

Introduction to React

goo.gl/9bbmhN

Rose Wang
Justin Kenel
Aboli Kumthekar

Agenda

What is React?

Setup

Let's go!

What is React - Intro

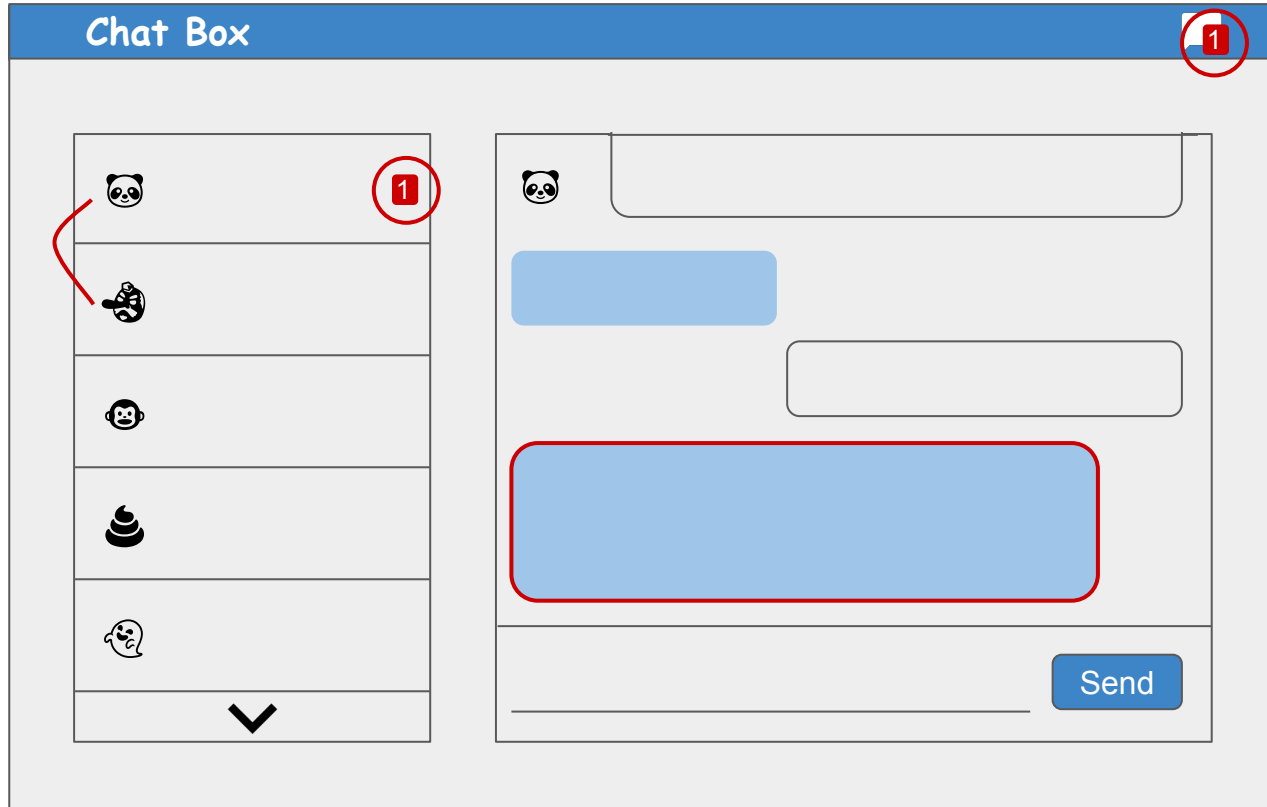
React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

- [ReactJS](#)

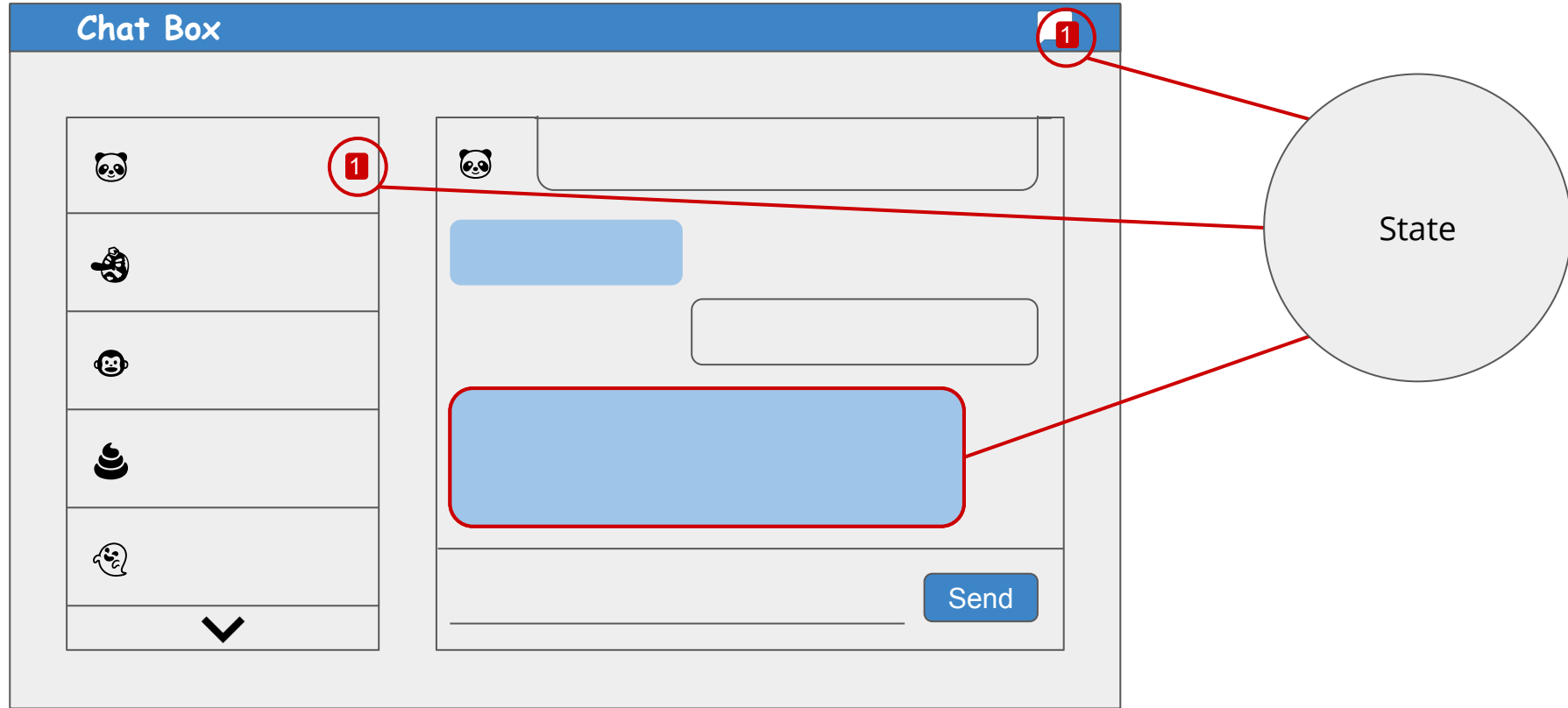
What is React - Components and the DOM



What is React - Updating the UI



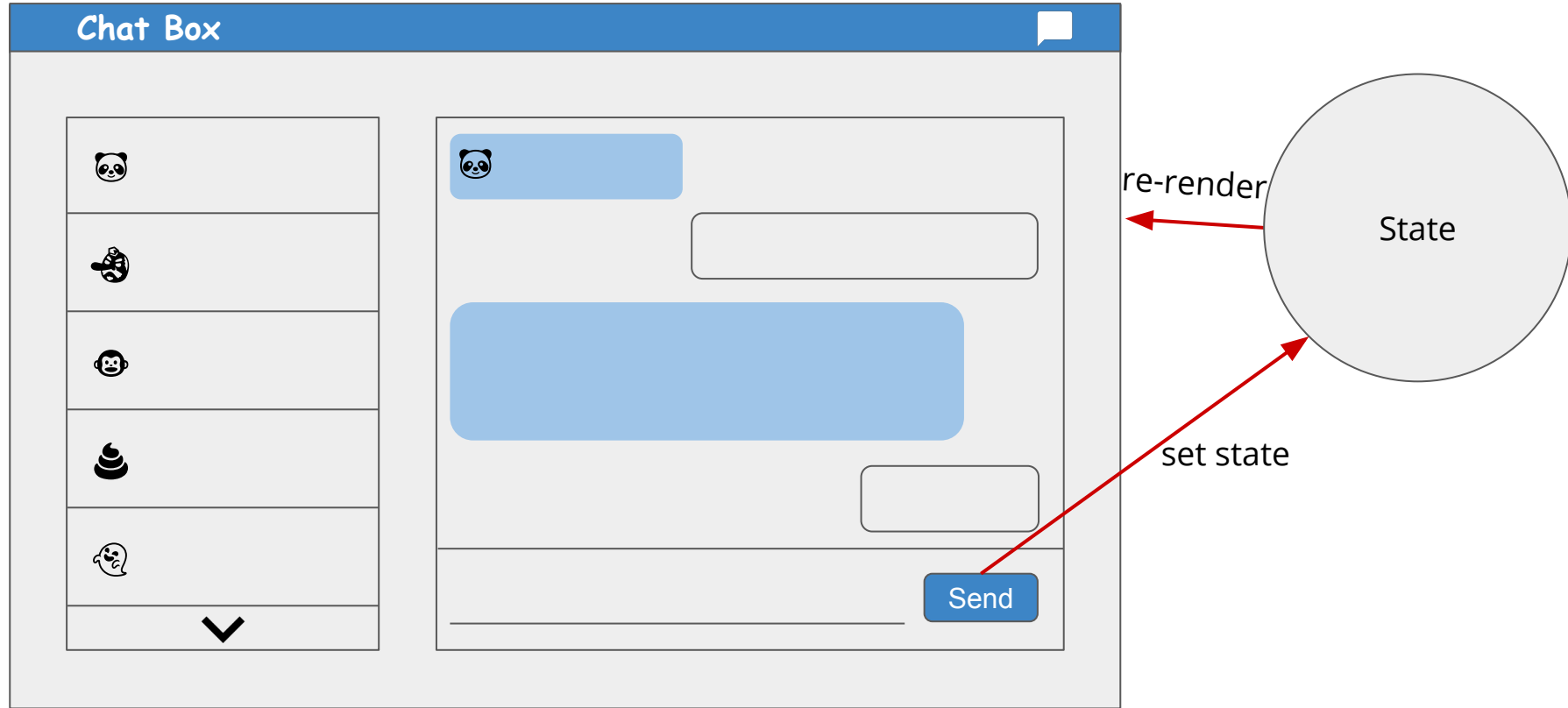
What is React - Centralized Logic for the View



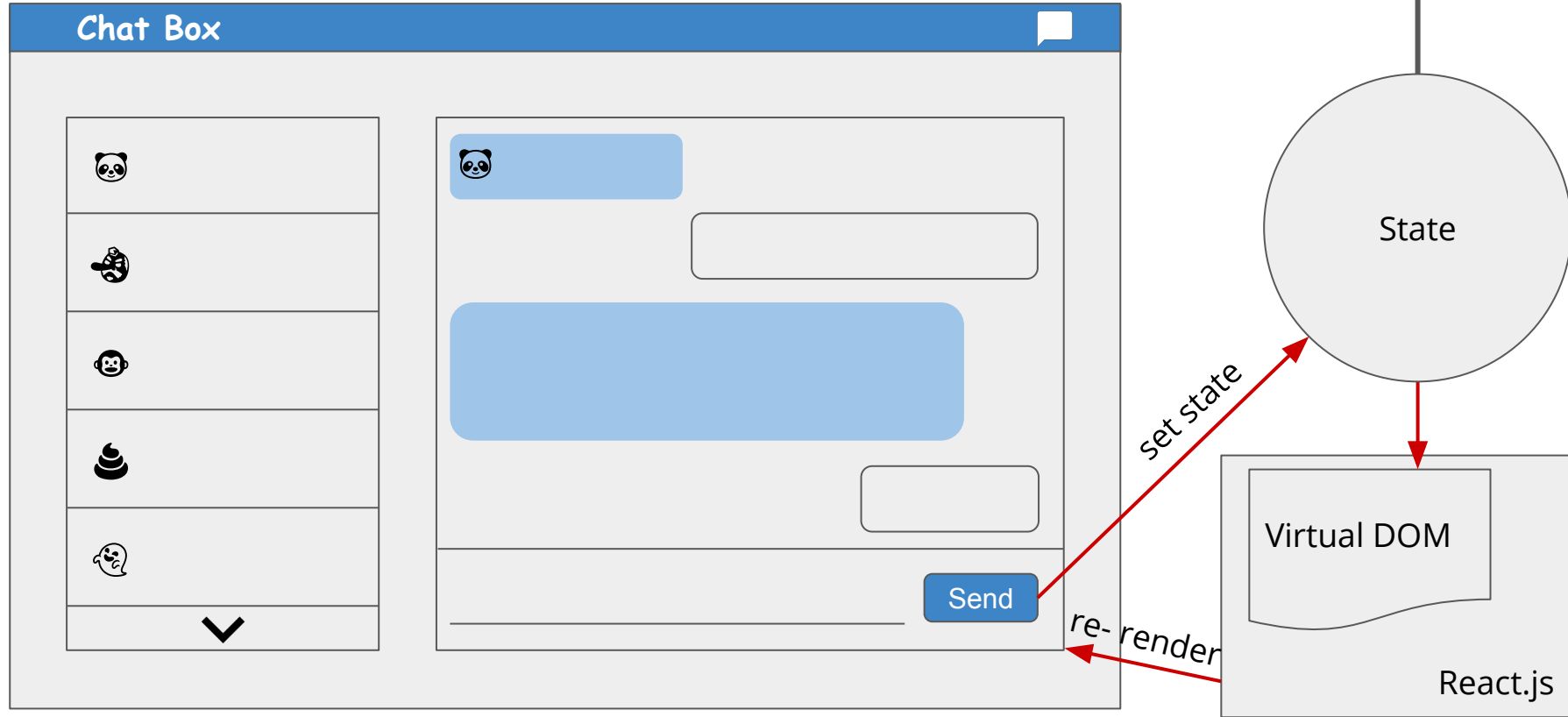
What is React - Rebuilding the UI



What is React - Re-Rendering



What is React - Re-Rendering



Setup

In case of wifi connectivity issues,
LAN credentials:
Router ssid: appian-hackathon
Router pass: appian2016
10.0.1.10:8000

1. Download and install Node.js (version 4.5):

<https://nodejs.org/en/download/>

Note: Windows users may use the Node.js command prompt

2. Download and install the latest version of [Sublime](https://www.sublimetext.com/) (or preferred editor):

<https://www.sublimetext.com/>

3. Download and unzip the [starter project](http://bit.ly/2fC4Bsw):

<http://bit.ly/2fC4Bsw>

4. Open Sublime or IDE of choice

- a. Install "[Package Control](#)" and "Babel"
 - i. CTRL/CMD + SHIFT + P and search for the package
- b. Change the syntax highlighting to Javascript (Babel)
 - i. View > Syntax > Babel > Javascript (Babel)
- c. Go to File > Open... > select the starter project folder

Setup - Dependencies and Builds



npm (Node.js Package Manager) - takes care of getting your project dependencies

`package.json` - tells npm which dependencies your project needs (also defines build targets)



Babel - a library for compiling jsx into js that will work on any browser.



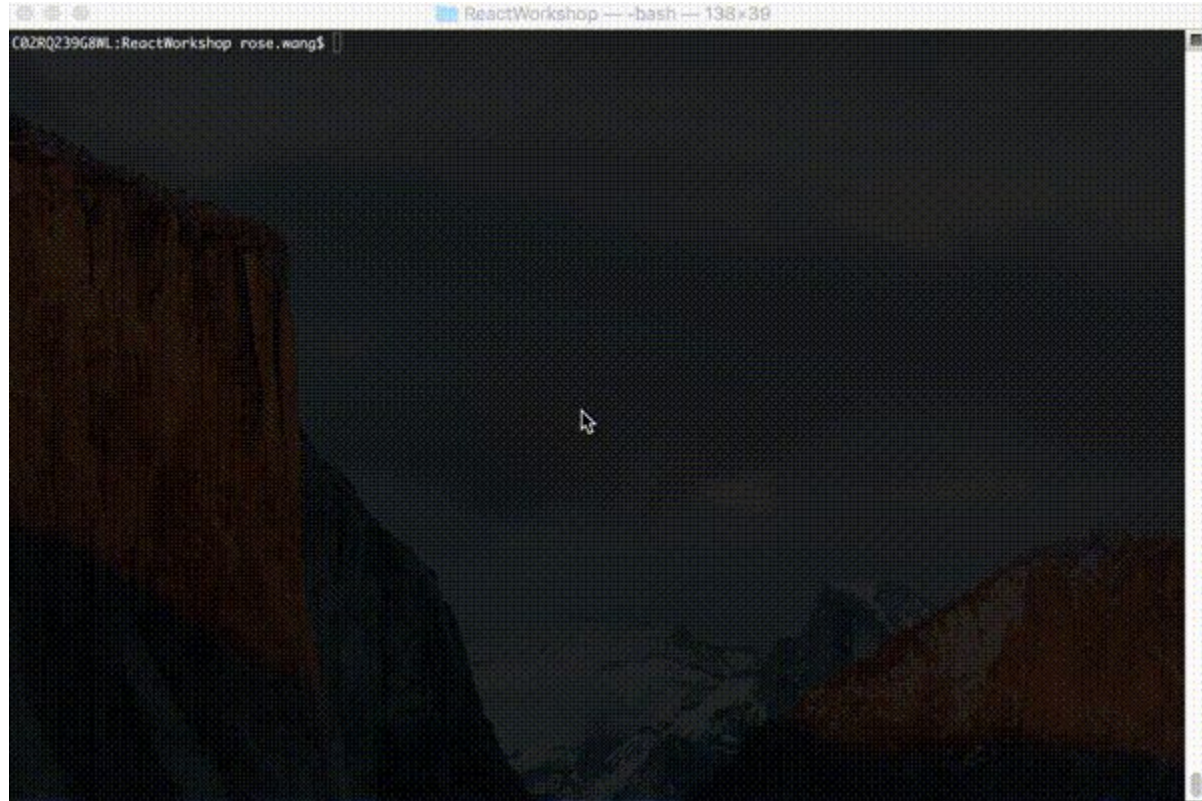
Webpack - A JS library that builds your project, and hosts it (locally)

- `webpack.config.js` - Tells Webpack how to build your project

Dependencies and Builds

Run `npm install` in the command line from the root of your project

This will download all the dependencies that are in your `package.json` file.



Setup - Key Files to Success

`package.json`: tells npm which dependencies your project needs (also defines build targets)

`webpack.config.js`: Tells Webpack how to build your project

`main.js`: This will be compiled into `index.js` by webpack (we defined it that way in `webpack.config.js`). It will load our React code.

`index.html`: The first page of your application

`App.jsx`: Component for our UI

App.jsx

```
// Import React and other stuff  
import React ....
```

```
const App = React.createClass({  
  getInitialState() {  
    // Initialize empty list of messages  
  },
```

```
  componentWillMount() {  
    // Turns on Firebase  
  },
```

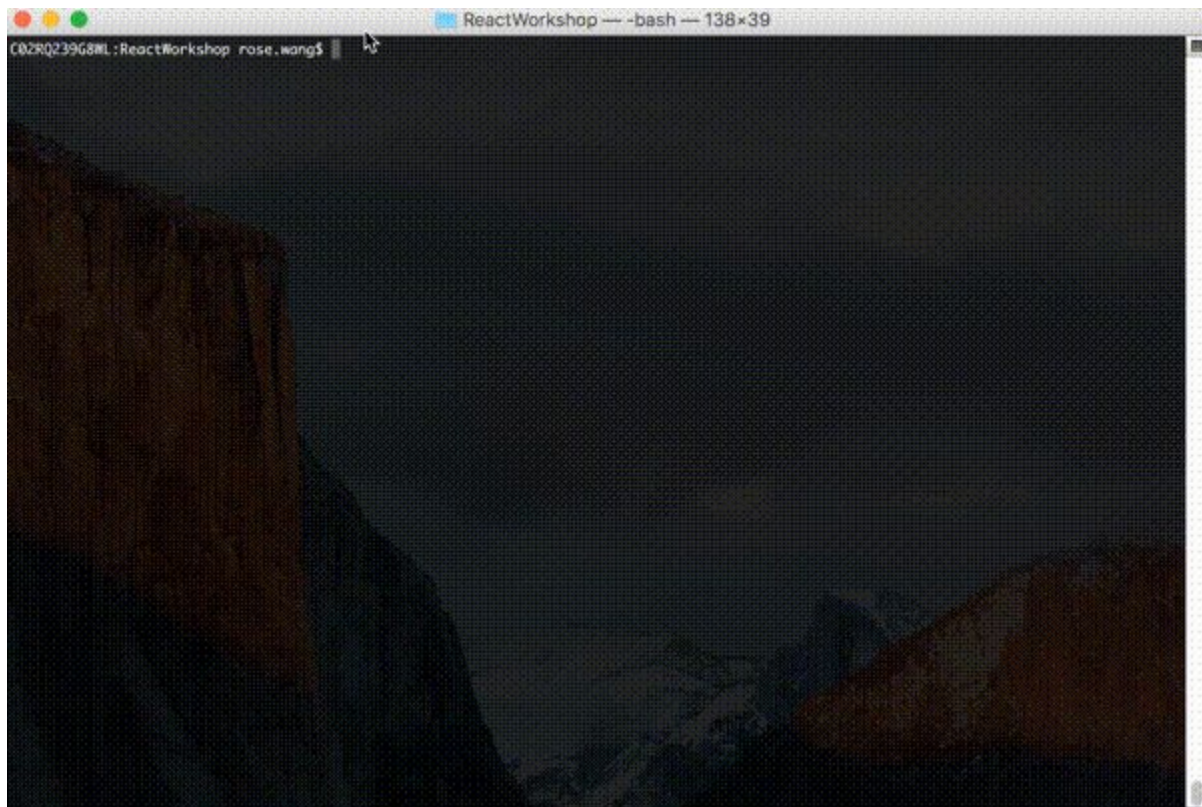
```
  componentWillUnmount() {  
    // Turns off Firebase  
  },
```

```
  render() {  
    // shows your content on the screen  
  }  
});
```

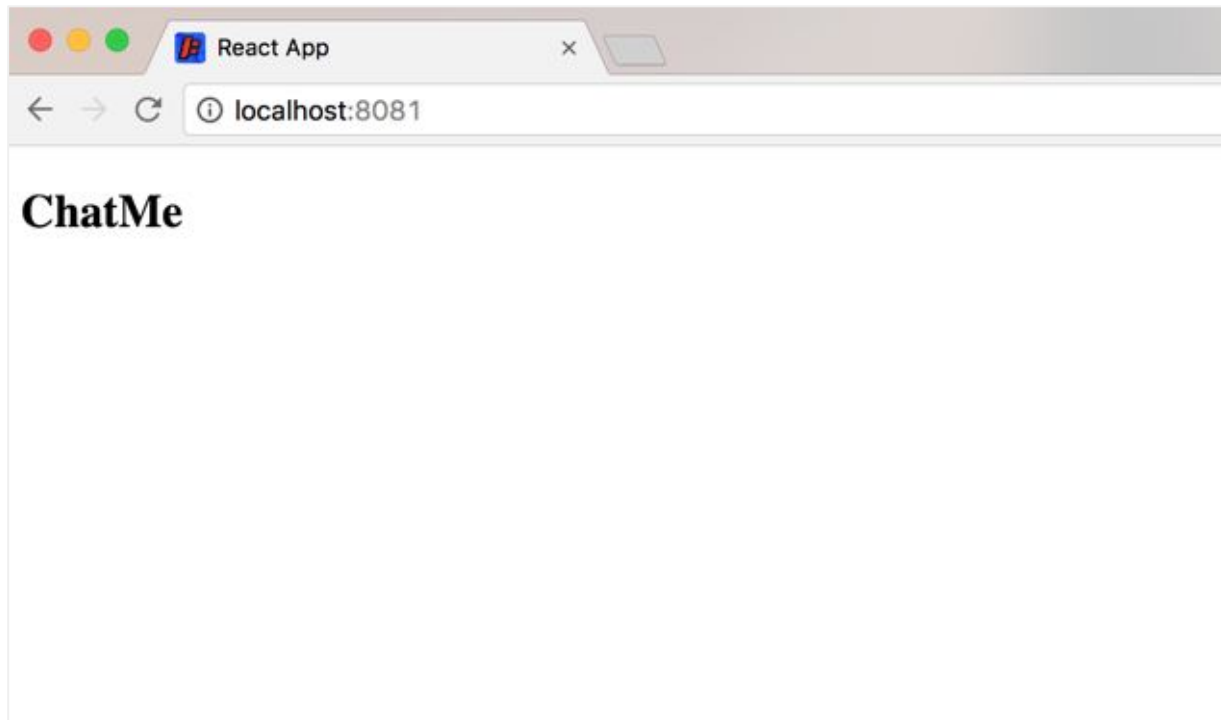
```
export default App;
```

Let's run our app!

1. Once `npm install` has completed, we can run `npm run start` from the command line. This will deploy our project.
2. Go to `localhost:8081` in your browser to see your app!

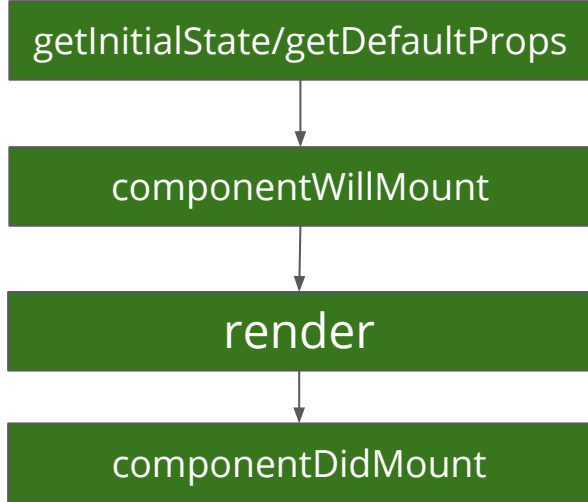


Let's run our app!

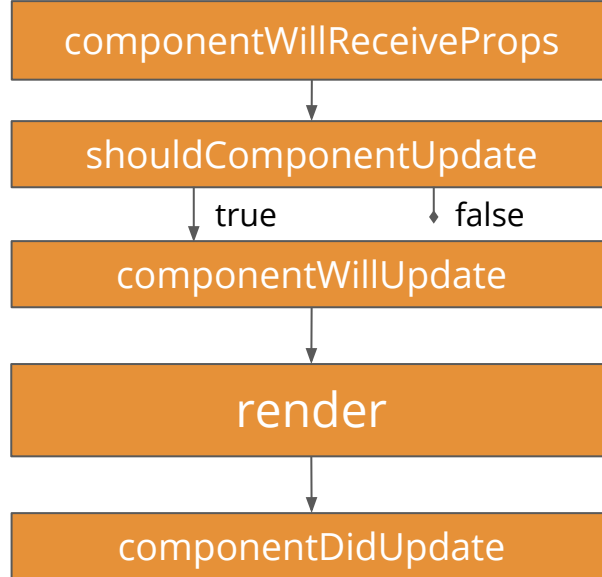


React Component Lifecycle

Mounting (Creating)



Updating

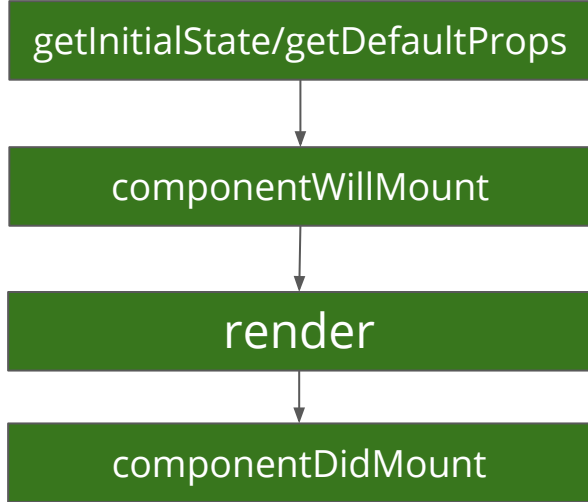


Unmounting (Destroying)

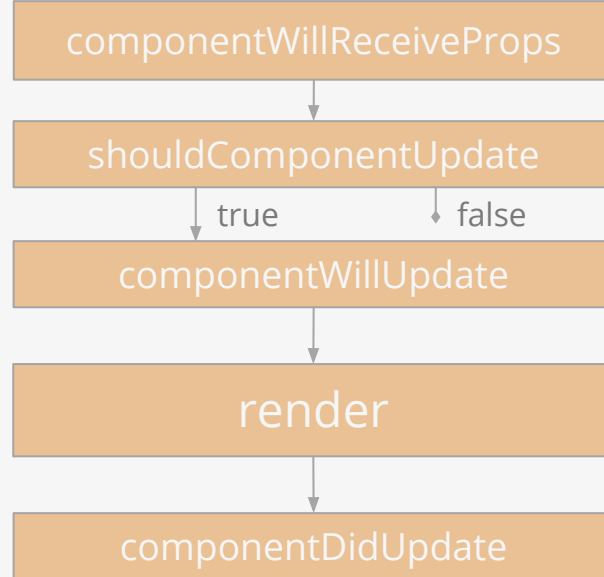


React Component Lifecycle: Mounting

Mounting (Creating)



Updating



Unmounting (Destroying)



App.jsx

Mounting

```
// Import React and other stuff  
import React ....
```

```
const App = React.createClass({  
  getInitialState() {
```

```
    return {  
      messages: [] // Initialize empty list of messages  
    };  
  },  
  
  componentWillMount() {  
    // Turns on Firebase  
  },
```

```
  componentWillUnmount() {  
    // Turns off Firebase  
  },
```

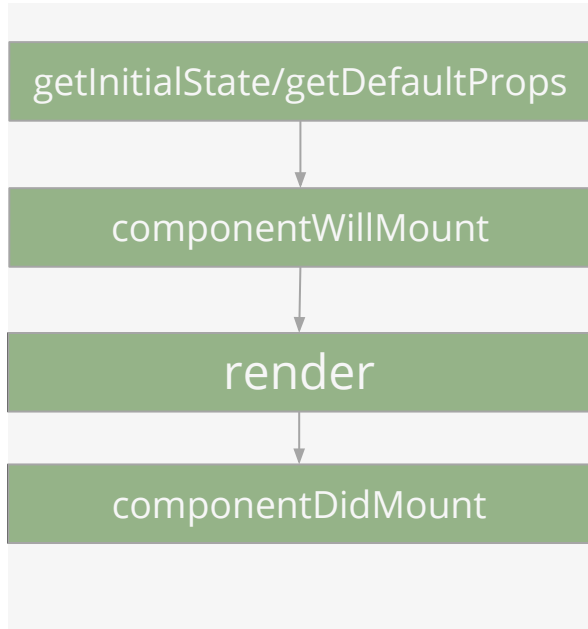
```
  render() {  
    // shows your content on the screen  
  }
```

```
});
```

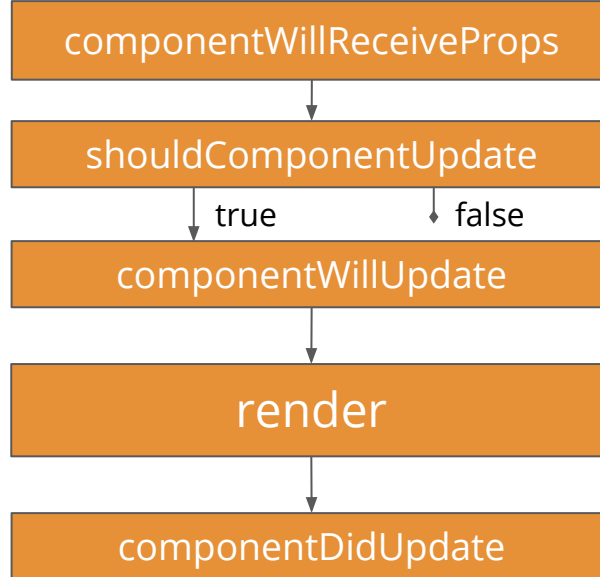
```
export default App;
```

React Component Lifecycle: Updating

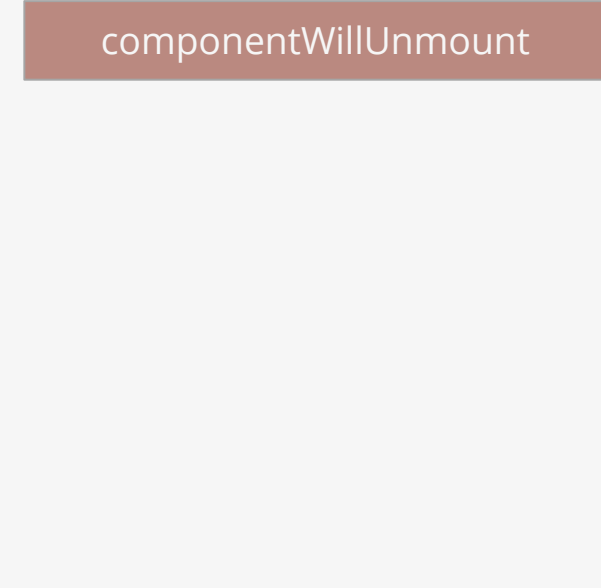
Mounting (Creating)



Updating



Unmounting (Destroying)



App.jsx

Updating

```
// Import React and other stuff  
import React ....
```

```
const App = React.createClass({  
  getInitialState() {  
    return {  
      messages: [] // Initialize empty list of messages  
    };  
  },
```

```
  componentWillMount() {  
    // Turns on Firebase  
  },
```

```
  componentWillUnmount() {  
    // Turns off Firebase  
  },
```

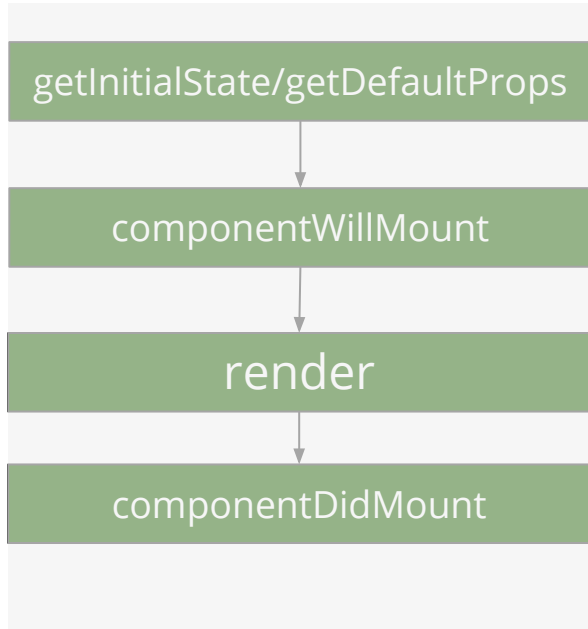
```
  render() {  
    // shows your content on the screen  
  }
```

```
});
```

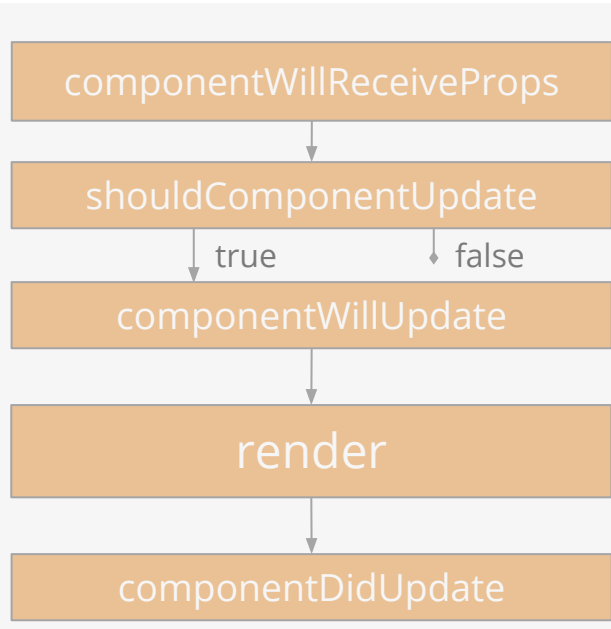
```
export default App;
```

React Component Lifecycle: Unmounting

Mounting (Creating)



Updating



Unmounting (Destroying)



App.jsx

Unmounting

```
// Import React and other stuff  
import React ....
```

```
const App = React.createClass({  
  getInitialState() {  
    return {  
      messages: [] // Initialize empty list of messages  
    };  
  },
```

```
  componentWillMount() {  
    // Turns on Firebase  
  },
```

```
  componentWillUnmount() {  
    // Turns off Firebase  
  },
```

```
  render() {  
    // shows your content on the screen  
  }  
});
```

```
export default App;
```

Can We Do Something Now?

Sure, we are going to use Firebase to host the messages for our chat application.

First, let's display the pre-recorded messages from our database

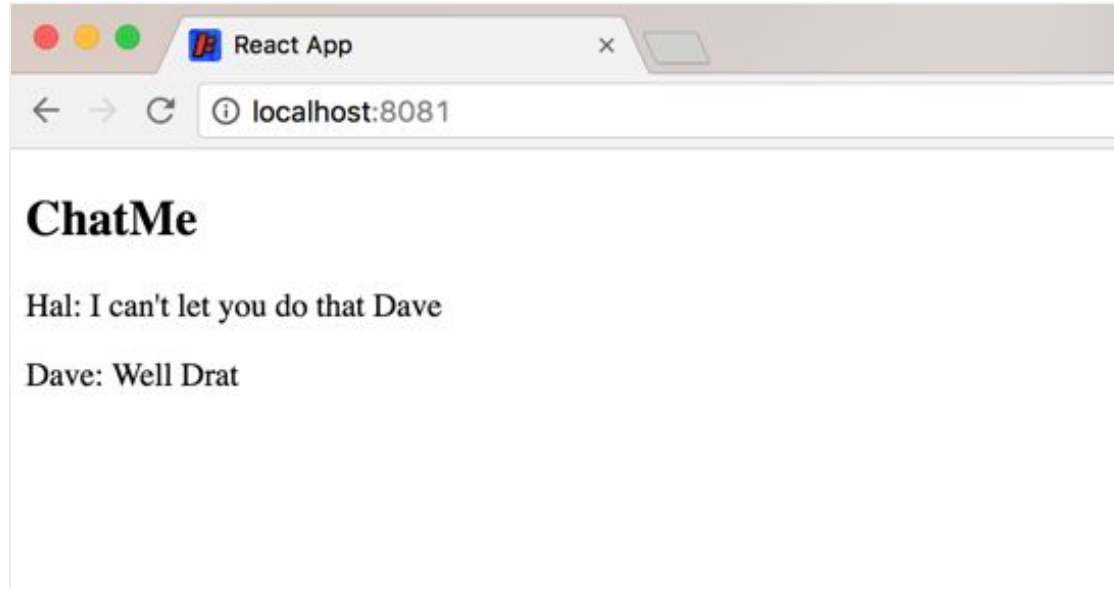
[See the changes here!](#) (or on the next slide)

Can We Do Something Now?

```
5 App.jsx View

@@ -51,7 +51,7 @@ const App = React.createClass({
  },
  renderMessageDiv(message) {
-   return <p />;
+   return <p key={message.key}>{message.name}: {message.message}</p>;
  },
  render() {
@@ -62,6 +62,9 @@ const App = React.createClass({
    return (
      <div>
        <h2>ChatMe</h2>
+       <div>
+         {messageDivs}
+       </div>
      </div>
    );
  }
}
```

Can We Do Something Now?



Common React/JS mistakes

- Syntax errors, like missing `,` `.` `;` `{` or `(`
- Mixing up `,` with `.` OR `(` with `{`
- Misspelling or mis-casing variable names
- Missing keywords like `var`, `const`, `let` or `import`

PRO-TIP: Run `npm run lint` in your directory to catch syntax errors!

Making A Component

You can make React components to modularize the UI

We are going to make each message a component

[See the changes here!](#)

Making A Component

3 ■■■ App.jsx View

```
...    ...    @@ -1,5 +1,6 @@
1      1      import React from 'react';
2      2      import * as firebase from 'firebase'; // Import Firebase library
3      3      +import Message from './components/Message.jsx';
3      4
```

53 54 renderMessageDiv(message) {
54 - return <p key={message.key}>{message.name}: {message.message}</p>;
55 + return <Message key={message.key} message={message}>/>;
55 56 },

5 ■■■■■ components/Message.jsx View

```
✱    @@ -7,6 +7,11 @@ const Message = React.createClass({
7      7      },
8      8
9      9      render() {
10     + // Destructuring pulls the different fields from
11     + // the message object
12     + const {name, message} = this.props.message;
13     +
14     + return <p>{name}: {message}</p>;
```

Destructuring Assignments

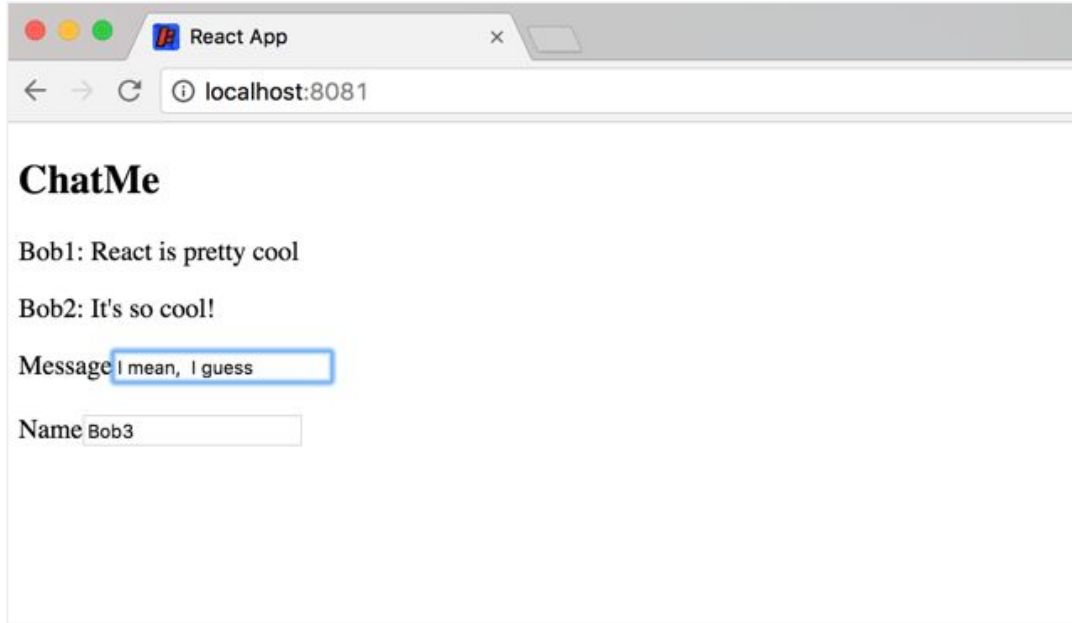
```
const {name, message} = this.props.message;
```

is the same as:

```
const name = this.props.message.name;
```

```
const message = this.props.message.message;
```

Adding new Messages



A screenshot of a web browser window titled "React App" with a close button. The address bar shows "localhost:8081". The page content is titled "ChatMe". It displays two messages: "Bob1: React is pretty cool" and "Bob2: It's so cool!". Below these is a form with a label "Message" and a text input field containing "I mean, I guess". At the bottom is a form with a label "Name" and a text input field containing "Bob3".

ChatMe

Bob1: React is pretty cool

Bob2: It's so cool!

Message

Name

Adding new Messages

For both **name** and **message** we need an both an **input** and a **label**

Our message will be sent to the server when we hit Enter, then the message field will be cleared.

[See the changes here!](#)

Bonus: Play around with the events to make the message send with different keys

Adding new Messages

34 ■■■■ App.jsx View

```
...    ...    @@ -1,6 +1,7 @@  
1      1      import React from 'react';  
2      2      import * as firebase from 'firebase'; // Import Firebase library  
3      3      import Message from './components/Message.jsx';  
4      4      +import Input from './components/Input.jsx';  
...    ...  
5      6      const App = React.createClass(  
6      7          getInitialState() {  
7      8              return {  
8      9                  - messages: [] // Initialize empty list of messages  
9      10                 + messages: [], // Initialize empty list of messages  
10     11                 + name: 'Bob',  
11     12                 + newMessage: ''  
12     13             };  
13     14         },  
...    ...
```

Adding new Messages

34 App.jsx

View

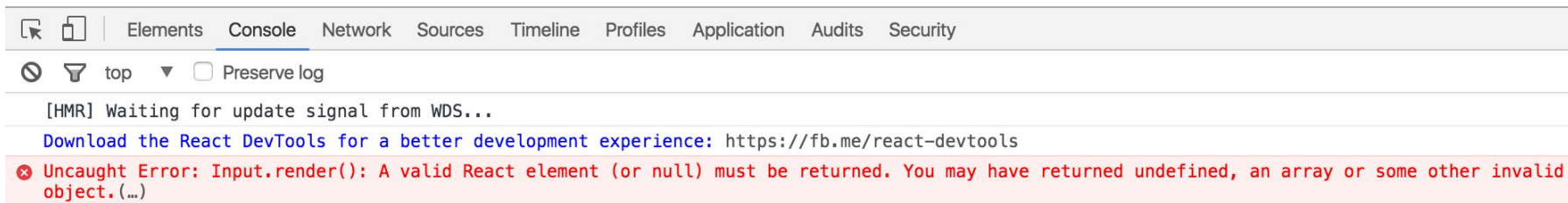
```
25 + handleChange(event) {
26 +   this.setState({name: event.target.value});
27 + },
28 +
29 + handleMessageChange(event) {
30 +   this.setState({newMessage: event.target.value});
31 + },
32 +
33 + handleKeyPress(event) {
34 +   const {name, newMessage} = this.state;
35 +   // If name or newMessage are blank, do not send new message
36 +   if (!name || !newMessage) {
37 +     return;
38 +   }
39 +
40 +   // If user hits Enter key, then send message to Firebase database
41 +   // and clear out the message box
42 +   if (event.key === 'Enter') {
43 +     Firebase.sendMessage({ name: name, message: newMessage });
44 +     this.setState({newMessage: ''});
45 +   }
46 + },
```

Adding new Messages

34 ■■■■ App.jsx

View

```
54 // for each message
55 const messageDivs = this.state.messages.map(this.renderMessageDiv);
56
57 + const {newMessage, name} = this.state;
58 +
59 return (
60   <div>
61     <h2>ChatMe</h2>
62     <div>
63       {messageDivs}
64     </div>
65 +   <div>
66 +     <Input label={'Message'} value={newMessage} onChange={this.handleMessageChange} onPress={this.handleKeyPress} />
67 +     <Input label={'Name'} value={name} onChange={this.handleNameChange} />
68 +   </div>
69 </div>
70 );
71 }
```



...because we're not done!

Adding new Messages

```
15 components/Input.jsx View
@@ -9,6 +9,21 @@ const Input = React.createClass({
 9     },
10
11     render() {
12 +     const {label, value, onChange, onKeyPress} = this.props;
13 +     // We can now pass in event handlers to allow users to
14 +     // interact with this component
15 +     return (
16 +       <div>
17 +         <label htmlFor={label}>{label}</label>
18 +         <input
19 +           id={label}
20 +           type="text"
21 +           value={value}
22 +           onChange={onChange}
23 +           onKeyPress={onKeyPress}
24 +         />
25 +       </div>
26 +     );
  }
```

Styling our Interface

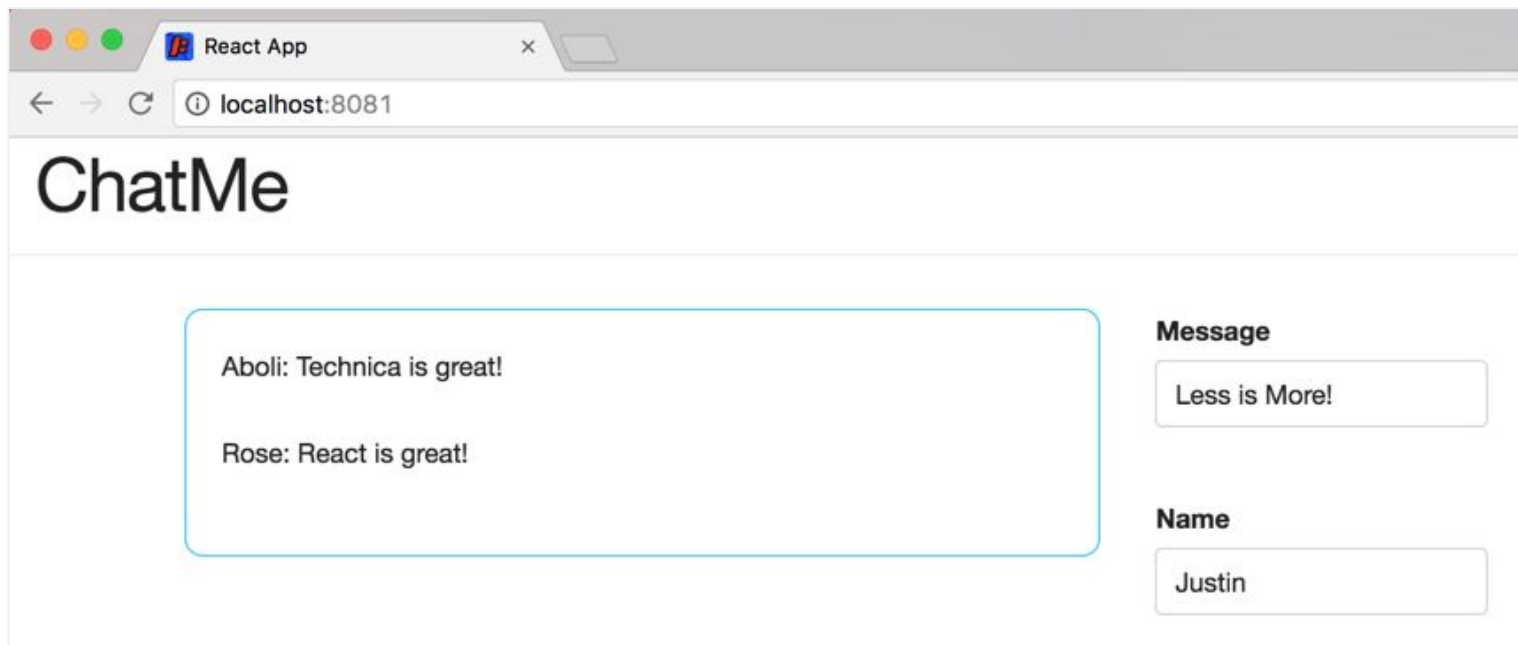
Our stylesheet is already included in the `styles/` directory

Let's import it and update our components to use the styles.

[See the changes here!](#)

Bonus: Change the class names on the different divs and see how the styles change

Styling our Interface



Styling our Interface

23  App.jsx View

		@@ -2,6 +2,7 @@ import React from 'react';
2	2	import * as firebase from 'firebase'; // Import Firebase library
3	3	import Message from './components/Message.jsx';
4	4	import Input from './components/Input.jsx';
	5	+import style from './styles/App.less';
5	6	

Styling our Interface

✱		@@ -90,13 +91,21 @@ const App = React.createClass({
90	91	
91	92	return (
92	93	<div>
93		- <h2>ChatMe</h2>
94		- <div>
95		- {messageDivs}
96		- </div>
97		- <div>
98		- <Input label='Message' value={newMessage} onChange={this.handleMessageChange} onPress={this.handleKeyPress}
99		- <Input label='Name' value={name} onChange={this.handleNameChange} />
	94	+ <nav className="navbar">
	95	+ <div className="container">
	96	+ <h2>ChatMe</h2>
	97	+ </div>
	98	+ </nav>
	99	+ <div className="container">
	100	+ <div className="eight columns messages">
	101	+ <div className="scrollview">
	102	+ {messageDivs}
	103	+ </div>
	104	+ </div>
	105	+ <div className="four columns">
	106	+ <Input label='Message' value={newMessage} onChange={this.handleMessageChange} onPress={this.handleKeyPress}
	107	+ <Input label='Name' value={name} onChange={this.handleNameChange} />
	108	+ </div>
100	109	</div>

Adding Emoji Support

React App

localhost:8081

ChatMe

Penguin: 🐧🐧

Agent Hops: 🐰

Nick: :fox:

Message

Name

Adding Emoji Support

Go to you message component and add the library.

[See the changes here!](#)

Try: "thumbsup:", "panda_face:", "money_with_wings:", "alien:", or "speedboat:"

[link to more emojis](#)

Adding Emoji Support

```
5 components/Message.jsx View
... @@ -1,4 +1,5 @@
1 1 import React from 'react';
2 2 +import * as emoji from 'node-emoji';
3 3
4 4 const Message = React.createClass({
5 5 // Setting propTypes ensure that your component is used correctly
6 6
7 7 @@ -11,7 +12,9 @@ const Message = React.createClass({
11 12 // the message object
12 13 const {name, message} = this.props.message;
13 14
14 14 - return <p>{name}: {message}</p>;
15 15 + const emojiString = emoji.emojiify(message);
16 16 +
17 17 + return <p>{name}: {emojiString}</p>;
18 18 }
19 19 });
20 20
```

More Common React/JS mistakes

- Missing imports
- Forgetting to install libraries before using them
- Not updating the state using `setState` on user interaction
- Confusing `=`, `==`, and `===`
- Accessing a variable, or even `this`, in the wrong scope
- Check out this [video](#)

Questions?