

A photograph of a waterfall cascading down a rocky cliff into a pool of water, surrounded by dense tropical foliage and trees.

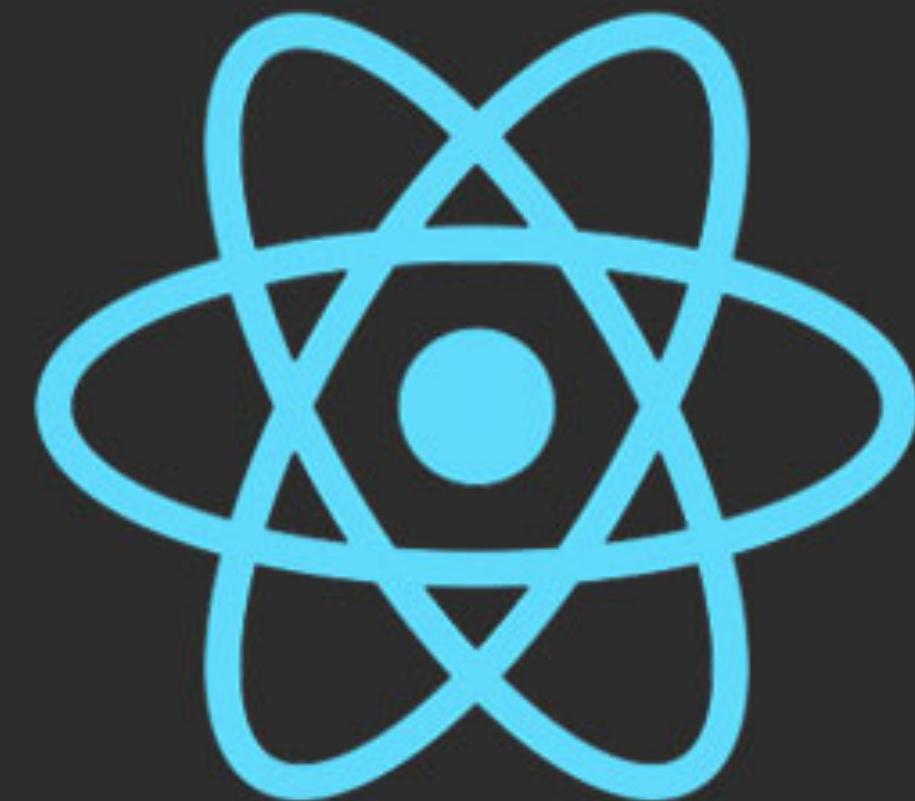
Modern JavaScript applications

Erik Wendel

**JavaScript developer
Working with large-scale JS apps**

Todays agenda

- React.js - DOM library
- Less - css preprocessor
- Patterns for modern JS apps



React

A dark, hazy landscape featuring rolling hills and mountains in the background. The foreground is dominated by a dark, textured surface, possibly volcanic rock or ash, with patches of dry, yellowish-brown grass in the lower left corner.

State

**Simple
Declarative over imperative
Reusable components
Inspired by functional programming**

A black and white photograph of a construction site. In the foreground, there are several large, cylindrical concrete pipes stacked on wooden pallets. Above them, a complex network of scaffolding and steel beams forms a grid-like structure. The background shows more industrial equipment and materials, suggesting a large-scale engineering project.

A component's responsibility:

Data → DOM

Virtual DOM

1. When something *could* have changed, re-render everything
2. Diff new output with previous output
3. Update *only* what has changed

A photograph of a narrow street in a historic town. The buildings are made of light-colored stone or brick, with tiled roofs. There are several signs on the buildings, including one for "POSTERS VINTAGE" and another for "STICK TO BILLS". Power lines run across the sky, and there are trees and bushes along the sidewalk. The overall atmosphere is old-world and charming.

performance?

batch updates (js is fast)

**render
diff
batch**

Components

```
var Example = React.createClass({  
  render: function() {  
    return React.createElement("div", null, "Hello World");  
  }  
});  
  
React.render(Example(), document.getElementById('main'));
```

JSX

fully optional

```
var Example = React.createClass({  
  render: function() {  
    return <div>  
      Hello World!  
    </div>  
  }  
});  
  
React.render(<Example/>, document.getElementById('main'));
```

A scenic view of a coastal town with hills in the background.

Lets talk about data

Props & State

props = immutable
state = changes over time

```
var Comment = React.createClass( {  
  render: function() {  
    return <div>  
      {this.props.name} says: {this.props.text}  
    </div>;  
  }  
});
```

```
React.render(  
  <Comment />,  
  document.getElementById('example')  
)
```

```
var LikeButton = React.createClass({
  getInitialState: function() {
    return {liked: false};
  },
  handleClick: function(event) {
    this.setState({liked: !this.state.liked});
  },
  render: function() {
    var text = this.state.liked ? 'like' : 'don`t like';
    return (
      <p onClick={this.handleClick}>
        You {text} this. Click to toggle.
      </p>
    );
  }
});
```

Data flow

data downwards, events upwards

Reusability

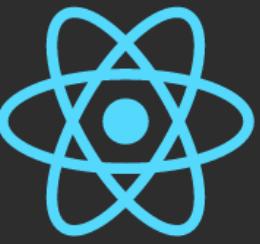
```
var CommentBox = React.createClass({  
  render: function() {  
    return (  
      <div className="commentBox">  
        <h1>Comments</h1>  
        <CommentList />  
        <CommentForm />  
      </div>  
    );  
  }  
});
```

Reusability

```
var CommentList = React.createClass({  
  render: function() {  
    return (  
      <div className="commentList">  
        <Comment author="Pete Hunt">This is one comment</Comment>  
        <Comment author="Jordan Walke">This is *another* comment</Comment>  
      </div>  
    );  
  }  
});
```

Reusability

```
var Comment = React.createClass({  
  render: function() {  
    return (  
      <div className="comment">  
        <h2 className="commentAuthor">  
          {this.props.author}  
        </h2>  
        {this.props.children}  
      </div>  
    );  
  }  
});
```



React Components

Component name, keyword or similar

Searchable database of **React** components

How it works

Every module registered on [NPM](#) using the keyword `react-component` will show up in the list. It really is that simple.

How do I add my component to the list?

1. Ensure your `package.json` file contains an array of keywords which includes `react-component`.
2. Publish your component to NPM (learn how at [npmjs.org](#)).
3. Wait for it to show up! Shouldn't take longer than 10-15 minutes.

Missing any features?

[Let us know!](#) We're always looking for ways to improve.

Who made this? Can I contribute?

Developed and currently hosted by [VaffelNinja](#), but it's an open-source, MIT-licensed solution.

Contributions are [very welcome!](#) Please make sure you read the contribution guidelines.

Latest components

- [react-infinity](#)
- [react-croton](#)
- [react-anything-sortable](#)
- [react-form-data](#)
- [react-bootpag](#)
- [react-typeahead](#)
- [react-form-builder](#)
- [react-clipboard](#)
- [react-microspreadsheet](#)
- [react-formly](#)

Recently updated

- [react-visibility-sensor](#)
- [react-treeview](#)
- [react-anchor](#)
- [merry-go-round](#)
- [react-markdown-textarea](#)
- [react-elements](#)
- [plexus-form](#)
- [react-googlemaps](#)
- [react-widgets](#)
- [react-datepicker](#)

Popular

Lifecycle events:

```
function componentDidMount() { . . . }  
function componentWillUpdate() { . . . }  
function componentDidUpdate() { . . . }  
function componentWillUnmount() { . . . }  
. . .
```

Lifecycle events

```
var Program = React.createClass({  
  componentDidMount: function() {  
    doSomething();  
  }  
  . . .  
})
```

Lifecycle events

```
var Program = React.createClass({  
  componentWillMount: function() {  
    cleanup();  
  }  
  . . .  
})
```

Performance

```
shouldComponentUpdate: function(nextProps, nextState) {  
  return nextProps.id !== this.props.id;  
}
```

The background of the image is a dense, dark forest scene. Sunlight filters through the canopy of leaves and branches, creating bright highlights and deep shadows. The overall atmosphere is mysterious and natural.

FLUX

the MC in MVC

Summary

- React is a view library

Summary

- React is a view library
- it performs really well

Summary

- React is a view library
- it performs really well
- but the main goal is to *simplify* development

A scenic landscape featuring a road leading towards a small, isolated house perched on a hillside. In the background, there are rolling hills and mountains under a dramatic sky filled with dark, heavy clouds. A bright, golden glow from the side suggests either sunrise or sunset, casting a warm light over the scene.

Lets get coding

The background of the image is a dense forest scene. Sunlight filters through the dense canopy of green leaves and branches, creating bright highlights and deep shadows. The overall atmosphere is lush and natural.

LESS
a CSS preprocessor

LESS

- extends CSS with new functionality
- allows for variables, mixins, nesting and more
- although useful, it contains some pitfalls

Nesting

```
.main-page .title {  
  margin-top: 20px;  
}
```

```
.main-page .subtitle {  
  margin-top: 10px;  
}
```

Nesting

```
.main-page {  
  .title {  
    margin-top: 20px;  
  }  
  
  .subtitle {  
    margin-top: 10px;  
  }  
}
```

Nesting: pitfalls

```
.main-page {  
  .sub-section {  
    .sidebar {  
      .selected {  
        color: green;  
      }  
    }  
  }  
}
```

Nesting: pitfalls

```
.main-page .sub-section .sidebar .selected {  
  color: green;  
}
```

Variables

- great way to ensure consistency
- simplifies refactoring

```
@base: #f938ab;  
@margin: 20px;  
  
.main-page {  
  color: @base;  
  margin-top: @margin;  
}
```

Mixins

allows us to group styles for reuse

```
.box-shadow {  
  -webkit-box-shadow: 10px 10px 5px 0px rgba(0,0,0,0.75);  
  -moz-box-shadow: 10px 10px 5px 0px rgba(0,0,0,0.75);  
  box-shadow: 10px 10px 5px 0px rgba(0,0,0,0.75);  
}
```

```
.mybox {  
  .box-shadow;  
}
```

Parametrized mixins

```
.borderRadius(@radius) {  
  -webkit-border-radius: @radius;  
  -moz-border-radius: @radius;  
  border-radius: @radius;  
}
```

Mixins with default values

```
.borderRadius(@radius: 5px) {  
  -webkit-border-radius: @radius;  
  -moz-border-radius: @radius;  
  border-radius: @radius;  
}
```

Modularity

- less allows for importing styles from other files
- this helps split up "the huge untouchable css file"

Modularity

```
@import "base.less"  
@import "icons.less"  
@import "fonts.less"  
@import "frontpage.less"  
@import "otherpage.less"
```

A wide-angle photograph of a rural landscape. A paved road curves from the bottom left towards the horizon. The foreground is filled with tall, dry grass. In the background, there are rolling hills and mountains. The sky is filled with large, dark clouds, with patches of bright light breaking through on the right side, suggesting either sunrise or sunset.

Modern JS apps

some ground rules

1. Separation of Concerns

one thing handles the DOM

another stores data

2. Files are cheap

Use Browserify to modularize.

3. No Data in the DOM

no!

4. Avoid global variables

`var a = 12;`

vs

`a = 12;`

**5. Find a good home
for state**

if it doesn't change, its props

*"The secret to building large apps is **never** to build large apps.
Break your applications into **small** pieces. Then, **assemble**
those testable, bites-sized pieces into your big application"*

- **Justin Meyer**

*"The more **tied** components are to each other, the **less reusable** they will be, and the more **difficult** it becomes to make changes to one without accidentally affecting another"*

- **Rebecca Murphey**

**Any questions?
Thanks!**

twitter: @ewndl