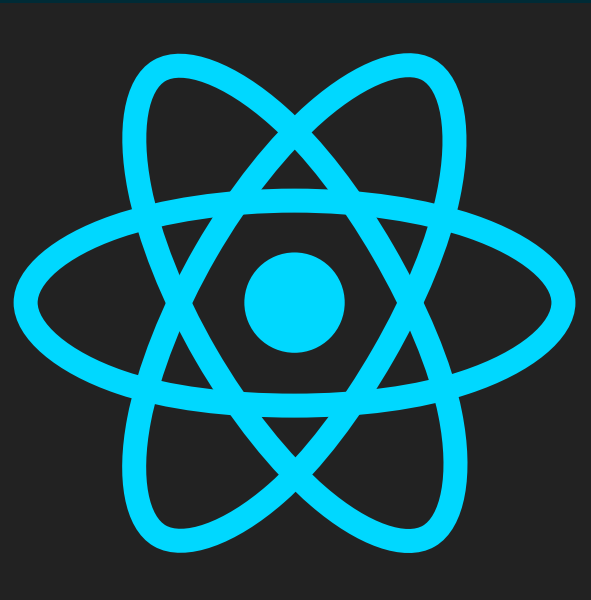


# React.js

## Introduction / Getting Started



Jeff Barczewski

[codewinds.com](https://codewinds.com)



# React.js Workshop

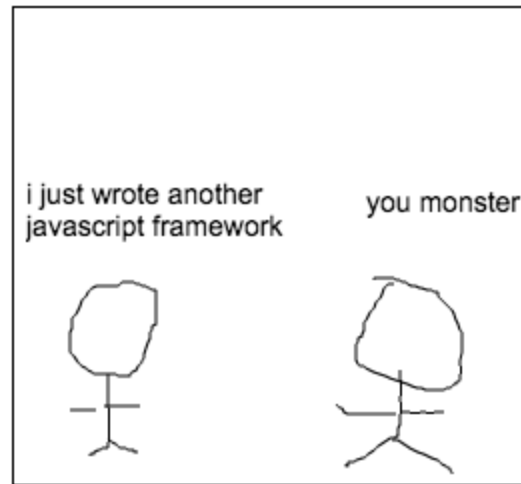
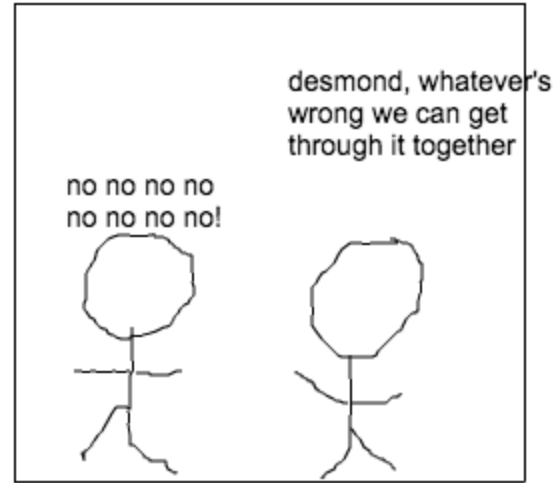
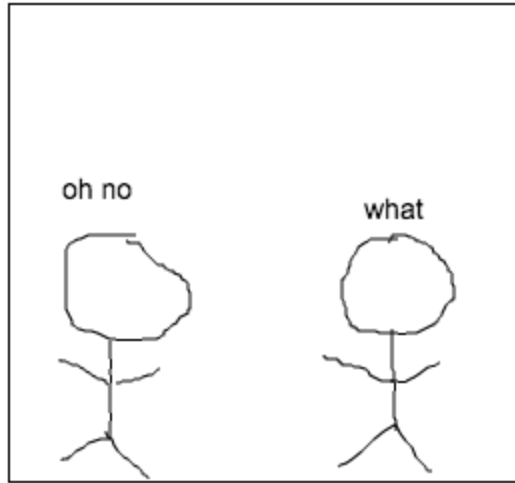
- Install Node.js <http://nodejs.org>
- clone or download/extract zip from <http://codewinds.com/jsconf-react>
- If using windows, use <http://codewinds.com/jsconf-react-win> instead
- cd into new dir and npm install

# Workshop resources

- <http://codewinds.com/jsconf>
- [jeff@codewinds.com](mailto:jeff@codewinds.com)
- [@jeffbski](#)

# Who am I?

- @jeffbbski jeff@codewinds.com
- 26 yrs professional developer
- Last 3+ yrs fulltime JavaScript / Node.js
- Created CodeWinds to publish video training courses



# Question for you

What are your greatest difficulties or challenges that you face when building web apps?

[codewinds.com](https://codewinds.com)

*CodeWinds*

# Goals

- Learn why React.js is special
- Core concepts
- Implement features in React.js

# What makes React.js special?

- Simple to learn
- Composable Components
- Declarative
- Easy to use with existing projects



# Quick demos

- React.js
- React Native

# React.js core concepts

- Just the view
- Virtual DOM
- Components
- Top down data flow

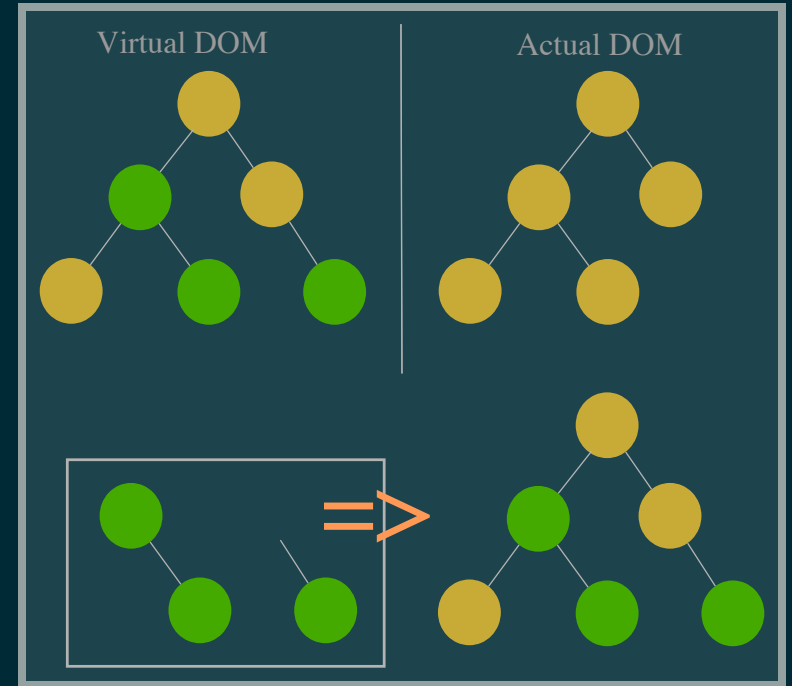
# Just the view

```
class Greeting extends React.Component {  
  render() { // the heart of React.js  
    // pass in any type of js objects, methods, or fns  
    return <div>{ this.props.user.name }</div>;  
  }  
}
```

---

# Virtual DOM

- Render virtual tree
- Fast diff
- Adapters to DOM, ...



# Components

```
class Name ... // can be in another file
class Greeting extends React.Component {
  render() { // focus on my props
    return (
      <div>
        Hello
        <Name value={this.props.name} />
      </div>
    );
  }
}
```

---

# Top down data flow

```
// from main.js
```

```
<App name={appName} items={items} />
```

---

```
// elsewhere in App's render()
```

```
return (<div>
```

```
  <MainTitle title={this.props.name} />
```

```
  <PagingList collection={this.props.items} />
```

```
</div>);
```

---

```
// elsewhere in MainTitle's render()
```

# React.js API - JSX

```
return ( // JSX is optional but really give it a try
```

```
  <div>
```

```
    <h1>{this.props.title}</h1>
```

```
    <MyDate val={this.props.date} />
```

```
  </div>
```

```
);
```

---

```
// transforms into
```

```
return React.createElement('div', {}, [
```

```
  React.createElement('h1', {}, this.props.title),
```

```
  React.createElement(MyDate, { val: this.props.date }, [])
```

```
]);
```

# React.js API - render, props

```
var mainDiv = document.querySelector('#mainDiv');  
React.render(<App title={myTitle} items={myItems} />,  
             mainDiv); // apply here
```

---

```
// elsewhere in App's render()
```

```
return (  
  <div>  
    <h1>{this.props.title}</h1>  
    <ul>{this.props.items.map(i =>  
      <li key={i.id}>{i.name}</li>)}  
    </ul>  
    ...  
  )
```



# Let's code

- `cd jsconf-react`
- `npm run watch`
- Opens browser to <http://localhost:3005>

# Survey Project

- <http://survey.codewinds.com/>
- <http://survey.codewinds.com/admin.html>
- <http://survey.codewinds.com/polls>

# First steps

- Fetch /polls with axios
- Display all the questions in a list
- Shuffle the polls with shuffle-array
- Display the first shuffled poll w/choices

# React.js API - JSX p2

```
const errorStyle = {
  color: 'red',
  marginRight: '10px'
};

const html = marked('The quick brown fox...');

return (
  <div className="foo bar" data-baz="a"> { /* comment */ }
    <span style={errorStyle}>{this.props.msg}</span>
    <input autoFocus="true" onClick={this.myfn.bind(this)} />
    <div dangerouslySetInnerHTML={{ __html: html }} />
  </div> );
```

---

# React.js API - state, events

```
class MyComp extends React.Component {  
  constructor(...args) {  
    super(...args);  
    this.state = { count: 0 };  
  }  
  render() { return (  
    <button onClick={this.clicked.bind(this)} >  
      {this.state.count} </button> );  
  }  
  clicked(e) { this.setState({ count: this.state.count + 1 }); }  
}
```

---

# React.js API - forms

*// uncontrolled comps, set orig value, can watch w/events*

```
return (<div>  
  <input name="first" defaultValue={this.props.first}  
    onChange={this.firstChanged.bind(this)} />  
  <textarea name="foo" defaultValue={this.props.foo} />  
</div>); // <select> also works similarly
```

---

*// controlled comp, force value w/state, this.state = {}*

```
return (<div><input name="first" value={this.state.first}  
  onChange={this.firstChgd.bind(this)} /></div>);
```

...

```
firstChgd(e) { this.setState({ first: e.target.value }); }
```

# React.js API - refs

```
return (<form onSubmit={this.sendData.bind(this)}>  
  <input name="foo"/>  
  <button ref="submitButton">Send</button>  
</form>);
```

---

```
sendData(e) {  
  e.preventDefault();  
  var submitButton = React.findDOMNode(this.refs.submitButton);  
  submitButton.disabled = true; // re-enable after post completes  
  // send data, then submitButton.disabled = false;  
}
```

# React.js API - Lifecycle

## React Component Lifecycle

### Mount

`componentWillMount()`

`componentDidMount()`

### Updates

`componentWillReceiveProps(nextProps)`

`shouldComponentUpdate(nextProps, nextState)`

`componentWillUpdate(nextProps, nextState)`

`componentDidUpdate(prevProps, prevState)`

### Unmount

`componentWillUnmount()`



# Survey Part 2

- Display poll as form
- Post { YOUR\_ID: IDX } to /polls/\${id}
- that returns poll result, display poll result
- Use PieChart from react-d3-components to display

# Survey Part 3

- Use state to display first poll
- Use state to hold shuffled polls
- Add next survey button which shifts poll
- `setState` trigger update to show new poll

# Survey Part 4

- Add admin.html
- Add conditional logic `location.pathname = '/admin.html'`
  - `fetch /admin/polls, display all`
- Add question form
  - `post { question: 'What', choices: [ 'Foo', 'Bar'] }`  
to `/polls`

# React.js API - prop validation

```
MyComponent.propTypes = {  
  foo: React.PropTypes.object, // any object  
  bar: React.PropTypes.shape({ f: React.PropTypes.string }),  
  baz: React.PropTypes.array, // also arrayOf(propType...)  
  cat: React.PropTypes.func.isRequired, // fn + required  
  dog: React.PropTypes.node, // number, string, array, element  
  egg: React.PropTypes.any.isRequired, // anything + required  
  fig: React.PropTypes.instanceOf(Message),  
  gib: React.PropTypes.oneOf(['optionA', 'optionB']), // enum  
  hib: function (props, propName, compName) { // custom  
    if (...) { return new Error('my validation error'); }  
  }  
};
```

---

# React.js API - default props

*// cached and used as the defaults*

```
MyComponent.defaultProps = {  
  foo: 'default value',  
  bar: 1  
};
```

---

# React.js API - perf tuning

```
return (  
  <ul>  
    { items.map(i => <li key={i.id}>{i.name}</li>) }  
  </ul>  
);
```

---

```
import PureComponent from 'react-pure-render/component';  
class Foo extends PureComponent {  
  // implements shouldComponentUpdate with shallow compare  
  // works for primitives and immutable objects  
}
```

# React.js API - render string

*// can render to string on server*

```
const str = React.renderToString(<App items={myItems} />);
```

*// alternatively if not using React in browser,*

*// renderToStaticMarkup also renders to string, but*

*// doesn't include the React attributes (id's and checksums)*

*// needed for reconciliation*

```
const str = React.renderToStaticMarkup(<App items={myItems} />);
```

---

# React.js family

- react-router
- Flux
- React Native



# react-router

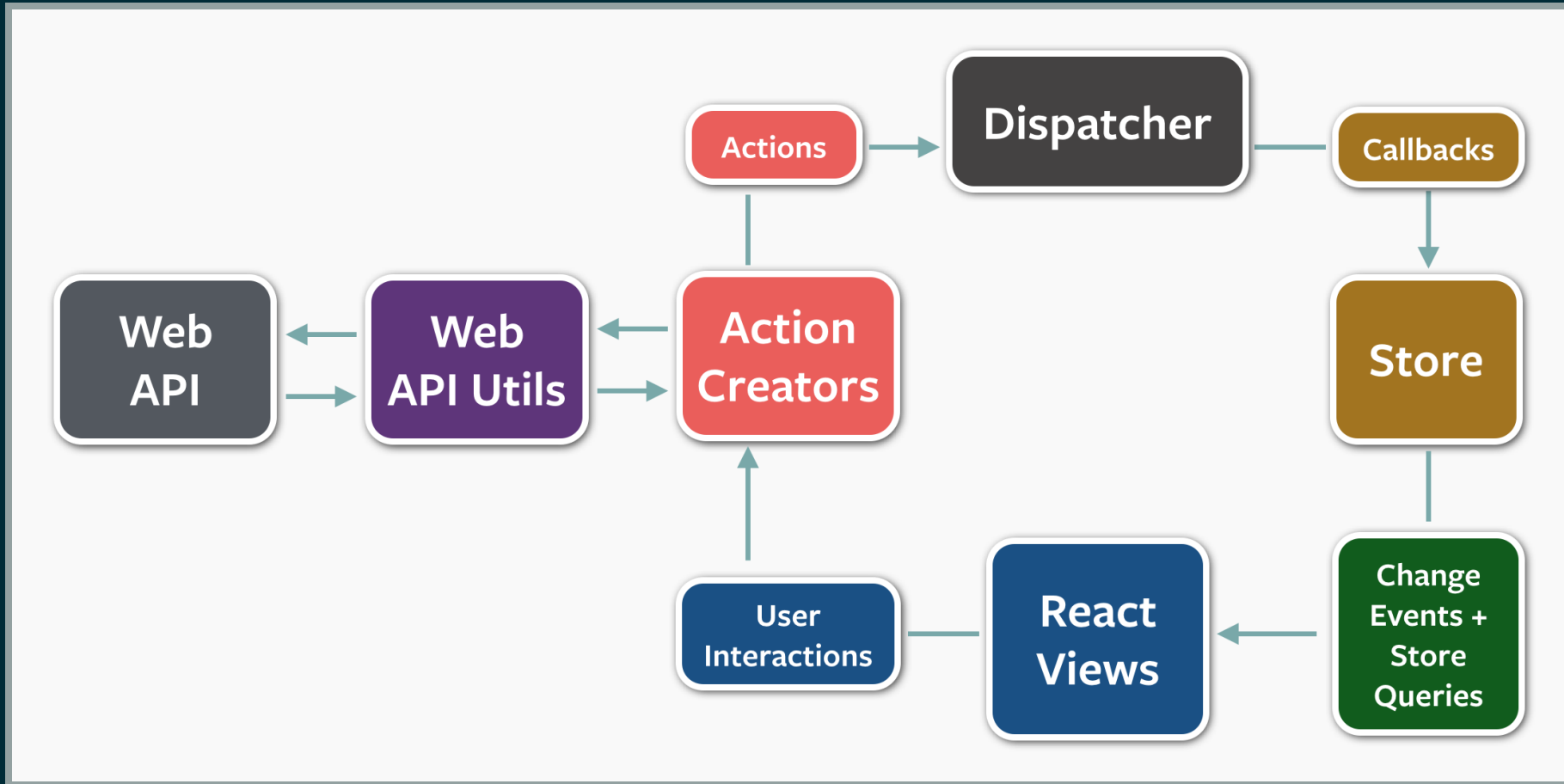
```
const routes = (<Route handler={App} path="/">
  <DefaultRoute handler={Home} />
  <Route name="about" handler={About} />
</Route>);

Router.run(routes, Router.HistoryLocation, (Handler, props) =>
  // can fetch data for props.routes here
  React.render(<Handler />, document.querySelector('#appDiv')));
});
```

---

```
return (<div><h1>App<h1> { /* in App render() */ }
  <RouteHandler />
</div>);
```

# Flux



# React Native

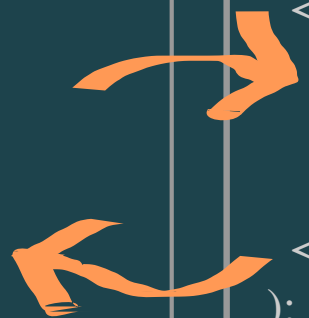
## React Native App

### Native IOS/Android

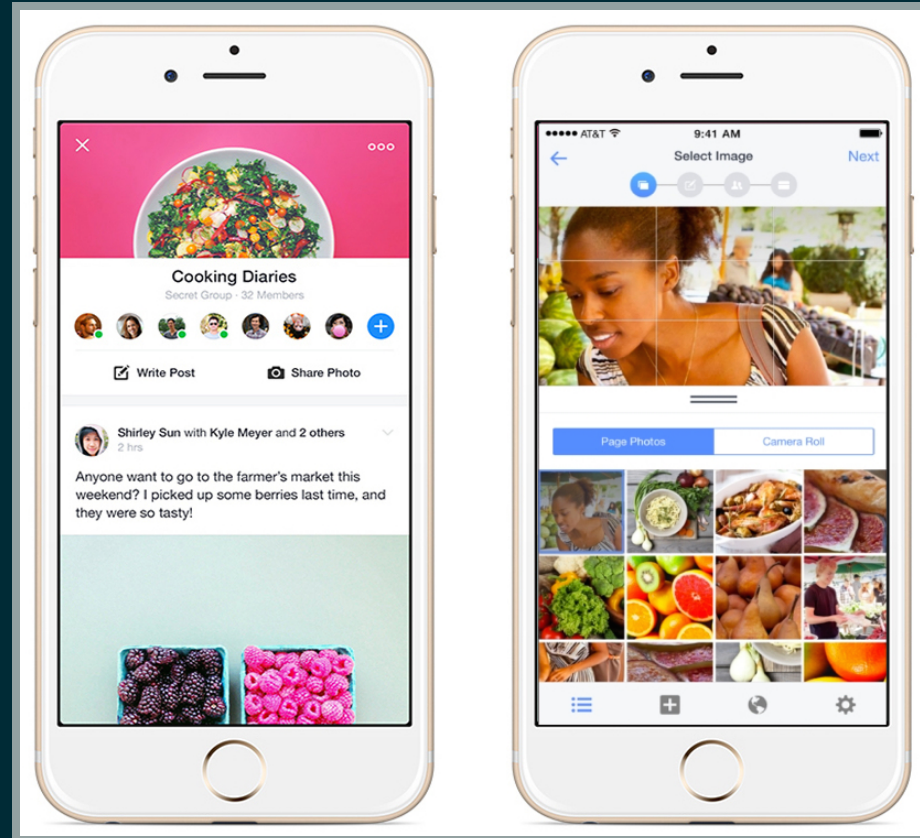
Image  
ListView  
MapView  
Navigator  
ScrollView  
Text  
TextInput  
View  
WebView  
...

### JS using React Native

```
render: function() {  
  return (  
    <View style={styles.container}>  
      <Text>{movie.title}</Text>  
      <Text>{movie.year}</Text>  
      <Image source={{uri: movie.img}} />  
    </View>  
  );  
}
```



# React Native p2



# Future

## Relay/GraphQL

```
{  
  user(id: 3500401) {  
    id,  
    name,  
    isViewerFriend,  
    profilePicture(size: 50) {  
      uri,  
      width,  
      height
```

## FalcorJS



# Summary

- React.js is a game changer
- Use it in your projects today
- Enjoy coding in a new way!
- <http://codewinds.com/jsconf>
  - React.js video training - summer 2015
- @jeffbbski jeff@codewinds.com