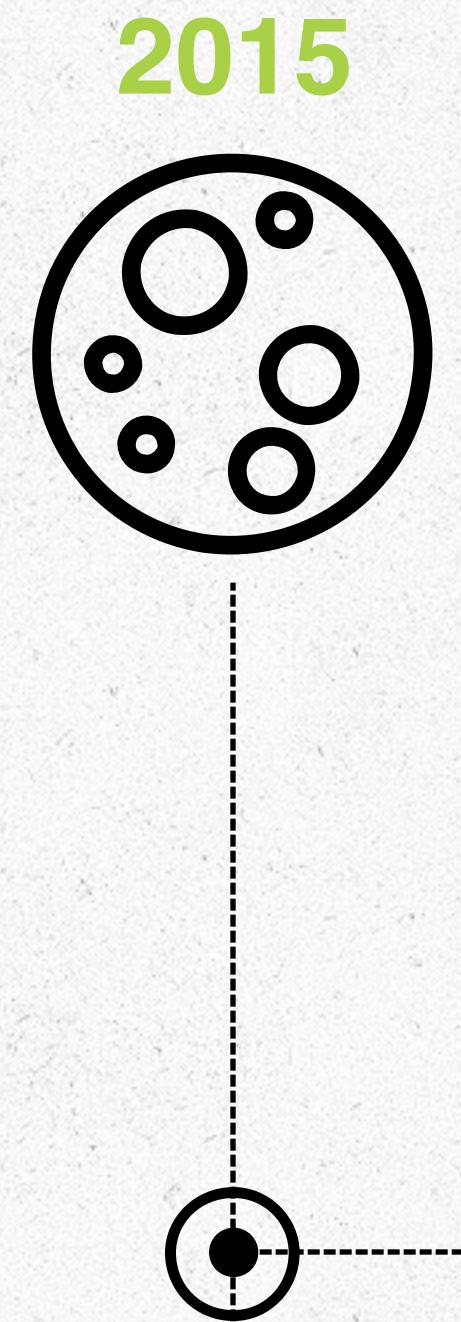
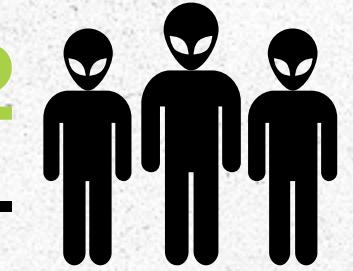
**2011**  
Prototype reached peak success.



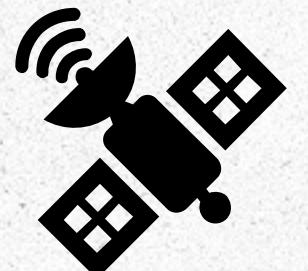
**2010**  
Joined CMN.com



**2012**  
Threw away the prototype.



**Explosive** growth & optimization.



# A/B/x Testing



**Responsive UI**  
Mobile vs. Desktop assets



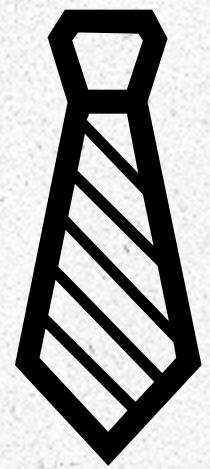
**Performance**  
Biggest impact on revenue.



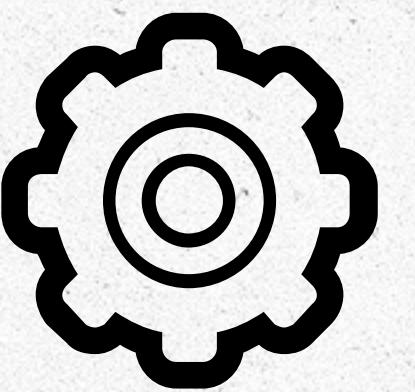
# Dismantling the Monolith

From PHP to Node

Admin



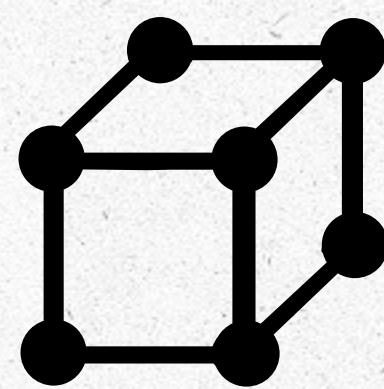
API



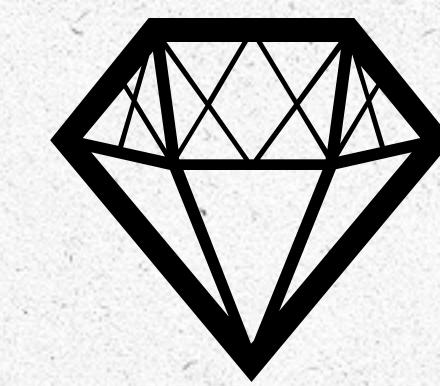
Business Logic



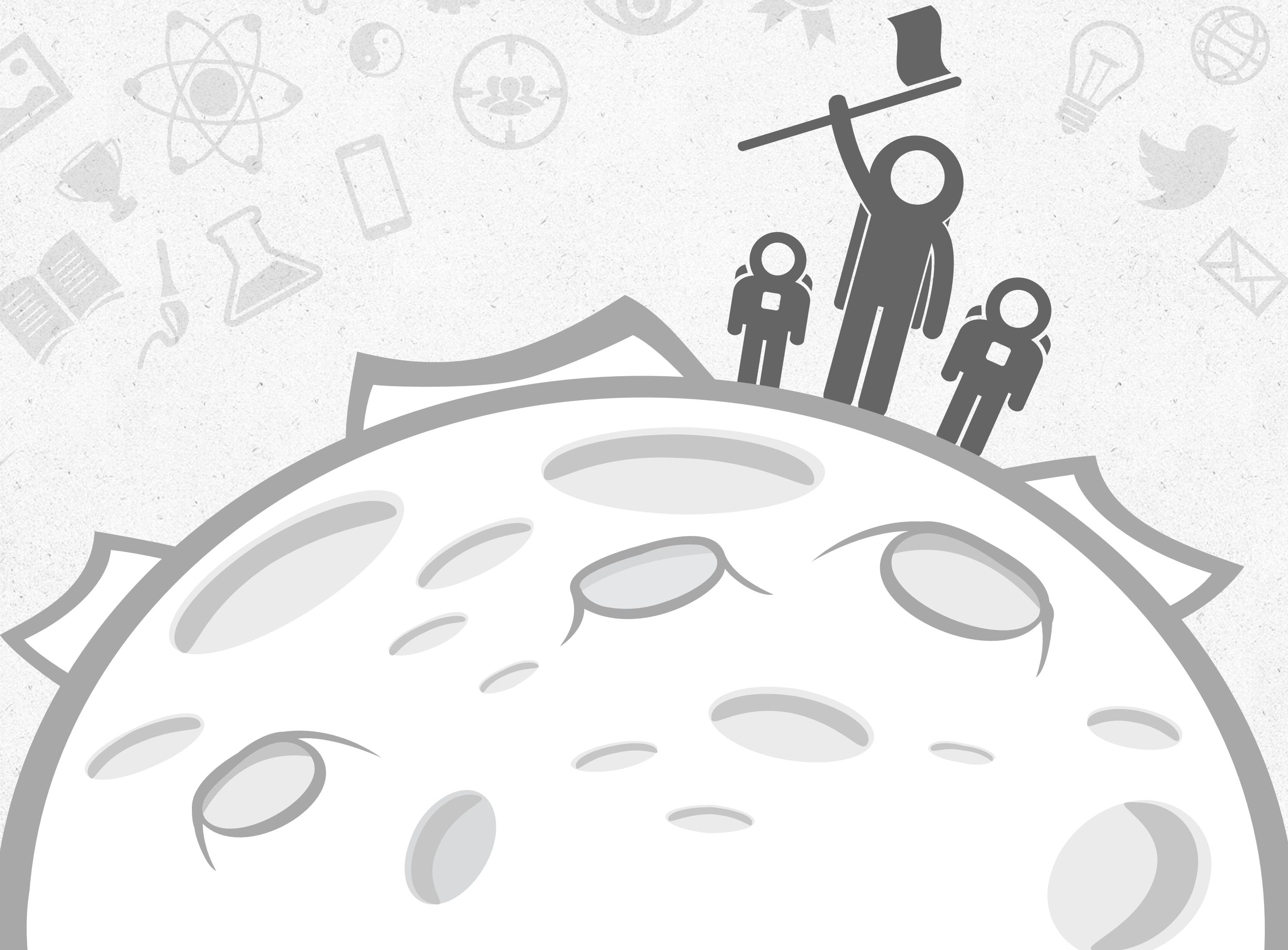
UI



Profit?

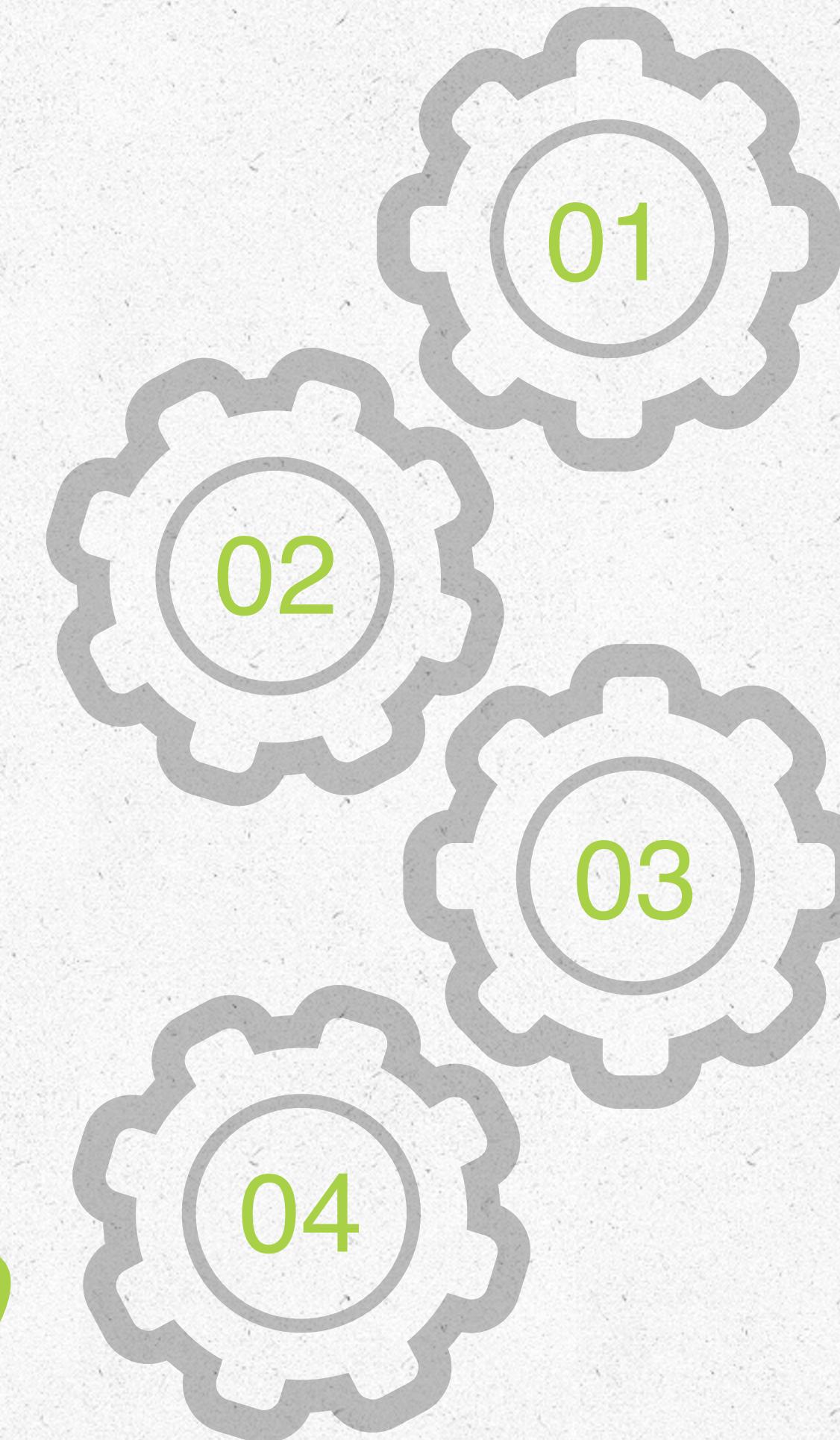


# Javascript is awesome!



# Requirements

**Simple**  
Conceptually easier to understand how logic works.



**Fast**  
Server responses, rendering, validation.



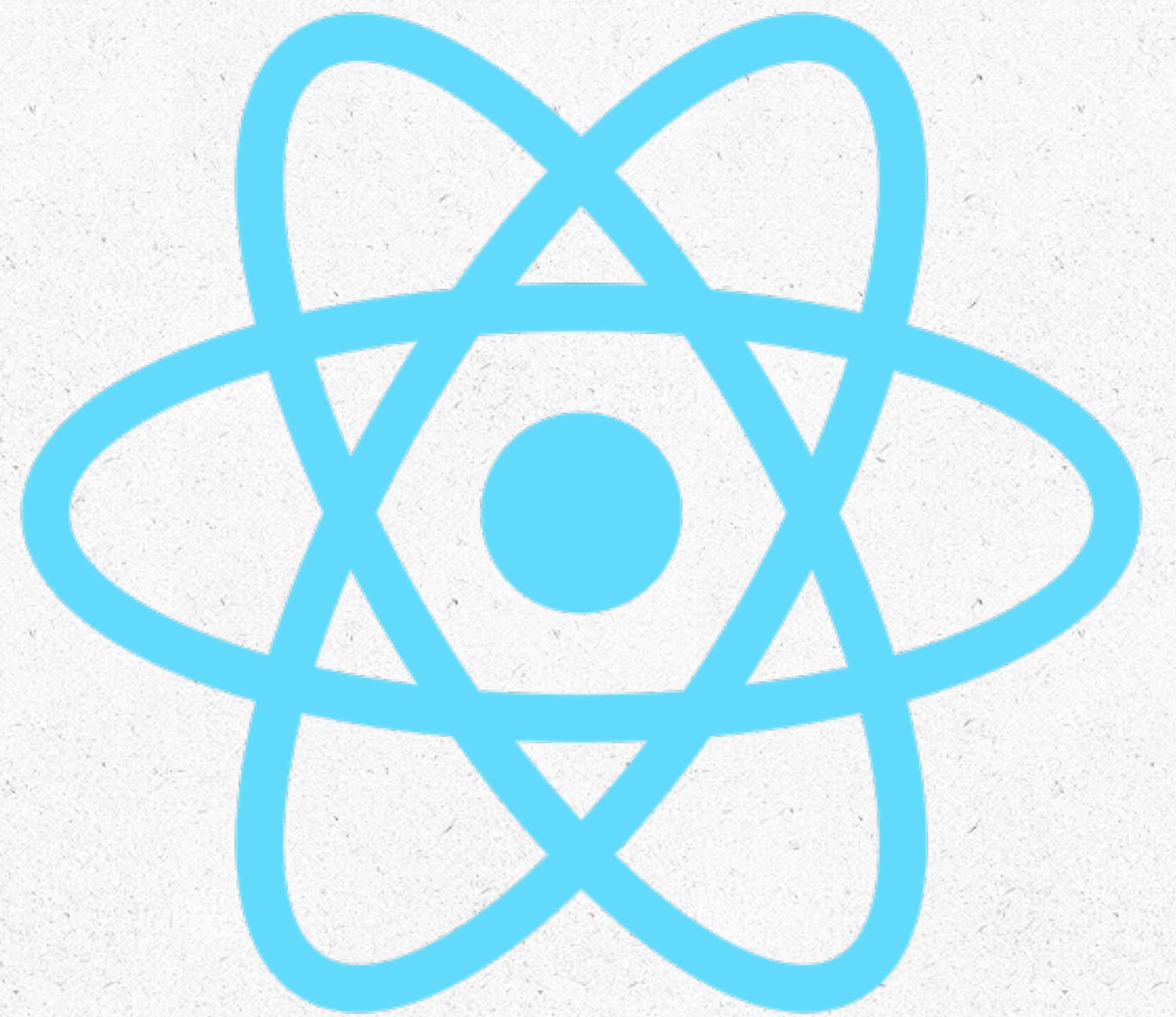
**Isomorphic**  
Rendering on the server is just *smart*.

**Modular**  
Technology moves fast enough to justify replacing parts within a year.



# The Solution

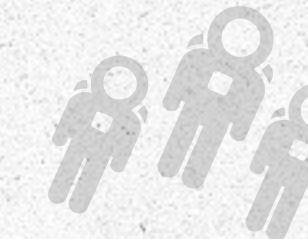
As of 2015



**React**  
Facebook

# Benefits

That match our requirements



## Community

Active F/OSS developers supporting their tools and React in mediums such as Slack pushing Javascript forward.



## Fast

Utilizes a *Virtual DOM* to only render what has changed.



## Adoption

Companies with much larger problem sets are leverage React.



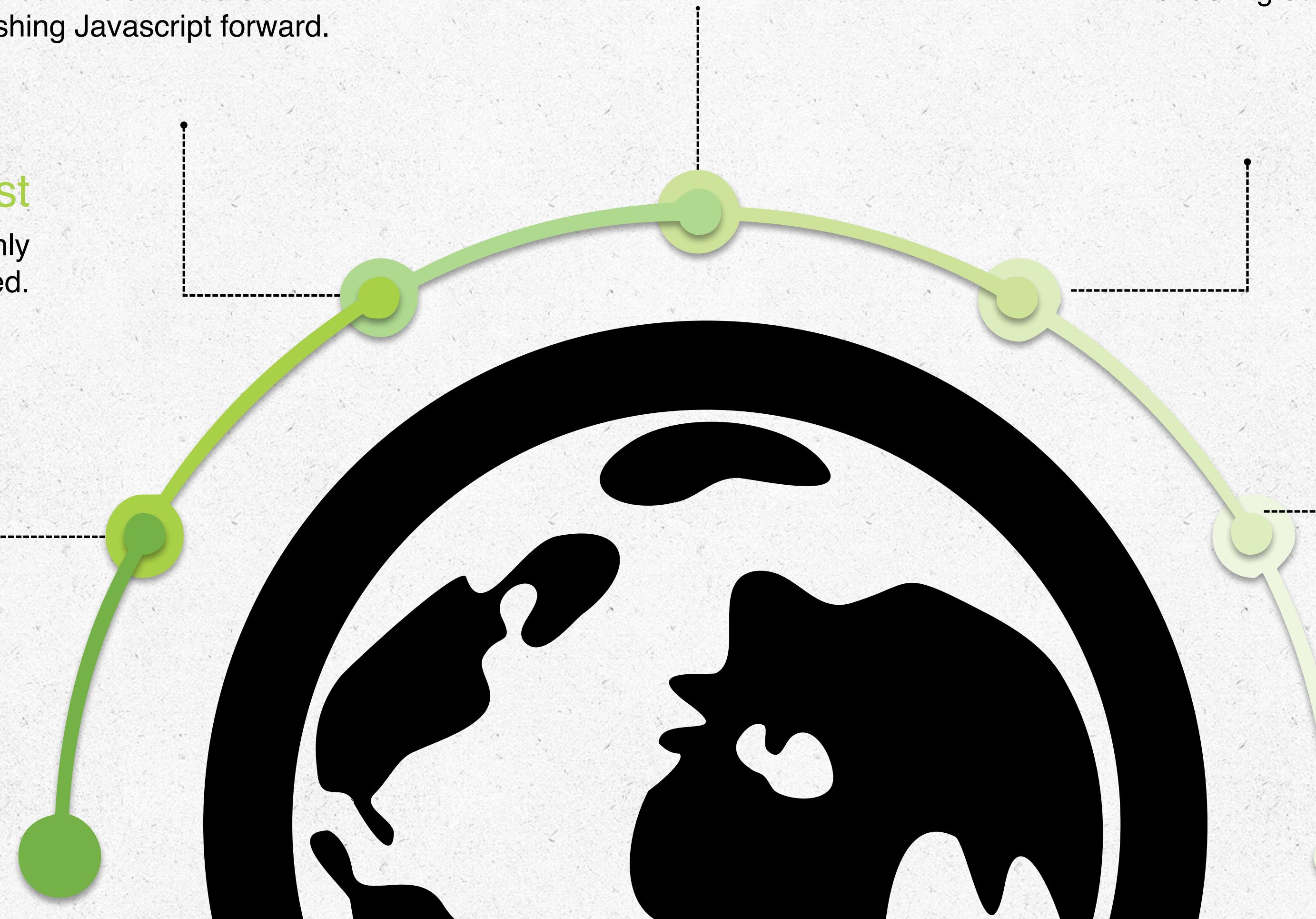
## Isomorphic

Server-side & client-side rendering out of the box.



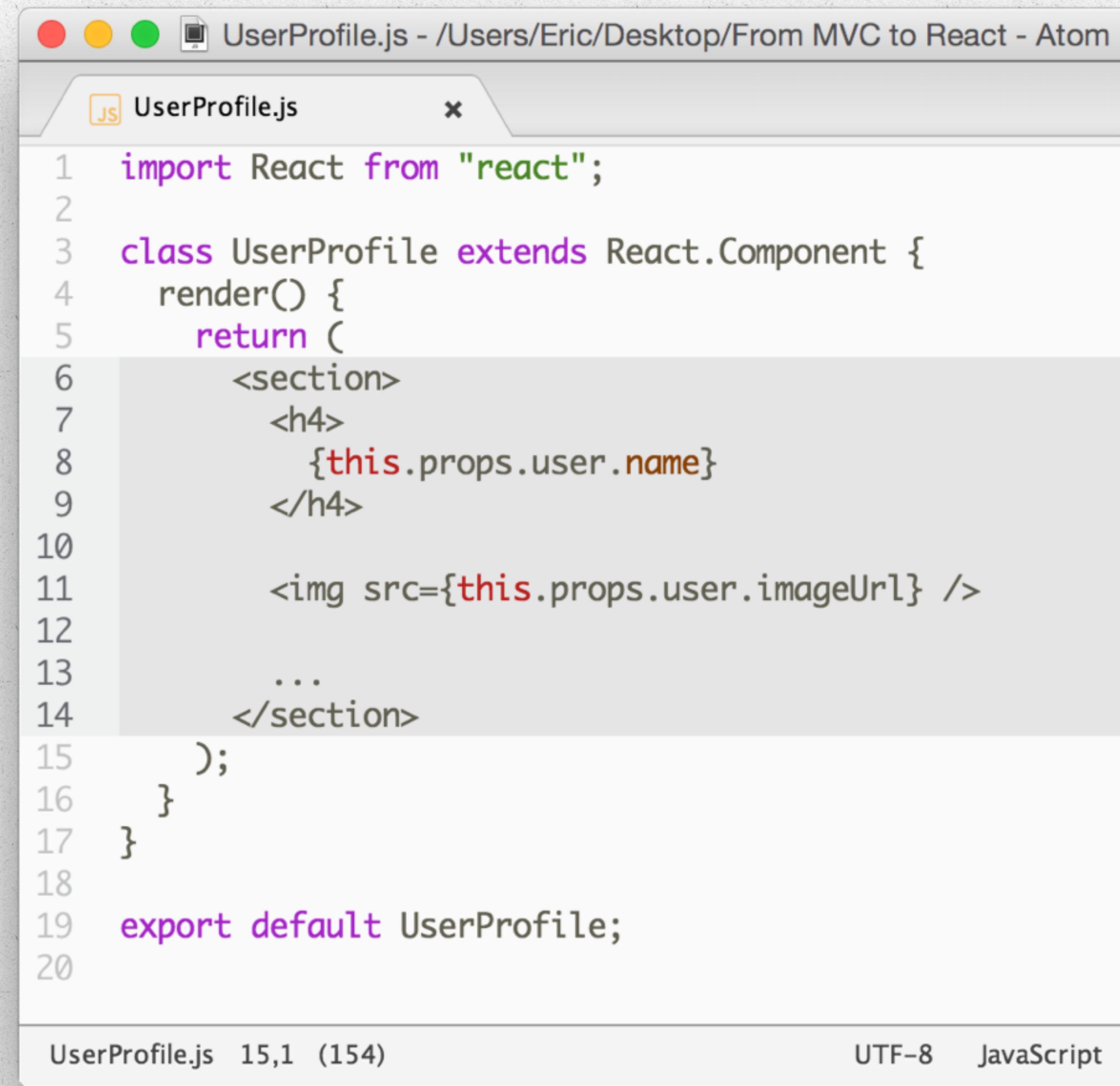
## Simple

Just the View layer, the rest of the architecture is up to you.



# What does it look like?

Simple Example



The image shows a screenshot of the Atom code editor. The title bar reads "UserProfile.js - /Users/Eric/Desktop/From MVC to React - Atom". The file tab also says "UserProfile.js". The code editor displays the following JavaScript code:

```
1 import React from "react";
2
3 class UserProfile extends React.Component {
4   render() {
5     return (
6       <section>
7         <h4>
8           {this.props.user.name}
9         </h4>
10
11        <img src={this.props.user.imageUrl} />
12
13        ...
14      </section>
15    );
16  }
17}
18
19 export default UserProfile;
20
```

The code uses ES6 syntax, including imports, classes, and arrow functions. It defines a component named `UserProfile` that renders a user's name and profile picture. The code editor interface includes status bars at the bottom showing "UserProfile.js 15,1 (154)", "UTF-8", and "JavaScript".

server.js - /Users/Eric/Desktop/From MVC to React - Atom

UserProfile.js client.js server.js

```
1 import React from "react";
2
3 import UserProfile from "./components/UserProfile";
4
5 const markup = React.renderToString(<UserProfile user={user} />);
6
7 reply(markup);
8
```

server.js 5,66 (65)

client.js - /Users/Eric/Desktop/From MVC to React - Atom

UserProfile.js client.js

```
1 import React from "react";
2
3 import UserProfile from "./components/UserProfile";
4
5 React.render(
6   <UserProfile user={user} />,
7   document.getElementById("app")
8 );
9
```

client.js 6,22 (4) UTF-8 JavaScript



## Components

Allows the author to describe *what* the UI consists of (e.g. TableGrid, UserProfile, CodeEditor, etc.) rather than *how*.



# Properties

Usually shown as HTML-like attributes,  
these are external values explicitly  
passed down to components by their  
parent.



**State**  
Internal properties that changes how  
a component is rendered  
(e.g. "on" vs. "off").



# Context

**Advanced**, scarcely documented feature that allows deeply-nested components to opt-in to a shared state for data-sharing and communication.



# From MVC...

What we know...

## Model

Domain logic

View data

Constraints

Validation

?

## Controller

Handles request

Fetches data

Performs logic

Renders view

Sends response

?

## View

Fast

Simple

Isomorphic

One-way data flow

React

# How does our current app work?

High-level overview

## Cart Page

User wants to adjust what's in their shopping cart.



## Request

A request is made to our server.





## Front Dispatcher

Entry-point receives the request.



## Router

The router identifies which route(s) can respond to the request.



## Controller

The controller fetches the data it designates useful for the view.



## Progressive Enhancement

Javascript updates the view as the user interacts to save round-trips to the server.



## View

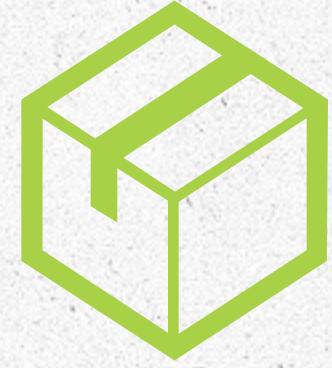
The view uses the normalized data from the Controller to present it in a useful manner for the user.

## Interaction

The user interacts with the page (e.g. a form).

## HTTP POST

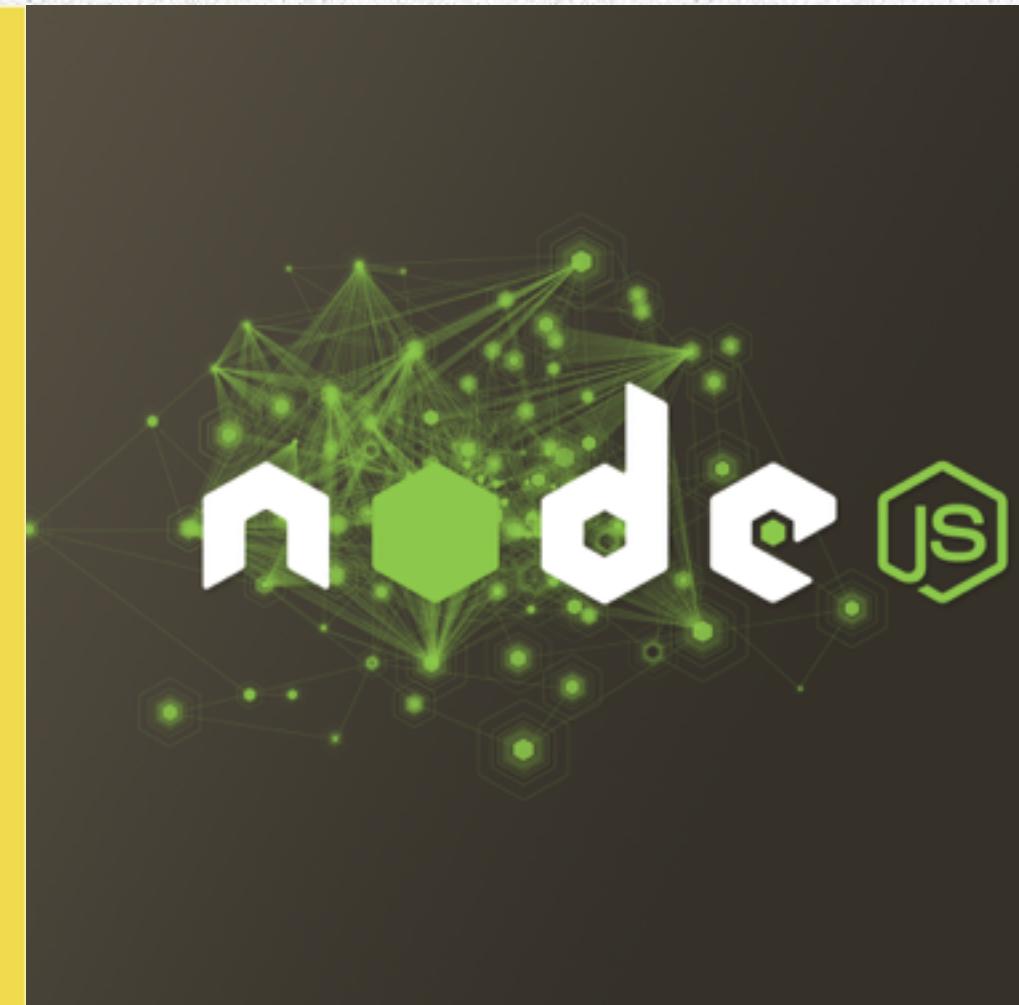
Data is sent to controller, validated, and prepared for storage.



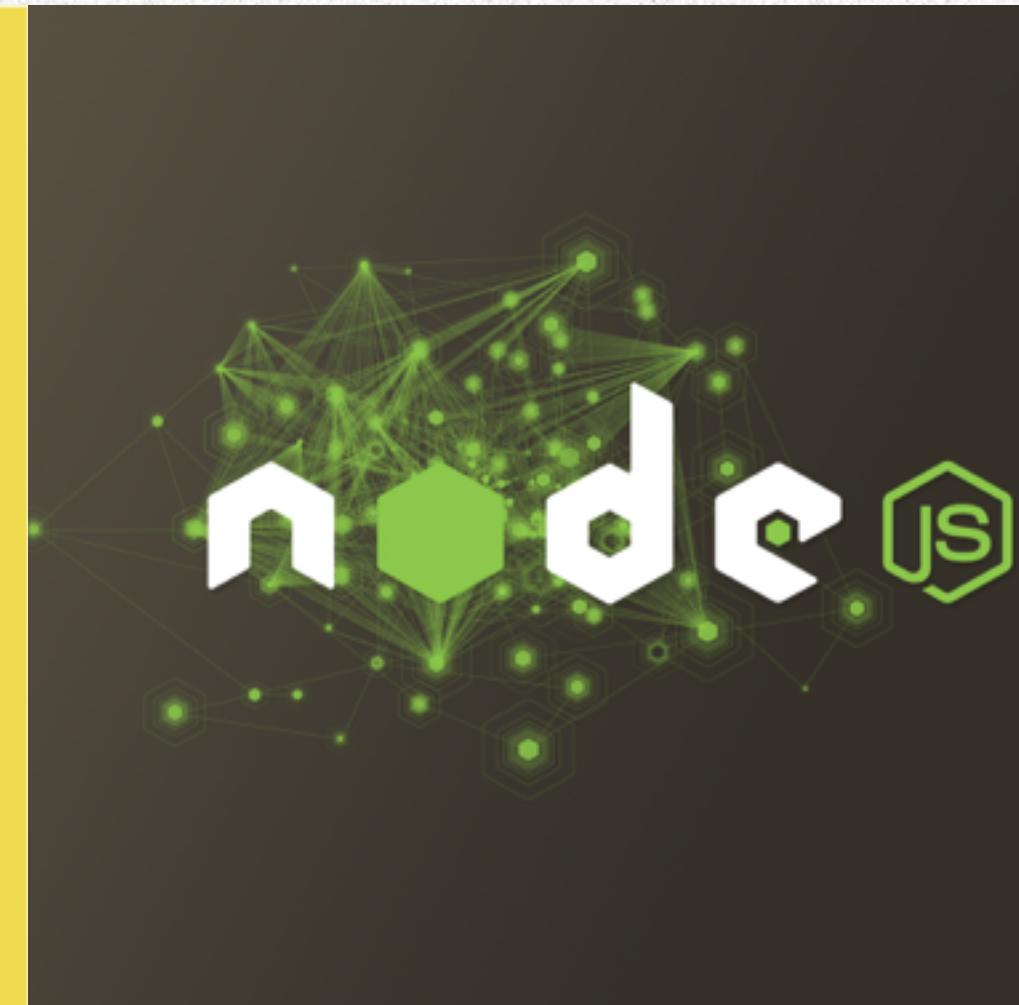
## Submit

User finalizes edits & submits the form.

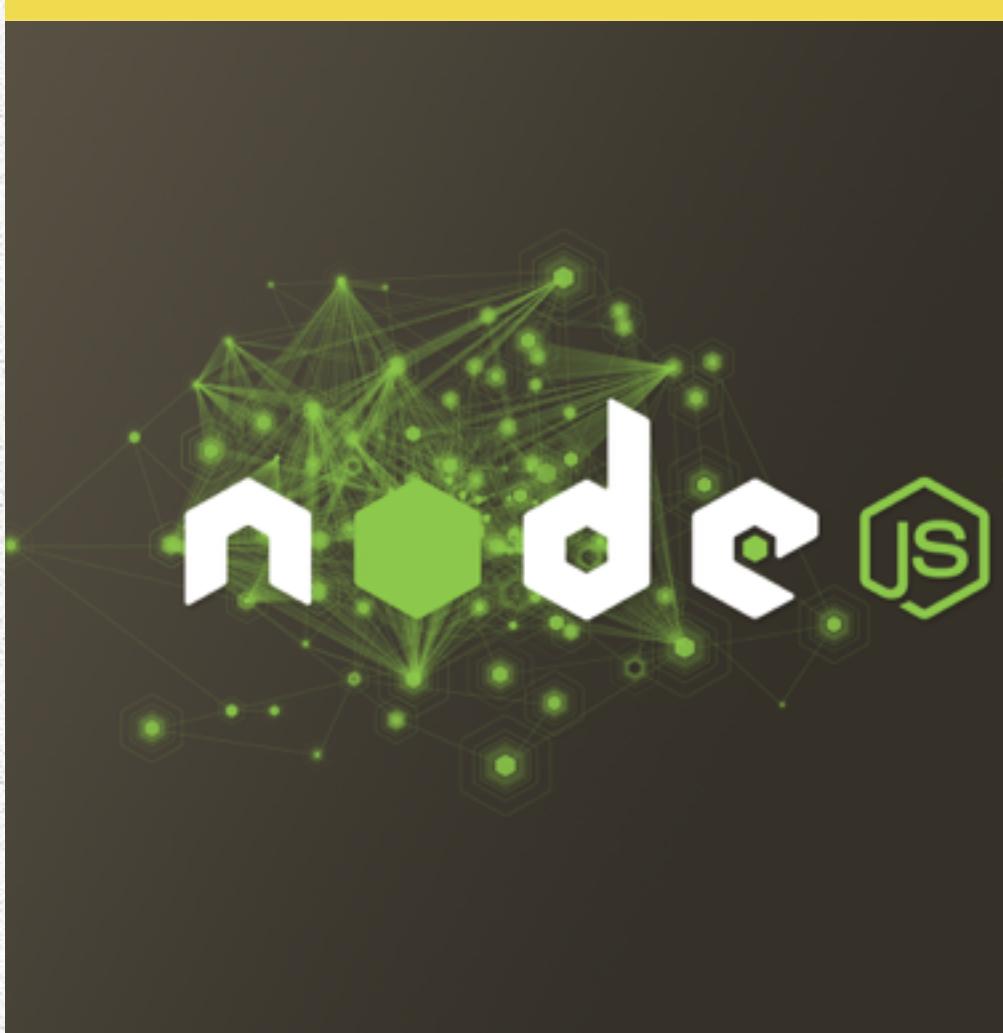
**JS**



**JS**



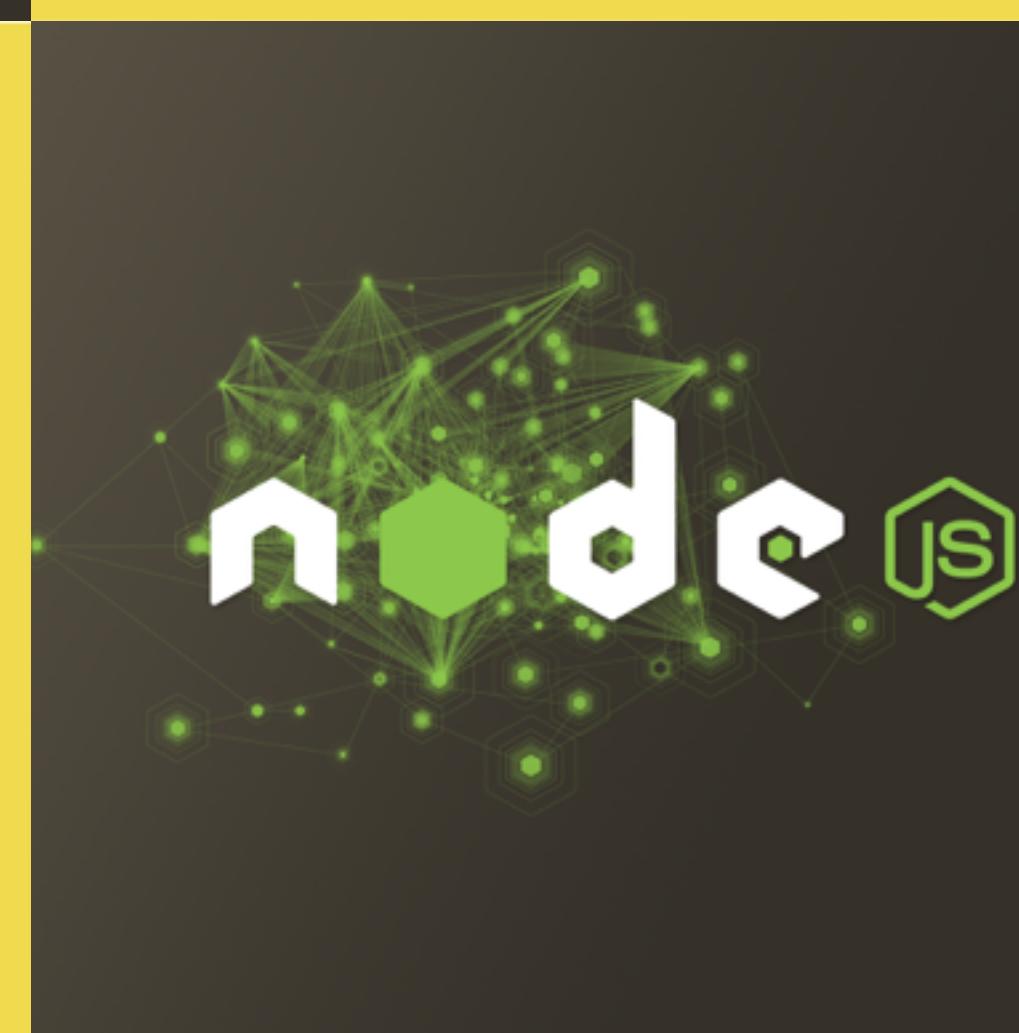
**JS**



**JS**



**JS**



**Now with Javascript.**

©Ria

# How can our Javascript app work?

High-level overview

**Cart Page**  
User wants to adjust what's in their shopping cart.



Request  
Node

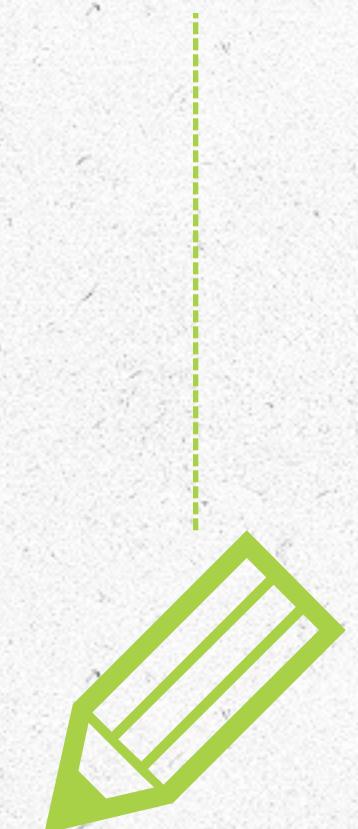




**Front Dispatcher**  
Hapi, Express, Koa, etc.



**Router**  
Server Router + React Router



**Handler**  
Component that handles primarily route layout.



## Progressive Enhancement

React re-renders the view accordingly.



## Components

Deeply-nested components that render the view.



## Interaction

User triggers actions that update props & state.

## AJAX POST

Data is validated and sent to an API endpoint for storage.

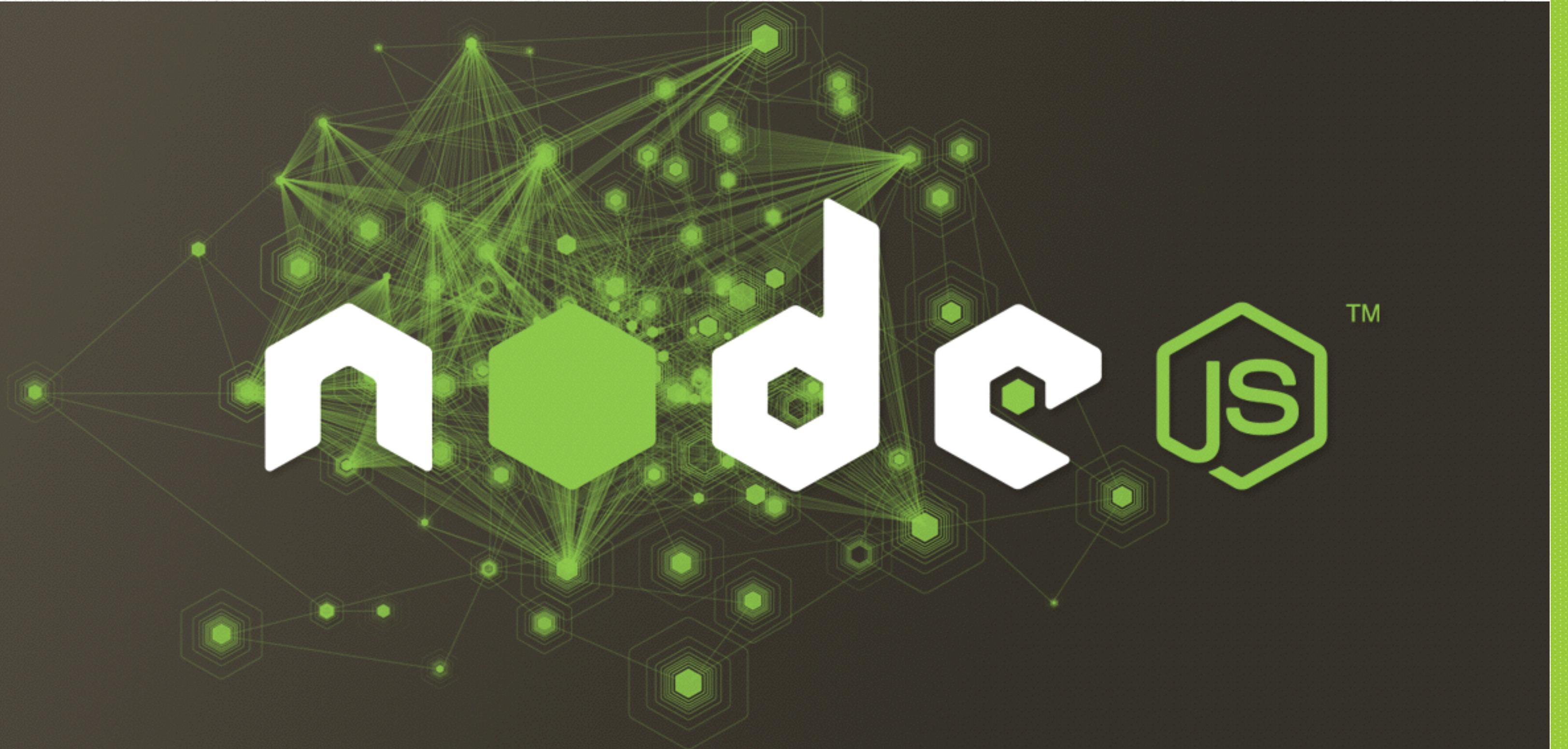


## Submit

User finalizes edits & submits the form.

# Example Project

<https://github.com/ericclemmons/mvc-to-react>



Let's See  
Some Code!

# Server

Hapi + React + ES6



A screenshot of the Atom code editor showing a file named 'server.js'. The title bar indicates the file path is '/Users/Eric/Projects/ericclemons/mvc-to-react' and the editor is Atom. The code itself is written in JavaScript and uses Hapi and React libraries. It defines a new Hapi Server, sets up a connection on port 3000, routes GET requests to a specific path, and handles them by rendering a React component ('Howdy!') to a string, then creating an HTML layout (DOCTYPE, html, head with meta charset=UTF-8, body with a div id=app containing the view) and replying with it.

```
1 import Hapi from "hapi";
2 import React from "react";
3
4 const server = new Hapi.Server();
5
6 server.connection({ port: 3000 });
7
8 server.route({
9   method: "GET",
10  path: "/{path*}",
11  handler: function(request, reply) {
12    const view = React.renderToString(<h1>Howdy!</h1>);
13
14    const layout = `
15      <!DOCTYPE html>
16      <html lang="en">
17        <head>
18          <meta charset="UTF-8">
19        </head>
20        <body>
21          <div id="app">${view}</div>
22        </body>
23      </html>
24    `;
25
26    reply(layout);
27  }
28 });
29
30 server.start();
```

# Client & Server

Hapi + React + Browserify + ES6

```
client.js - /Users/Eric/Projects/ericclemmons/mvc-to-react - Atom
client.js      x  server.js      x
1 import React from "react";
2
3 React.render(<h1>Howdy there!</h1>, document.getElementById("app"));
4
```

client.js 1,1      UTF-8    JavaScript (JSX)    master ↓ 8 2

```
server.js - /Users/Eric/Projects/ericclemmons/mvc-to-react - Atom
client.js      x  server.js      x
8 server.route({
9   method: "GET",
10  path: "/client.min.js",
11  handler: function(request, reply) {
12    reply.file(`$__dirname__/client.min.js`);
13  }
14 });
15
16 server.route({
17   method: "GET",
18   path: "/{path*}",
19   handler: function(request, reply) {
20     const view = React.renderToString(<h1>Howdy!</h1>);
21
22     const layout =
23       `<!DOCTYPE html>
24       <html lang="en">
25         <head>
26           <meta charset="UTF-8">
27         </head>
28         <body>
29           <div id="app">${view}</div>
30           <script src="client.min.js" async defer></script>
31         </body>
32       </html>
33     `;
34
35     reply(layout);
36   }
37 });
```

# Props

Client & Server

server.js - /Users/Eric/Projects/ericclemonns/mvc-to-react - Atom

```
1 import Hapi from "hapi";
2 import React from "react";
3
4 import Home from "./components/Home";
5
6 const server = new Hapi.Server();
7
8 server.connection({ port: 3000 });
9
10 server.route({
11   method: "GET",
12   path: "/{path*}",
13   handler: function(request, reply) {
14     const view = React.renderToString(<Home path={request.path} />);
15
16     const layout = `
17       <!DOCTYPE html>
18       <html lang="en">
19         <head>
20           <meta charset="UTF-8">
21         </head>
22         <body>
23           ${view}
24         </body>
25       </html>
26     `;
27
28     reply(layout);
29   }
30 });
31
32 server.start();
33
34 console.log("Server running on port 3000");
35
```

server.js 23,1 (69)      UTF-8    JavaScript (JSX)    master ↓ 6    2

Home.js - /Users/Eric/Projects/ericclemonns/mvc-to-react - Atom

```
1 import React from "react";
2
3 class Home extends React.Component {
4   render() {
5     return (
6       <h1>Howdy from <code>{this.props.path}</code>!</h1>
7     );
8   }
9 }
10
11 export default Home;
```

components/Home.js 7,1 (58)      UTF-8    JavaScript (JSX)    master ↓ 6    2

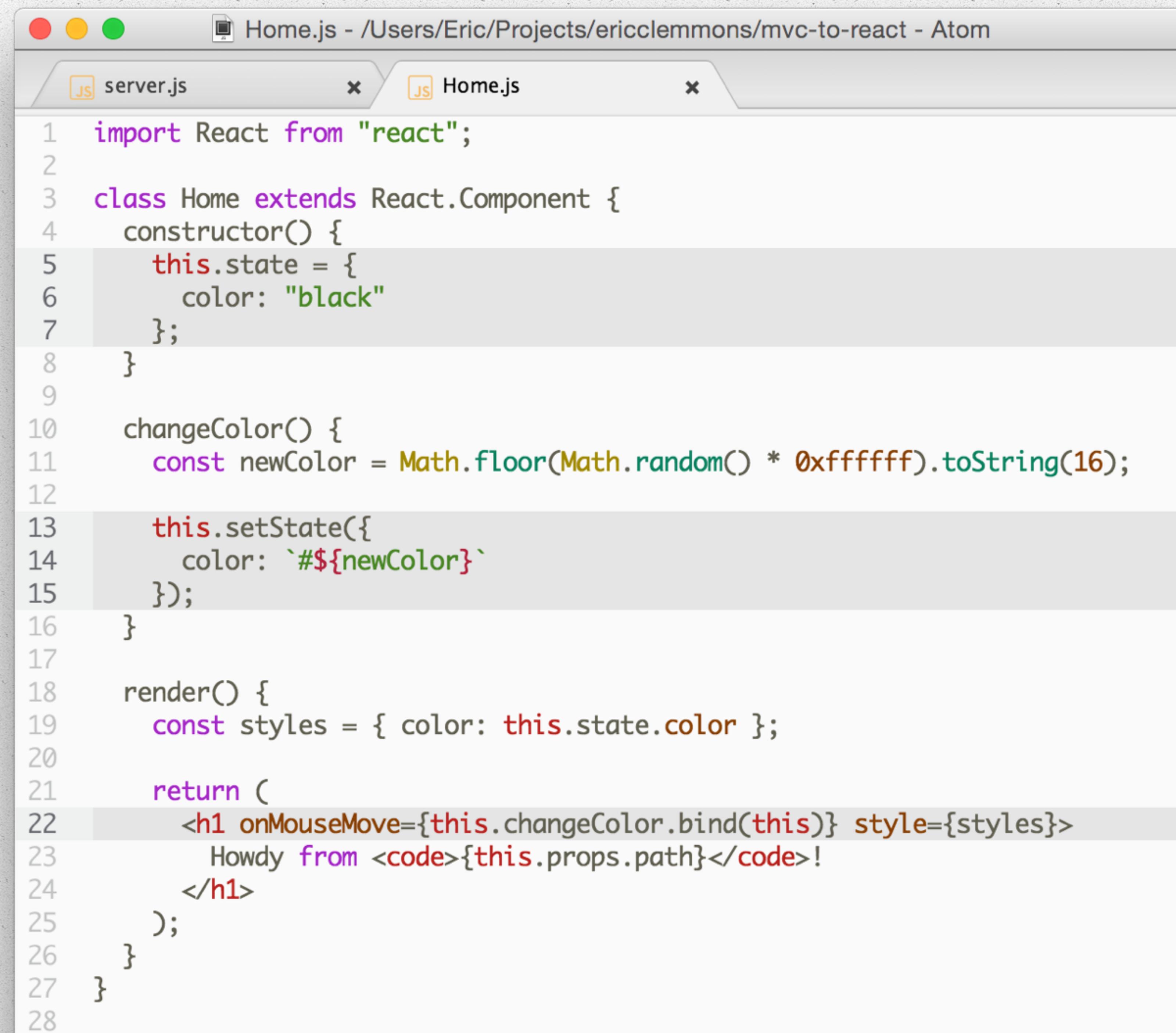
client.js - /Users/Eric/Projects/ericclemonns/mvc-to-react - Atom

```
1 import React from "react";
2
3 import Home from "./components/Home";
4
5 React.render(
6   <Home path={window.location.pathname} />,
7   document.getElementById("app")
8 );
9
```

client.js\* 4,1 (38)      UTF-8    JavaScript (JSX)    master ↓ 6    2

# State

Home



The screenshot shows a Mac OS X desktop environment with the Atom code editor open. The window title is "Home.js - /Users/Eric/Projects/ericclemmons/mvc-to-react - Atom". The editor has two tabs: "server.js" and "Home.js", with "Home.js" being the active tab. The code in "Home.js" is a React component named "Home". It imports React, defines a constructor that initializes state with a color of "black", and includes a "changeColor" method that generates a new random color and updates the state. The "render" method creates an 

# element with a mouse move event handler that calls "changeColor" and a style object that uses the current state color. The component also receives a "path" prop from its parent.

```
1 import React from "react";
2
3 class Home extends React.Component {
4     constructor() {
5         this.state = {
6             color: "black"
7         };
8     }
9
10    changeColor() {
11        const newColor = Math.floor(Math.random() * 0xffff).toString(16);
12
13        this.setState({
14            color: `#${newColor}`
15        });
16    }
17
18    render() {
19        const styles = { color: this.state.color };
20
21        return (
22            <h1 onMouseMove={this.changeColor.bind(this)} style={styles}>
23                Howdy from <code>{this.props.path}</code>!
24            </h1>
25        );
26    }
27}
28
```

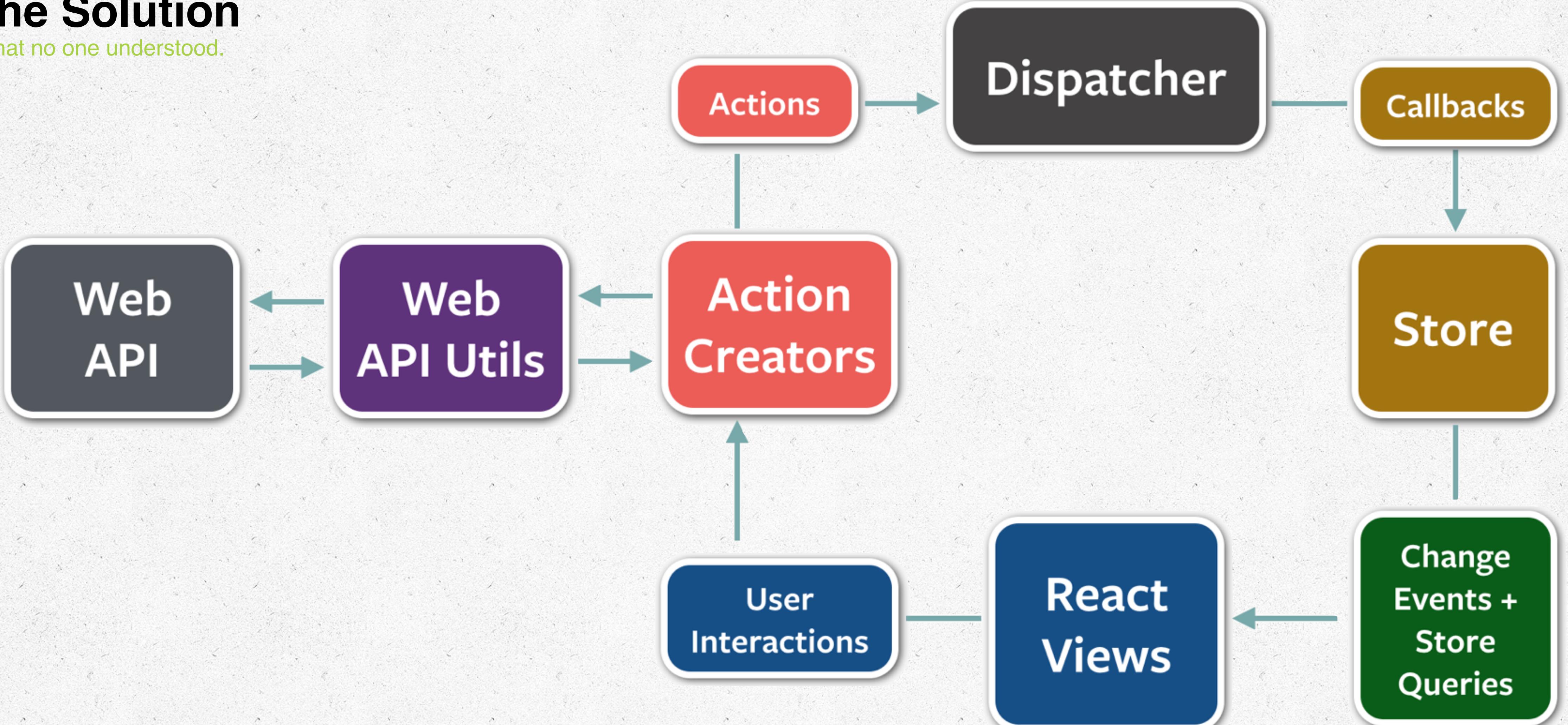


# Events

Untangling the complexity that  
is Progressive Enhancement.

# The Solution

...that no one understood.

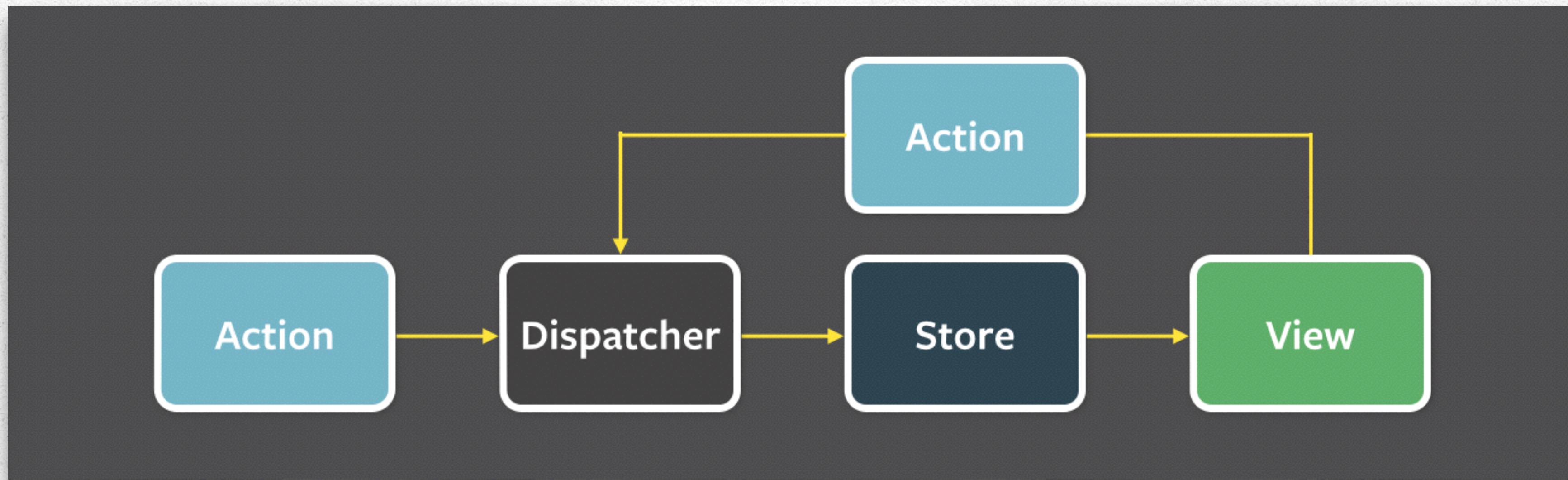


**Flux**

Facebook

# The Solution

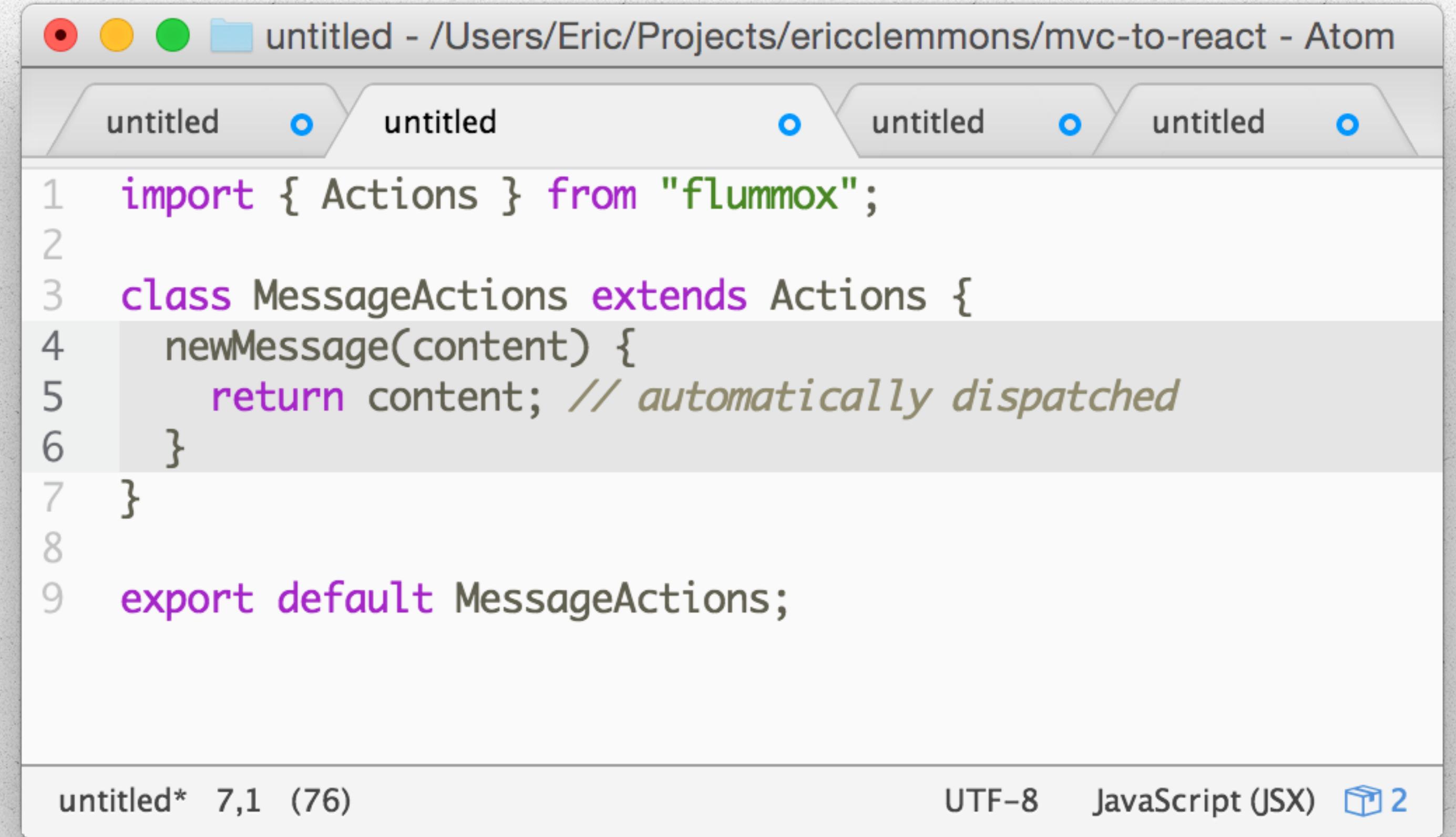
...that makes sense.



**Flux**  
Facebook

# Actions

Flummox



The screenshot shows a window titled "untitled - /Users/Eric/Projects/ericclemonns/mvc-to-react - Atom". The window contains a code editor with the following content:

```
1 import { Actions } from "flummox";
2
3 class MessageActions extends Actions {
4     newMessage(content) {
5         return content; // automatically dispatched
6     }
7 }
8
9 export default MessageActions;
```

The code uses ES6 syntax, including imports, classes, and arrow functions. It defines a class `MessageActions` that extends the `Actions` class from the `flummox` library. The class has a single method `newMessage` which returns its argument. A comment indicates that this return value is "automatically dispatched". Finally, the class is exported as the default. The file is saved as "untitled" and has a size of 7,1 (76).

# Store

Flummox

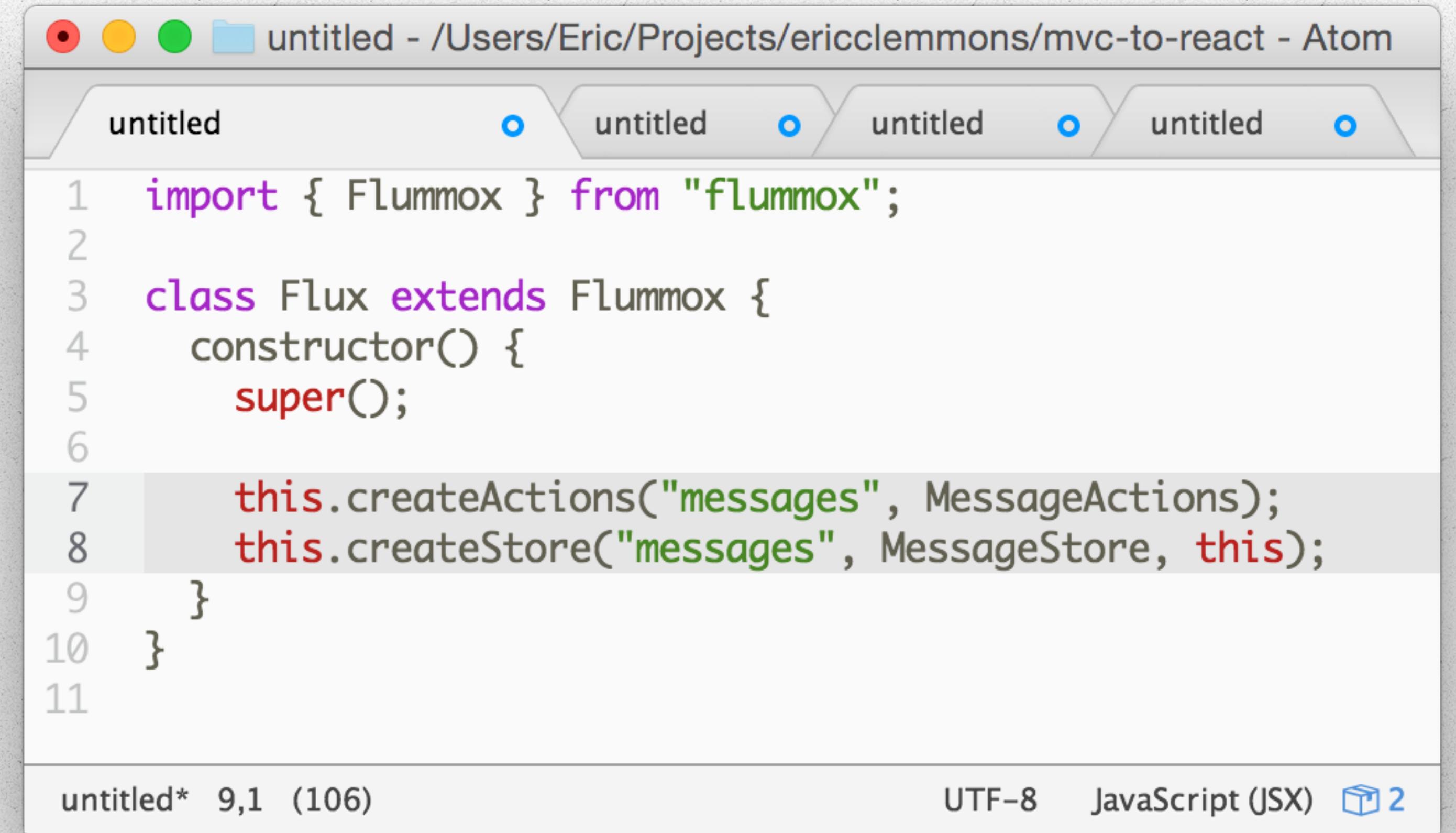
The screenshot shows a window titled "untitled - /Users/Eric/Projects/ericclemonns/mvc-to-react - Atom". The code editor displays a file named "untitled" containing the following JavaScript code:

```
1 import { Store } from "flummox";
2
3 class MessageStore extends Store {
4   constructor(flux) {
5     super();
6
7     const actions = flux.getActions("messages");
8
9     this.register(actions.newMessage, this.handleNewMessage);
10
11    this.messageCounter = 0;
12    this.state = {};
13  }
14
15  handleNewMessage(content) {
16    const id = this.messageCounter++;
17
18    this.setState({
19      [id]: {
20        content,
21        id,
22      },
23    });
24  }
25}
26
```

The code defines a `MessageStore` class that extends `Store` from `flummox`. It has a constructor that initializes `messageCounter` to 0 and `state` to an empty object. The `handleNewMessage` method increments `messageCounter`, creates a new state entry with the current `content` and `id`, and then sets the new state.

# App Flux

Flummox



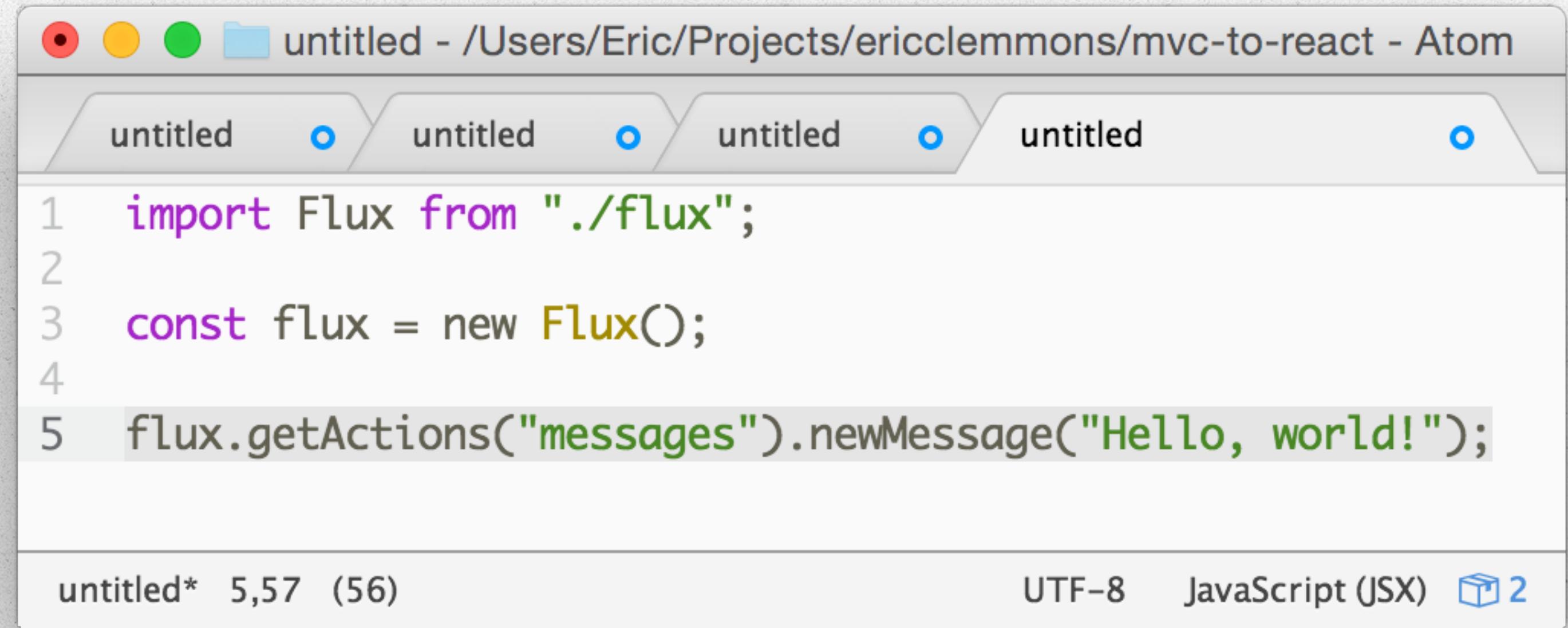
A screenshot of the Atom code editor interface. The title bar shows "untitled - /Users/Eric/Projects/ericclemonns/mvc-to-react - Atom". The tab bar has five tabs, all titled "untitled". The main editor area contains the following JavaScript code:

```
1 import { Flummox } from "flummox";
2
3 class Flux extends Flummox {
4   constructor() {
5     super();
6
7     this.createActions("messages", MessageActions);
8     this.createStore("messages", MessageStore, this);
9   }
10 }
11
```

The code uses syntax highlighting where "Flummox" and "Flux" are purple, "this", "super", and "createActions" are red, and "createStore" is green. Lines 7 and 8 are highlighted with a light gray background. The status bar at the bottom shows "untitled\* 9,1 (106)", "UTF-8", "JavaScript ( JSX )", and a file icon with the number "2".

# Usage

Flummox



A screenshot of the Atom code editor interface. The title bar shows "untitled - /Users/Eric/Projects/ericclemonns/mvc-to-react - Atom". Below the title bar, there are four tabs labeled "untitled" with blue circular icons. The main editor area contains the following code:

```
1 import Flux from "./flux";
2
3 const flux = new Flux();
4
5 flux.getActions("messages").newMessage("Hello, world!");
```

The fifth line, "flux.getActions("messages").newMessage("Hello, world!");", is highlighted with a light gray background.

At the bottom left, it says "untitled\* 5,57 (56)". At the bottom right, it shows "UTF-8 JavaScript ( JSX ) 2".

# Separation of Concerns

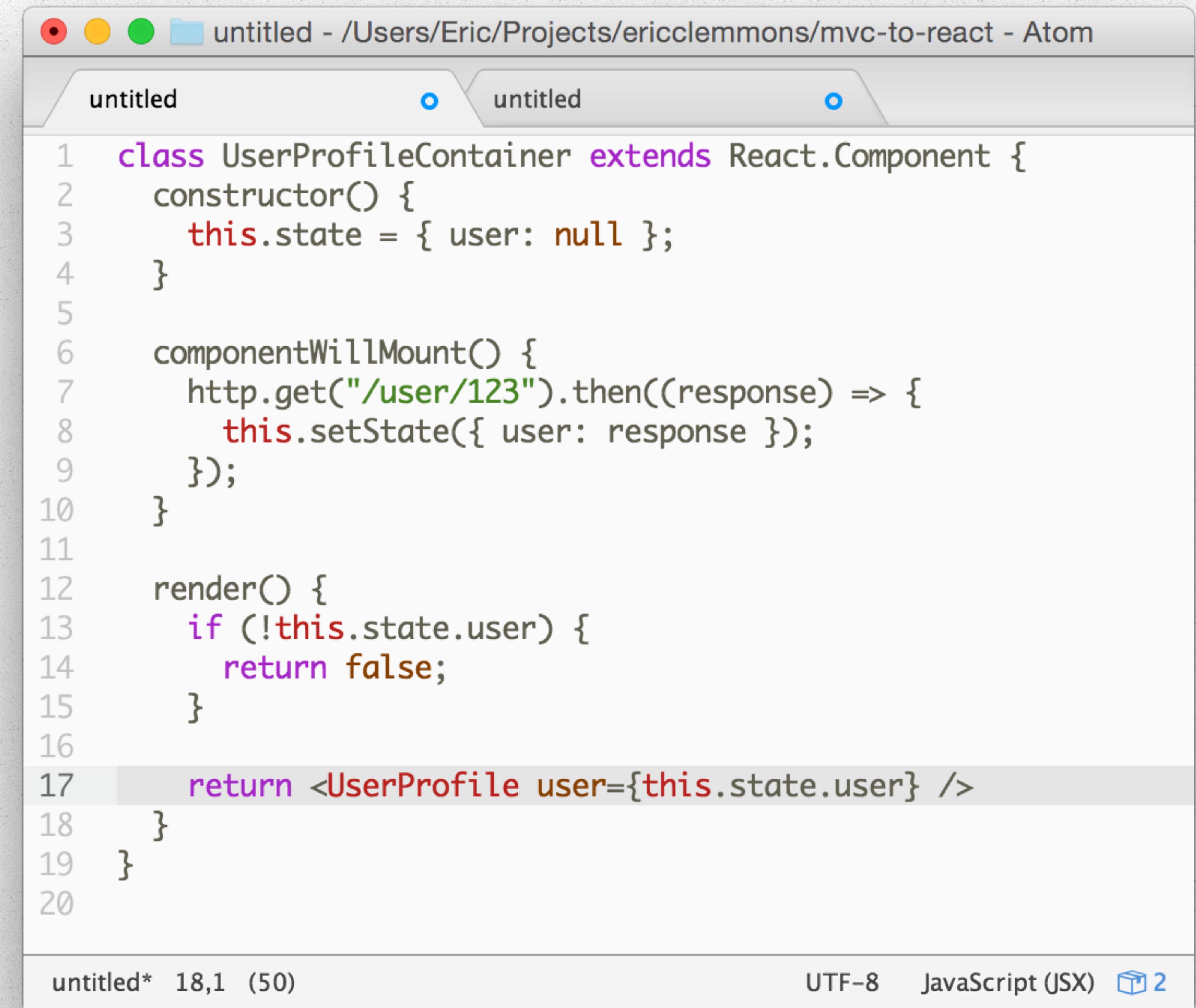
“

The value of separation of concerns is simplifying development and maintenance of [components]. When concerns are well-separated, individual sections can be reused, as well as developed and updated independently.

”

# Complex Container

Separation of Concerns



The screenshot shows an Atom editor window with two tabs open. The active tab, titled 'untitled', contains the following code:

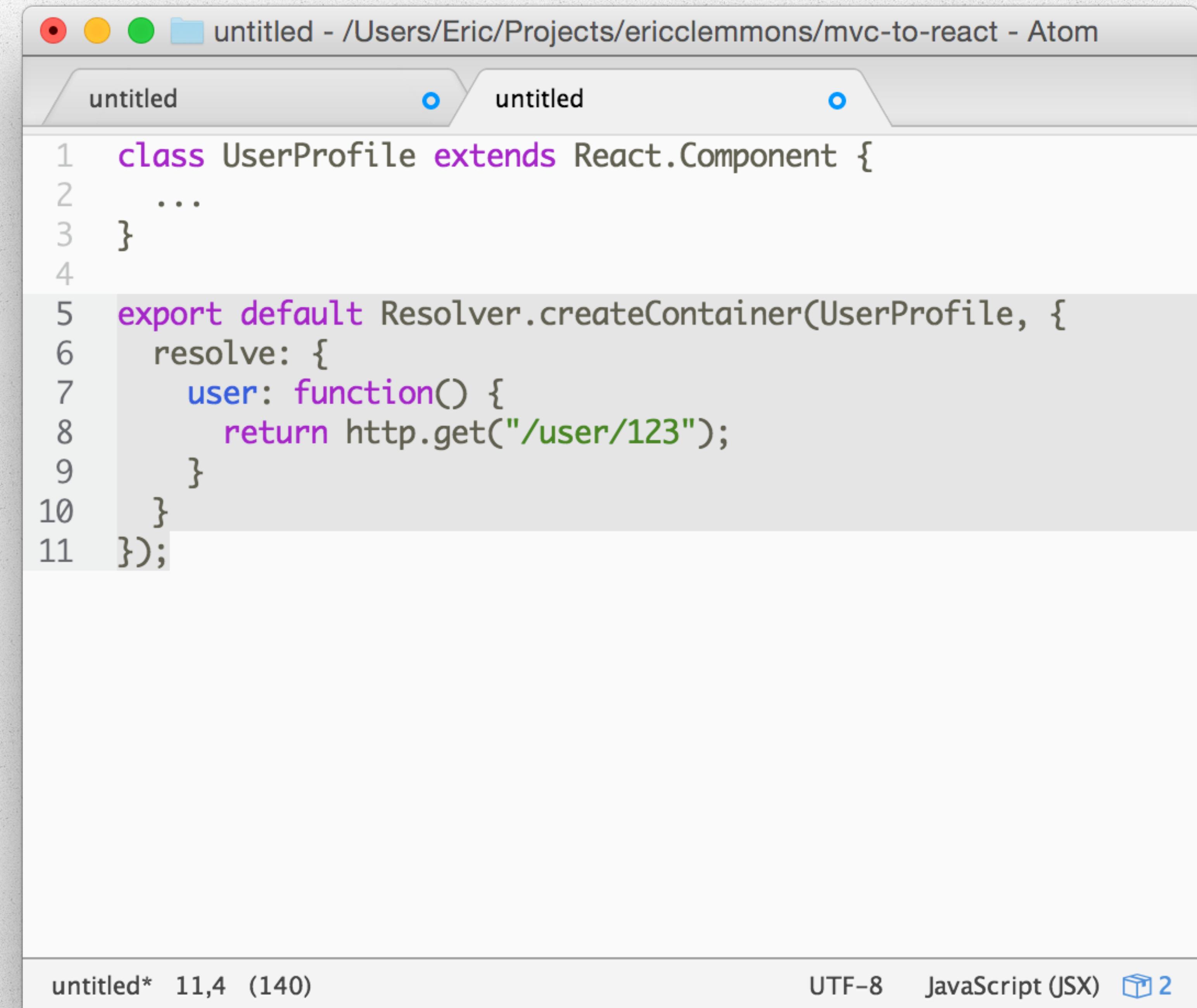
```
1 class UserProfileContainer extends React.Component {  
2     constructor() {  
3         this.state = { user: null };  
4     }  
5  
6     componentWillMount() {  
7         http.get("/user/123").then((response) => {  
8             this.setState({ user: response });  
9         });  
10    }  
11  
12    render() {  
13        if (!this.state.user) {  
14            return false;  
15        }  
16  
17        return <UserProfile user={this.state.user} />  
18    }  
19}  
20
```

The code defines a React component named 'UserProfileContainer'. It includes a constructor to initialize state, a `componentWillMount` lifecycle method to fetch user data from a URL, and a `render` method that returns a `<UserProfile>` component with the fetched user data as a prop. The code is written in JSX syntax.

# Simple Container

Separation of Concerns

Releasing soon:  
[ericclemmons/react-resolver](https://github.com/ericclemmons/react-resolver)



The screenshot shows a window titled "untitled - /Users/Eric/Projects/ericclemmons/mvc-to-react - Atom". The window contains two tabs: "untitled" and "untitled". The "untitled" tab is active and displays the following code:

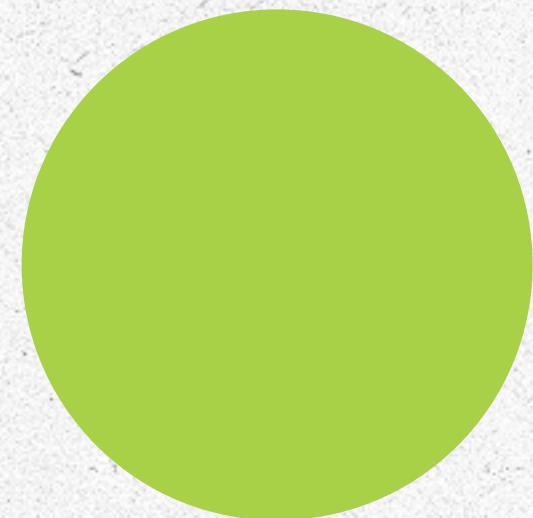
```
1 class UserProfile extends React.Component {  
2   ...  
3 }  
4  
5 export default Resolver.createContainer(UserProfile, {  
6   resolve: {  
7     user: function() {  
8       return http.get("/user/123");  
9     }  
10    }  
11  });
```

The code defines a class-based React component named `UserProfile` and a container function using the `Resolver.createContainer` method. The container function returns a resolver object with a single entry for the `user` key, which returns a promise from `http.get("/user/123")`.

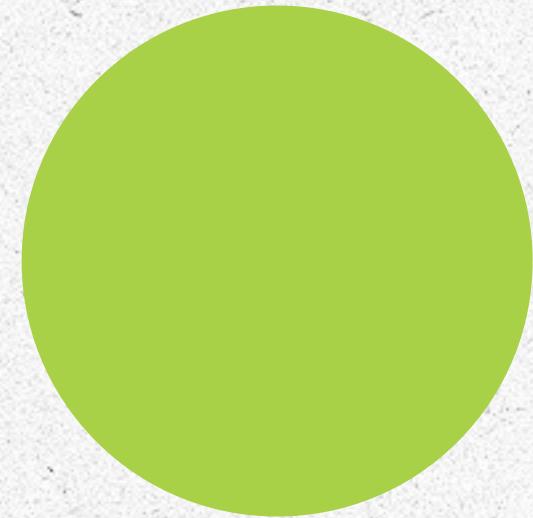
At the bottom of the editor, the status bar shows "untitled\* 11,4 (140)", "UTF-8", "JavaScript (JSX)", and a file icon with the number "2".

# Wrapping Up

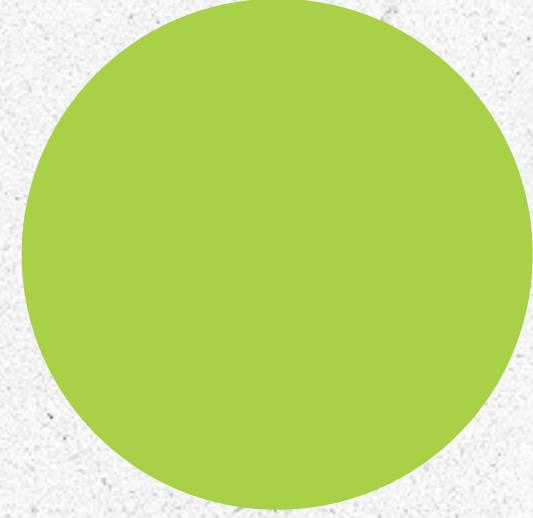
Deliverables



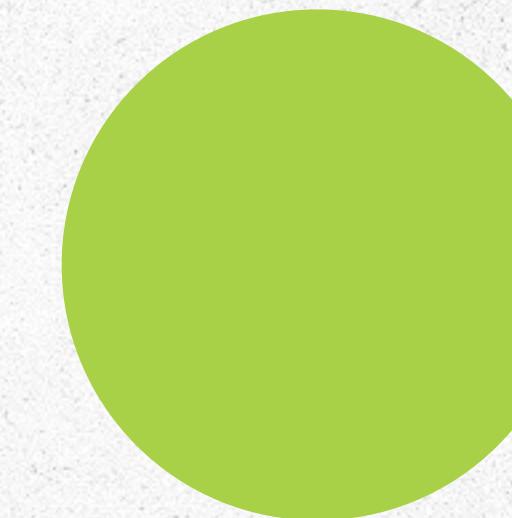
**Isomorphic Development**



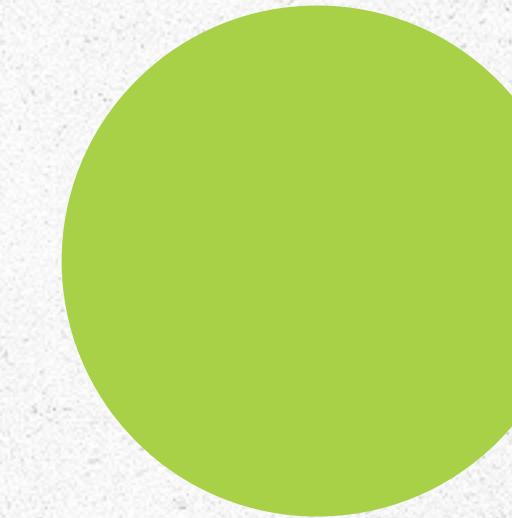
**Rapid Development**



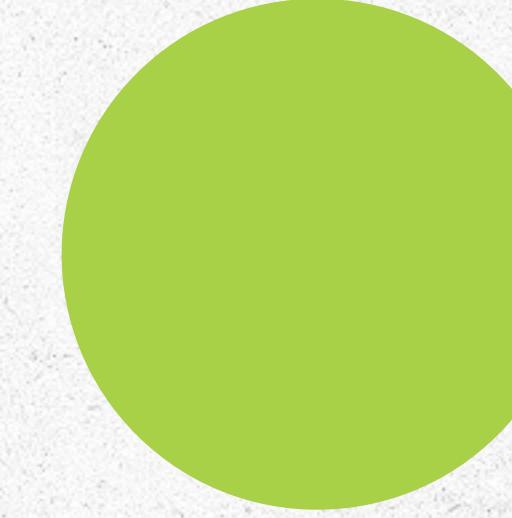
**Clearer Code**



**Separation of Concerns**



**Performance**



**Thriving Ecosystem**

## Learning

<http://facebook.github.io/react/>  
<http://conf.reactjs.com/schedule.html>  
<https://www.tildedave.com/2014/11/15/introduction-to-contexts-in-react-js.html>  
<https://github.com/ericclemmons/mvc-to-react>

## Getting Started

<https://github.com/RickWong/react-isomorphic-starterkit>  
<https://github.com/rackt/react-router/>

## Other Environments

<https://github.com/reactjs/react-rails>  
<https://github.com/reactjs/React.NET>  
<https://github.com/reactjs/react-python>  
<https://github.com/reactjs/react-php-v8js>

## Flux Implementations

<http://goatslacker.github.io/alt/>  
<https://github.com/acdlite/flummox/>  
<https://facebook.github.io/flux/>  
<http://fluxible.io/>  
<http://martyjs.org/>  
<https://github.com/spoike/refluxjs/>

## Community

<http://reactiflux.com/>  
<https://reactiflux.herokuapp.com/>  
<https://join-reactiflux.herokuapp.com/>

Slack

## #general

react, and rails, enabling the es6 and jsx transpilers, and node integration.

**hstove** 8:16 PM joined #general

**buck** 8:55 PM ★ @justin\_gordon: i've derived our work off of your react-webpack-rails tutorial (i'm bbefore on github) ... thanks for that

**justin\_gordon** 8:56 PM anything I can grab from what you've done?  
or care to make a PR?

**buck** 8:56 PM for our implementation i found the alt flux implementaiton by @josh\_perez to be the easiest to onboard our engineers  
i have been hoping to find the time to make a PR for you but i've been unsuccessful  
<https://github.com/goatslacker/alt>

GitHub  
**goatslacker/alt**  
alt - Isomorphic Flux

ericlemonns online

# Thank You!

<https://github.com/ericclemmons/>

<https://twitter.com/ericclemmons/>

