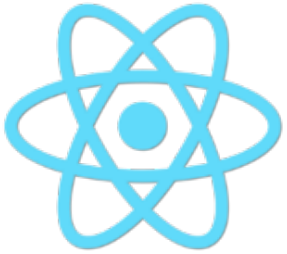
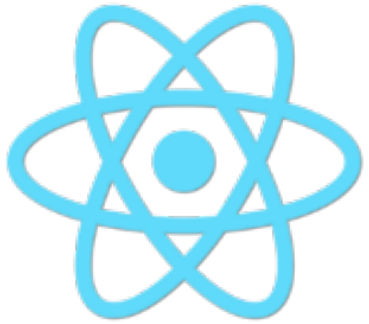
A World of Components

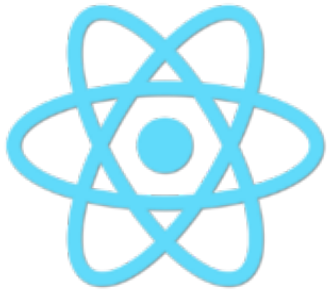


Why React?



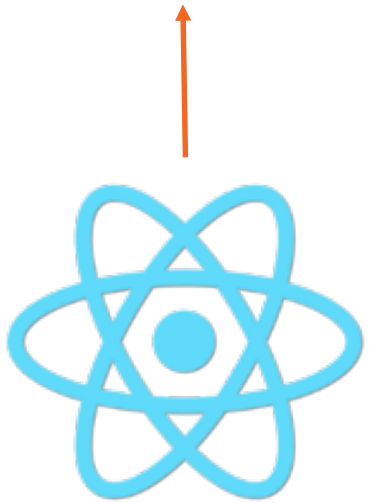
- Fast
- Composable
- Pluggable
- Isomorphic Friendly
- Simple
- Battle Proven

Battle Proven



M

V

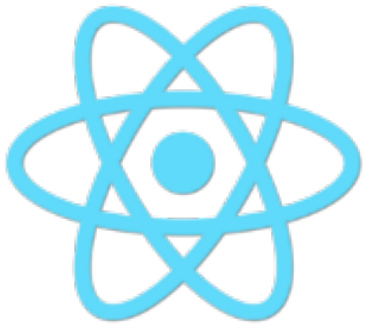


C

M

V

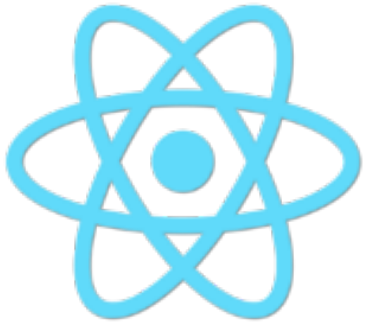
C



The Risk of Two-Way Binding

Unpredictable | Cascading updates | Tricky debugging

JSX



"HTML" in JavaScript

Differences: className, htmlFor

Compiles to JavaScript

Optional

```
"use strict";

var React = require('react');

var AboutPage = React.createClass({
    render: function() {
        return (
            <div>
                <h1>About</h1>
                <p>This is a React and Flux demo project.</p>
            </div>
        );
    }
});

module.exports = AboutPage;
```

JSX Compiles to JS

```
<h1 color="red">Heading here</h1>
```

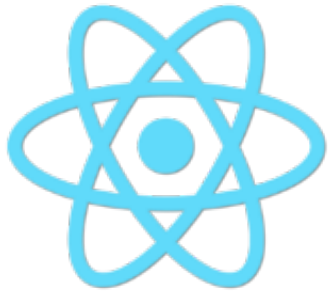
```
React.createElement("h1", {color: "red"}, "Heading here")
```

```jsx
var createAuthorRow = function(author) {
    return (
        <tr key={author.id}>
            <td>{author.id}</td>
            <td>{author.firstName} {author.lastName}</td>
        </tr>
    );
};
return (
    <table className="table">
        <thead>
            <th>ID</th>
            <th>Name</th>
        </thead>
        <tbody>
            {this.props.authors.map(createAuthorRow)}
        </tbody>
    </table>
);
```

```jsx
var createAuthorRow = function(author) {
    return (                //Note that you need to specify a key when iterating.
        React.createElement("tr", {key: author.id},
            React.createElement("td", null, author.id),
            React.createElement("td", null, author.firstName, " ", author.lastName)
        )
    );
};
return (
    React.createElement("table", {className: "table"},
        React.createElement("thead", null,
            React.createElement("th", null, "ID"),
            React.createElement("th", null, "Name")
        ),
        React.createElement("tbody", null,
            this.props.authors.map(createAuthorRow)
        )
    )
);
```

"JS" in HTML

```
<div ng-repeat="user in users">
{{#each user in users}}
data-bind="foreach: users">
```



"HTML" in JS

```
{users.map(createUserRow)}
```

Must stay in sync.

No explicit interface!

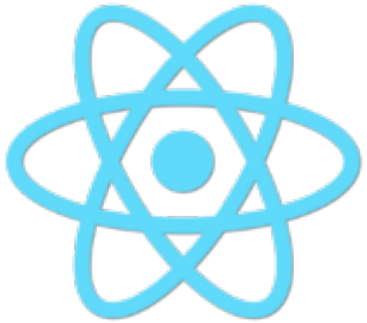
JSX Friendly Editors

2015

Why Lint?

- Avoid errors
- Enforce best practices
- Maintain code consistency
- Many to choose from
  - JSLint
  - JSHint
  - ESHint

Why Virtual DOM?



Updating the DOM is expensive

The Virtual DOM

Without Virtual DOM

Blindly update DOM

using new state.

With Virtual DOM

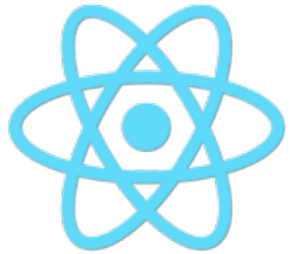
Compare DOM's current state to desired new state.

Update the DOM in the most

efficient way.

Removing a Row…

Redraw table



Removes the row

Hard to argue with the results…

```
$\displaystyle \lim_{n\to\infty}2^n\underbrace{\sqrt{2-\sqrt{2+\sqrt{2+\dots+\sqrt2}}}}_{n \textrm{ square roots}}$.
$\displaystyle \lim_{n\to\infty}2^n\underbrace{\sqrt{2-\sqrt{2+\sqrt{2+\dots+\sqrt2}}}}_{n \textrm{ square roots}}$.
$\displaystyle \lim_{n\to\infty}2^n\underbrace{\sqrt{2-\sqrt{2+\sqrt{2+\dots+\sqrt2}}}}_{n \textrm{ square roots}}$.
```

$$\lim_{n\to\infty}2^n\underbrace{\sqrt{2-\sqrt{2+\sqrt{2+\cdots+\sqrt2}}}}_{n \textrm{ square roots}}.$$

$$\lim_{n\to\infty}2^n\underbrace{\sqrt{2-\sqrt{2+\sqrt{2+\cdots+\sqrt2}}}}_{n \textrm{ square roots}}.$$

$$\lim_{n\to\infty}2^n\underbrace{\sqrt{2-\sqrt{2+\sqrt{2+\cdots+\sqrt2}}}}_{n \textrm{ square roots}}.$$

$$\lim_{n\to\infty}2^n\underbrace{\sqrt{2-\sqrt{2+\sqrt{2+\cdots+\sqrt2}}}}_{n \textrm{ square roots}}.$$

$$\lim_{n\to\infty}2^n\underbrace{\sqrt{2-\sqrt{2+\sqrt{2+\cdots+\sqrt2}}}}_{n \textrm{ square roots}}.$$

$$\lim_{n\to\infty}2^n\underbrace{\sqrt{2-\sqrt{2+\sqrt{2+\cdots+\sqrt2}}}}_{n \textrm{ square roots}}.$$

Performance: Go Even *Faster*

shouldComponentUpdate

PureRenderMixin + immutability

Virtual DOM: More Than Performance

Synthetic Events

Isomorphic Support

React Native

Summary

- Fast, Pluggable, Battle Proven
- JSX
  - "HTML" that compiles to JS
  - Strict compile time checking
- Virtual DOM
  - Performance
  - Simple Mental model
  - Synthetic Events
  - Enables Isomorphic rendering and React Native