React Core Concepts

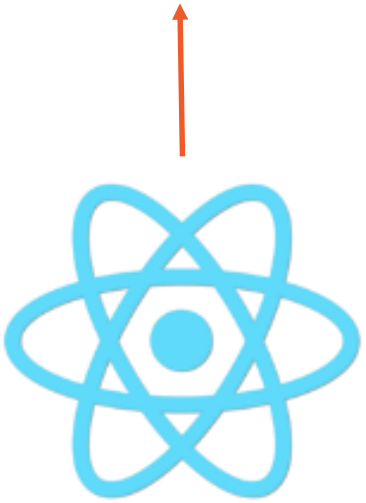
Cory House
PRINCIPAL CONSULTANT

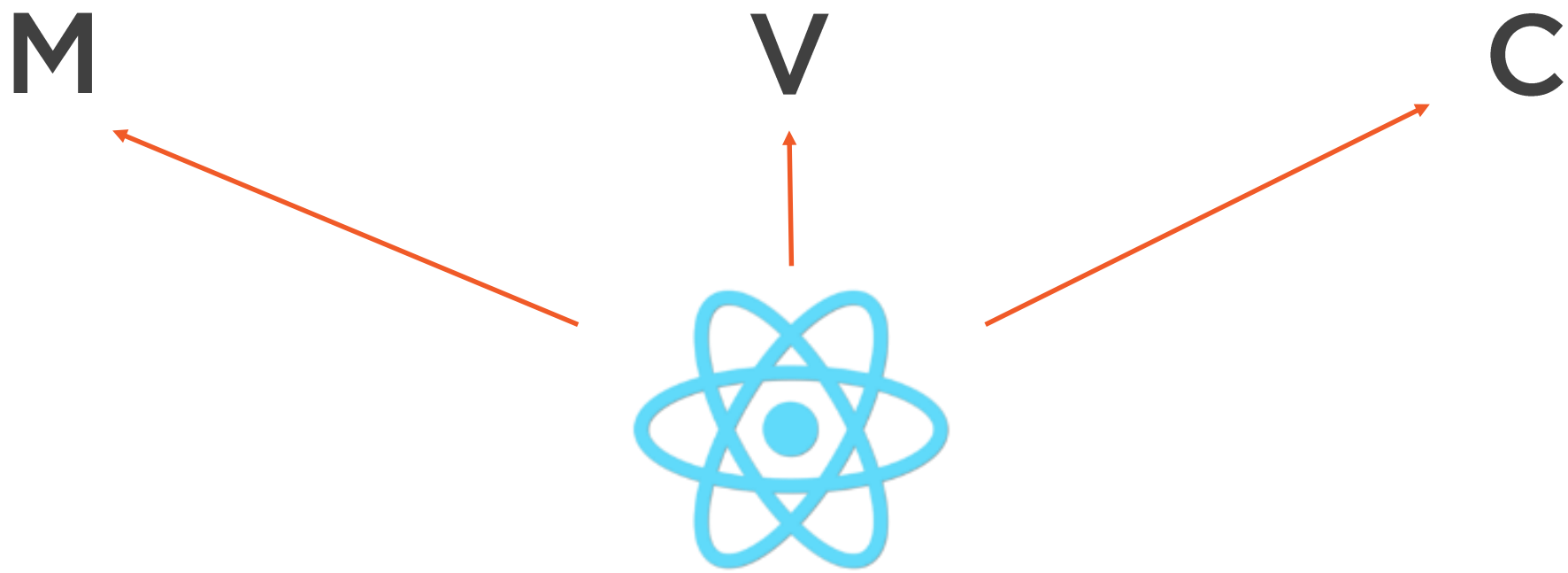
@housecor reactjsconsulting.com

Agenda

- React and MVC
- JSX
- Virtual DOM
- Separation of concerns
- Ways to declare React components
- Create our first React components

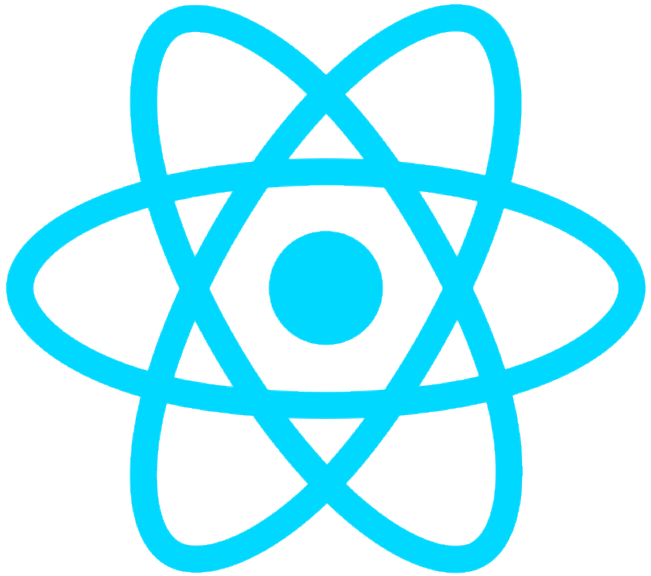
M V C




M
V
C

JSX

JSX



"HTML" in JavaScript

Differences: className, htmlFor

Compiles to JavaScript

Optional


This baby is clearly adorable.
Imagine an ugly baby here.

```jsx
import React from "react";

function About() {
  return <h1>About</h1>;
}
```

```jsx
import React from "react";

function About() {
  return <h1>About</h1>;
}
```

Babel transpiles JSX to a plain func call

```jsx
function About() {
  return React.createElement('h1', null, 'About');
}
```

JSX Compiles to JS

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

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

SETTINGS

- [ ] Evaluate
- [x] Line Wrap
- [ ] Minify
- [ ] Prettify
- [ ] File Size
- [ ] Time Travel

Source Type

Module

PRESETS

- [ ] es2015
- [ ] es2015-loose
- [ ] es2016
- [ ] es2017
- [ ] stage-0
- [ ] stage-1
- [ ] stage-2
- [ ] stage-3
- [x] react
- [ ] flow
- [ ] typescript

ENV PRESET

v7.4.2

```
1 import React from "react";
2
3 function About() {
4   return <h1>About</h1>;
5 }
6
```

```
1 import React from "react";
2
3 function About() {
4   return React.createElement("h1", null, "About");
5 }
```

SETTINGS

- [ ] Evaluate
- [x] Line Wrap
- [ ] Minify
- [ ] Prettify
- [ ] File Size
- [ ] Time Travel

Source Type

Module

PRESETS

- [ ] es2015
- [ ] es2015-loose
- [ ] es2016
- [ ] es2017
- [ ] stage-0
- [ ] stage-1
- [ ] stage-2
- [ ] stage-3
- [x] react
- [ ] flow
- [ ] typescript

ENV PRESET

v7.4.2

```
1 import React from "react";
2
3 function About() {
4   return <h1>About</h1>;
5 }
6
```

```
1 import React from "react";
2
3 function About() {
4   return React.createElement("h1", null, "About");
5 }
```

SETTINGS

- [ ] Evaluate
- [x] Line Wrap
- [ ] Minify
- [ ] Prettify
- [ ] File Size
- [ ] Time Travel

Source Type

Module

PRESETS

- [ ] es2015
- [ ] es2015-loose
- [ ] es2016
- [ ] es2017
- [ ] stage-0
- [ ] stage-1
- [ ] stage-2
- [ ] stage-3
- [x] react
- [ ] flow
- [ ] typescript

ENV PRESET

v7.4.2

```
1 import React from "react";
2
3 function About() {
4   return <div>
5     <h1>About</h1>
6     <p>This is the about page.</p>
7   </div>
8 }
9
```

```
1 import React from "react";
2
3 function About() {
4   return React.createElement("div", null,
  React.createElement("h1", null, "About"),
  React.createElement("p", null, "This is the about
  page."));
5 }
```

SETTINGS

- [ ] Evaluate
- [x] Line Wrap
- [ ] Minify
- [ ] Prettify
- [ ] File Size
- [ ] Time Travel

Source Type

Module

PRESETS

- [ ] es2015
- [ ] es2015-loose
- [ ] es2016
- [ ] es2017
- [ ] stage-0
- [ ] stage-1
- [ ] stage-2
- [ ] stage-3
- [x] react
- [ ] flow
- [ ] typescript

ENV PRESET

v7.4.2

```
1  import React from "react";
2
3  function Users() {
4    return <table>
5      <thead>
6        <tr>
7        <td>ID</td>
8        <td>Name</td>
9        </tr>
10     </thead>
11     <tbody>
12       <tr>
13         <td>1</td>
14         <td>Cory</td>
15       </tr>
16     </tbody>
17   </table>
18 }
19
```

```
1  import React from "react";
2
3  function Users() {
4    return React.createElement("table", null,
   React.createElement("thead", null,
   React.createElement("tr", null,
   React.createElement("td", null, "ID"),
   React.createElement("td", null, "Name"))),
   React.createElement("tbody", null,
   React.createElement("tr", null,
   React.createElement("td", null, "1"),
   React.createElement("td", null, "Cory"))));
5  }
```

```
import React from "react";

function About() {
  return <h1>About</h1>;
}
```

Easier to read
Easier to type
More friendly to designers

Inline Styles

```
function About() {
  return (
    <h1 style={{
      color: 'white',
      backgroundColor: "#000000”,
      height: 20

    }}>
      About
    </h1>
  )
}
```

Uses JS
Note camelCase
Size in pixels inferred
Optional
Use sparingly

HTML vs JSX

class -> className

for -> htmlFor

camelCased attributes

- tabindex -> tabIndex

DOM Elements

React implements a browser-independent DOM system for performance and cross-browser compatibility. We took the opportunity to clean up a few rough edges in browser DOM implementations.

In React, all DOM properties and attributes (including event handlers) should be camelCased. For example, the HTML attribute `tabindex` corresponds to the attribute `tabIndex` in React. The exception is `aria-*` and `data-*` attributes, which should be lowercased. For example, you can keep `aria-label` as `aria-label`.

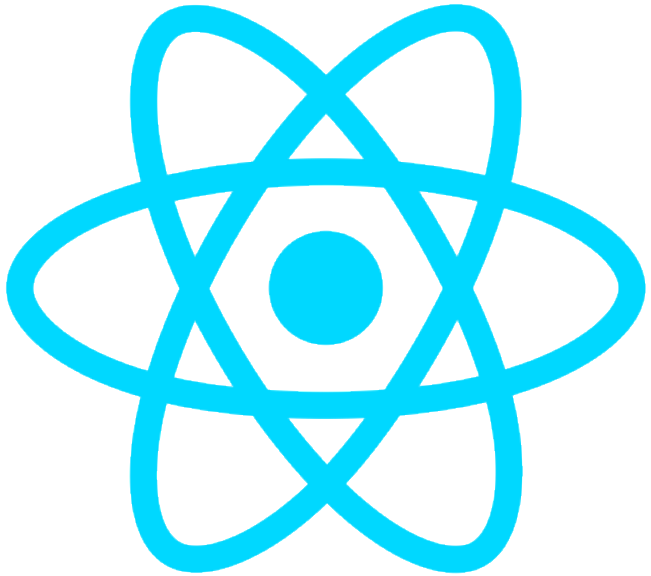
Differences In Attributes

There are a number of attributes that work differently between React and HTML:

Typo?
JSX tells you what line.

Virtual DOM

Why Virtual DOM?



Updating the DOM is expensive

The Virtual DOM

Without Virtual DOM

Blindly update DOM

using new state.

With Virtual DOM

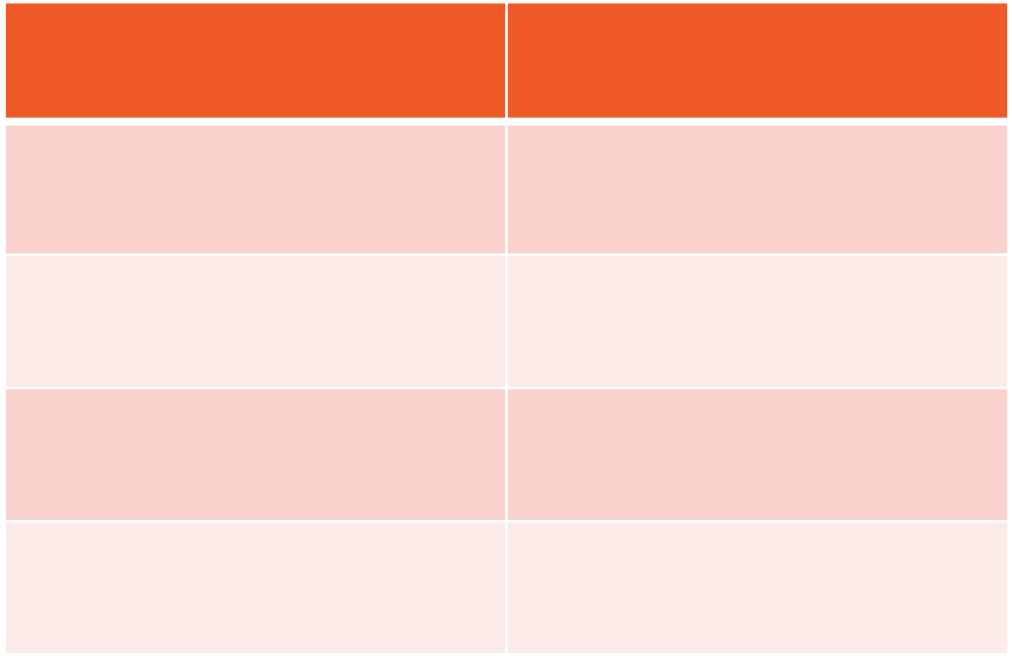
Compare DOM's current state to desired new state.

Update the DOM in the most

efficient way.

Removing a Row...

Many React competitors..



Redraw table

Removes the row

Hard to argue with the results...

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

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

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

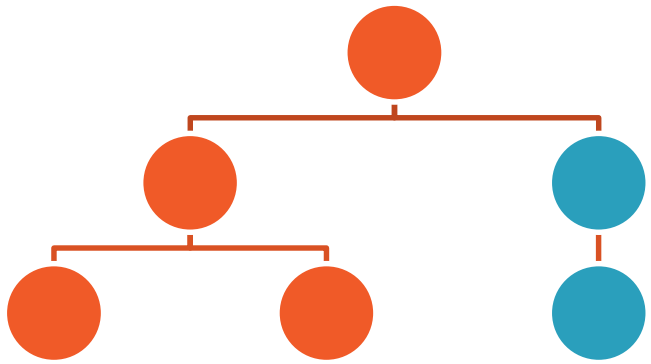
$$\lim_{n\to\infty}2^n\underbrace{\sqrt{2-\sqrt{2+\sqrt{2+\cdots+\sqrt2}}}}_{n \textrm{ square roots}}.$$

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

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

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

Virtual DOM: More Than Performance



Synthetic events

Isomorphic support

React Native

Separation of Concerns

"JS" in HTML

```
<div ng-repeat="user in users">
{{#each user in users}}
<li v-for="user in users">
```

"HTML" in JS

```
{users.map}
```

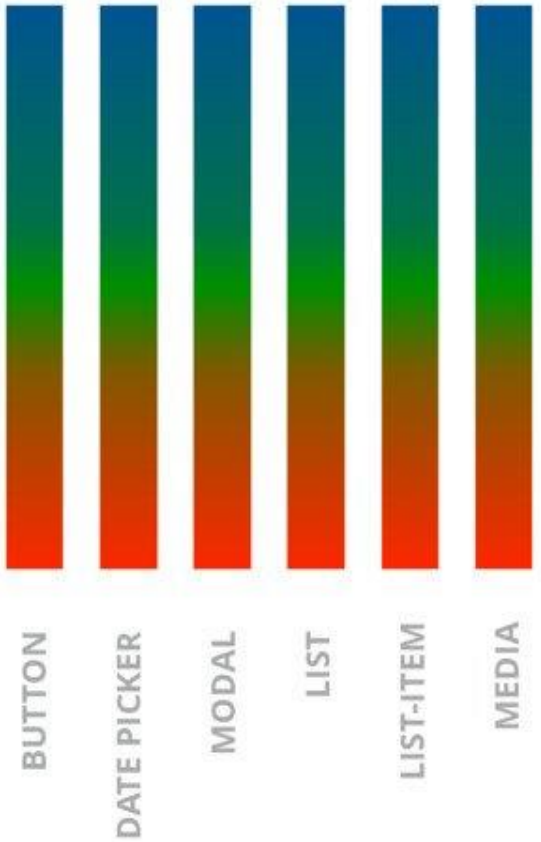
Must stay in sync.

No explicit interface!

Separation of Concerns

Separation of Concerns

(only, from a different point of view)



speakerdeck.com/didoo/let-there-be-peace-on-css

Integrating intertwined
concerns helps debugging

Four ways to create React components

Ways to Create Components

- createClass
- ES class
- Function
- Arrow function

createClass Component

```
var HelloWorld = React.createClass({
  render: function () {
    return (
      <h1>Hello World</h1>
    );
  }
});
```

JS Class Component

```
class HelloWorld extends React.Component {
  constructor(props) {
    super(props);
  }

  render() {
    return (
      <h1>Hello World</h1>
    );
  }
}
```

Function Component

```
function HelloWorld(props) {
  return (
    <h1>Hello World</h1>
  );
}
```

Arrow Function

```
const HelloWorld = (props) => <h1>Hello World</h1>
```

Function Component Benefits

- Easier to understand
- Avoid `this` keyword
- Less transpiled code
- High signal-to-noise ratio
- Enhanced code completion / intellisense
- Bloated components are obvious
- Easy to test
- Performance
- Classes may be removed in future

PRESETS

- [ ] es2015
- [ ] es2015-loose
- [ ] es2016
- [ ] es2017
- [ ] stage-0
- [ ] stage-1
- [ ] stage-2
- [ ] stage-3
- [x] react

ENV PRESET 1.6.2

- [x] Enabled

BROWSERS

> 2%, ie 11, safari > 9

ELECTRON 1.8

NODE 8.9

BUILT-INS

SPEC

LOOSE

PLUGINS

v6.26.0

```
1  import React, { Component } from "react";
2
3  class Hi extends Component {
4    render() {
5      return (
6        <h1>hi</h1>
7      );
8    }
9  }
10
11 function HiFunc() {
12   return <h1>hi</h1>;
13 }
```

Class

```
16
17 var Hi = function (_Component) {
18   _inherits(Hi, _Component);
19
20   function Hi() {
21     _classCallCheck(this, Hi);
22
23     return _possibleConstructorReturn(this,
   (Hi.__proto__ ||
   Object.getPrototypeOf(Hi)).apply(this, arguments));
24   }
25
26   _createClass(Hi, [{
27     key: "render",
28     value: function render() {
29       return _react2.default.createElement(
30         "h1",
31         null,
32         "hi"
33       );
34     }
35   }]);
36
37   return Hi;
38 }(_react.Component);
39
```

Function

```
40 function HiFunc() {
41   return _react2.default.createElement(
42     "h1",
43     null,
44     "hi"
45   );
46 }
```

When Should I Use Each? Pre 16.8...

Class Component	Function Components
State	Everywhere else ☺
Refs	
Lifecycle methods	

When Should I Use Each? After 16.8...

Class Component	Function Components
componentDidError	Everywhere else ☺
getSnapshotBeforeUpdate	

Prefer function components.

We'll create both class and function components.

Summary

JSX

- "HTML" that compiles to JS
- Strict compile time checking

Virtual DOM

- Performance
- Simple Mental model
- Synthetic Events
- Enables server rendering and React Native

We'll create both classes and functions

Next up: Create React components