Here's the Plan

- **React component creation approaches**
- **Container vs Presentational Components**

Four+ Ways to Create React Components?!

Ways to Create Components

- **ES5 createClass**
- **ES6 class**
- **ES5 stateless function**
- **ES6 stateless function**
- **Many more...**

ES5 Class Component

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

ES6 Class Component

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

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

React in ES6

No autobind

PropTypes declared separately

Default props declared separately

Set initial state in constructor

ES5 Stateless Functional Component

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

ES6 Stateless Functional Component

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

```jsx
import React from 'react';

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

  sayHi(event) {
    alert(`Hi ${this.props.name}`);
  }

  render() {
    return (
      <div>
        <a
          href="#"
          onClick={this.sayHi.bind(this)}>Say Hi</a>
      </div>
    );
  }
}

HelloWorld.propTypes = {
  name: React.PropTypes.string.isRequired
};

export default HelloWorld;
```

```jsx
import React from 'react';

const HelloWorld = (props) => {
  const sayHi = (event) => {
    alert(`Hi ${props.name}`);
  };

  return (
    <div>
      <a
        href="#"
        onClick={sayHi}>Say Hi</a>
    </div>
  );
};

HelloWorld.propTypes = {
  name: React.PropTypes.string.isRequired
};

export default HelloWorld;
```

Stateless Functional Components: 9 Benefits

```
import React from 'react';

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

  sayHi(event) {
    alert(`Hi ${this.props.name}`);
  }

  render() {
    return (
      <div>
        <a
          href="#"
          onClick={this.sayHi.bind(this)}>Say Hi</a>
      </div>
    );
  }
}

HelloWorld.propTypes = {
  name: React.PropTypes.string.isRequired
};

export default HelloWorld;
```

```
import React from 'react';

const HelloWorld = (props) => {
  const sayHi = (event) => {
    alert(`Hi ${props.name}`);
  };

  return (
    <div>
      <a
        href="#"
        onClick={sayHi}>Say Hi</a>
    </div>
  );
};

HelloWorld.propTypes = {
  name: React.PropTypes.string.isRequired
};

export default HelloWorld;
```

No class needed

Avoid `this` keyword

Enforced best practices

High signal-to-noise ratio

Enhanced code completion / intellisense

Bloated components are obvious

Easy to understand

Easy to test

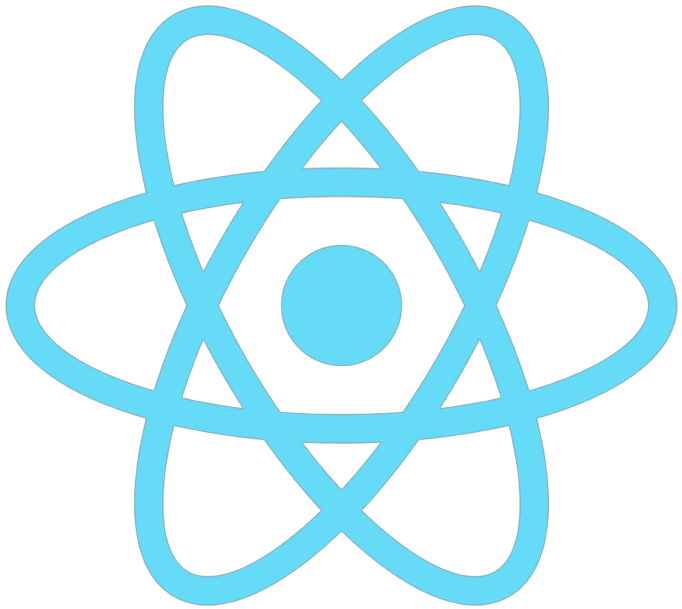
Performance

Use stateless functional components when possible.

When Should I Use Each?

Class Component	Stateless Components
State	Everywhere else ☺
Refs	
Lifecycle methods	
Child functions (for performance)	

Other Ways to Create Components



Object.create

Mixins

Parasitic Components

StampIt

More info: bit.ly/react-define-component

Container vs Presentation Components

Most components

Container	Presentation
Little to no markup	Nearly all markup
Pass data and actions down	Receive data and actions via props
Knows about Redux	Doesn't know about Redux
Often stateful	Typically functional components

Alternative Jargon

Container	Presentational
Smart	Dumb
Stateful	Stateless
Controller View	View

”When you notice that some components don’t use props they receive but merely forward them down...it’s a good time to introduce some container components.”

Dan Abramov

Wrap up

ES5 createClass

ES6 Class

ES5 Stateless Function

ES6 Stateless Function

Many more!

Container vs Presentation Components

Next up: Let's start building!