React.js

Paul Fodor

CSE316: Fundamentals of Software Development

Stony Brook University

http://www.cs.stonybrook.edu/~cse316

React

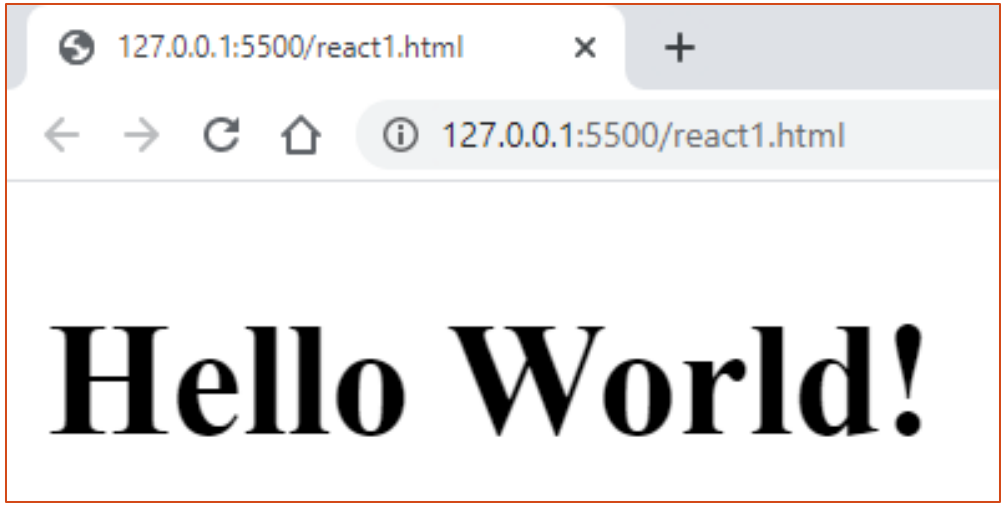
- React is a Javascript library for building user interfaces or UI components.
  - React creates a Virtual DOM in Javascript that mimics the browser DOM
  - Helps render web pages with consistent look and feel
  - It is maintained by Facebook and a community of individual developers and companies.
    - React was created by Jordan Walke, a software engineer at Facebook and deployed on Facebook's News Feed in 2011 and later on Instagram in 2012
    - Initial Public Release on 29 May 2013
    - It was open-sourced in March 2015

React Directly in HTML

- The quickest way start React is to write React directly in your HTML files.
- Start by including three scripts, the first two let us write React code in our JavaScripts, and the third, Babel, allows us to write JSX syntax

```
<script src="https://unpkg.com/react@16/umd/react.production.min.js"></script>
<script src="https://unpkg.com/react-dom@16/umd/react-dom.production.min.js"></script>
<script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>
```

```
<!DOCTYPE html>
<html>
 <script src="https://unpkg.com/react@16/umd/react.production.min.js"></script>
 <script src="https://unpkg.com/react-dom@16/umd/react-dom.production.min.js"></script>
 <script src="https://unpkg.com/babel-standalone@6.15.0/babel.min.js"></script>
 <body>

  <div id="mydiv"></div>

  <script type="text/babel">
   class Hello extends React.Component {
    render() {
     return <h1>Hello World!</h1>
    }
   }
   ReactDOM.render(<Hello />, document.getElementById('mydiv'))
  </script>

</body>
</html>
```



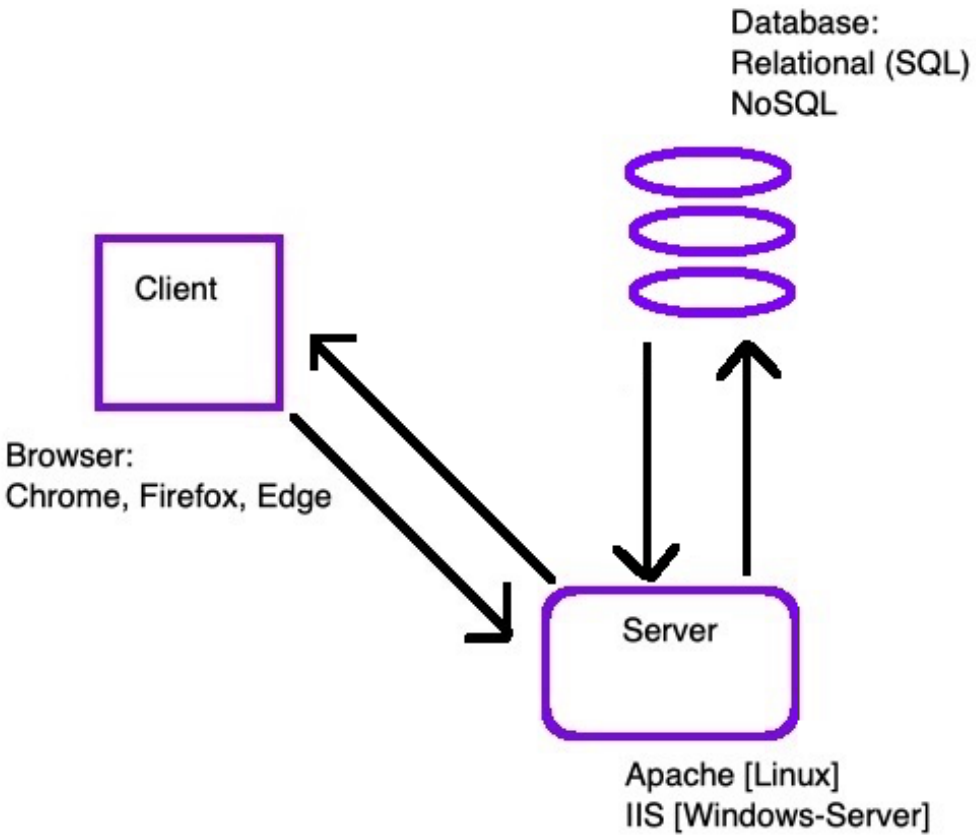
React

- React applications are composed of class components that:
  - Track state
  - Render page updates based on that state

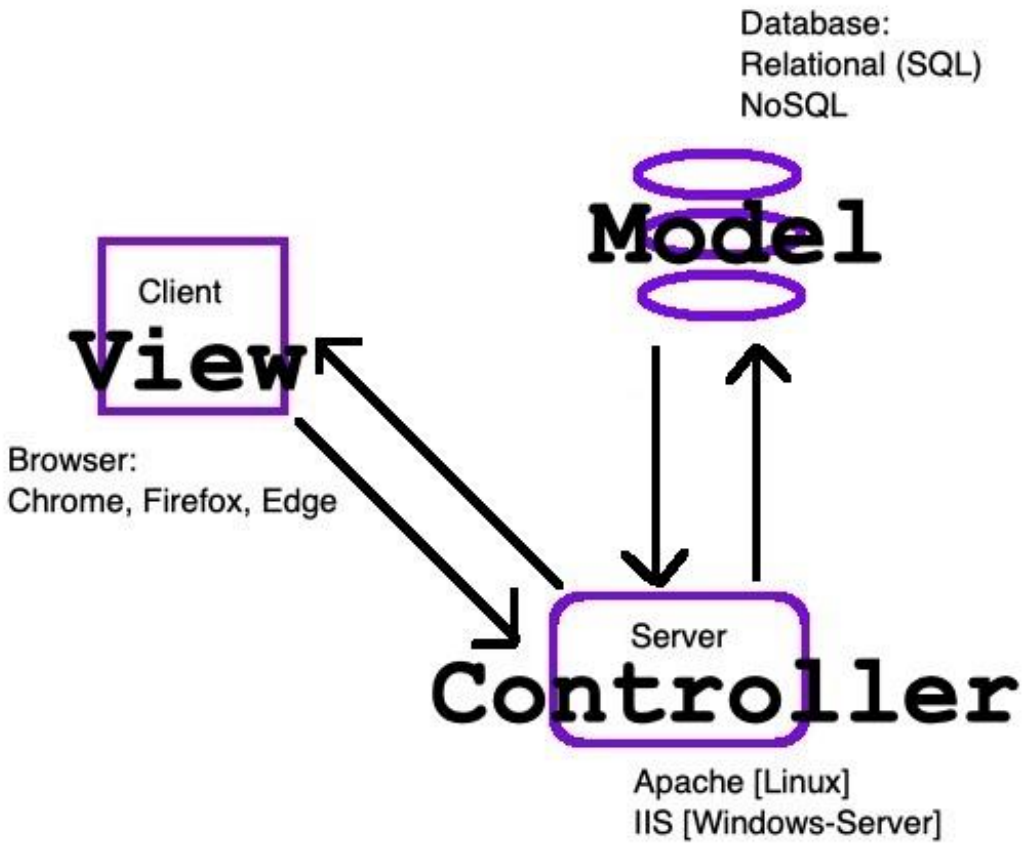
MVC

- At some point, Facebook described React as the V in MVC
- MVC is an architectual Design Pattern
- MVC is NOT a Framework (like Rails, CakePhp, Laravel, and django)
- Some web frameworks incorporate concepts of MVC

MVC


Database:
Relational (SQL)
NoSQL
Client
Browser:
Chrome, Firefox, Edge
Server
Apache [Linux]
IIS [Windows-Server]

MVC


Database:
Relational (SQL)
NoSQL
Model
Client
View
Browser:
Chrome, Firefox, Edge
Server
Controller
Apache [Linux]
IIS [Windows-Server]

MVC

- Code affects the structure or content of data => **Model**
- Code that processes data to or from DB or prior to view => **Controller**
- Code outputs visible images and structures on browser => **View**

React

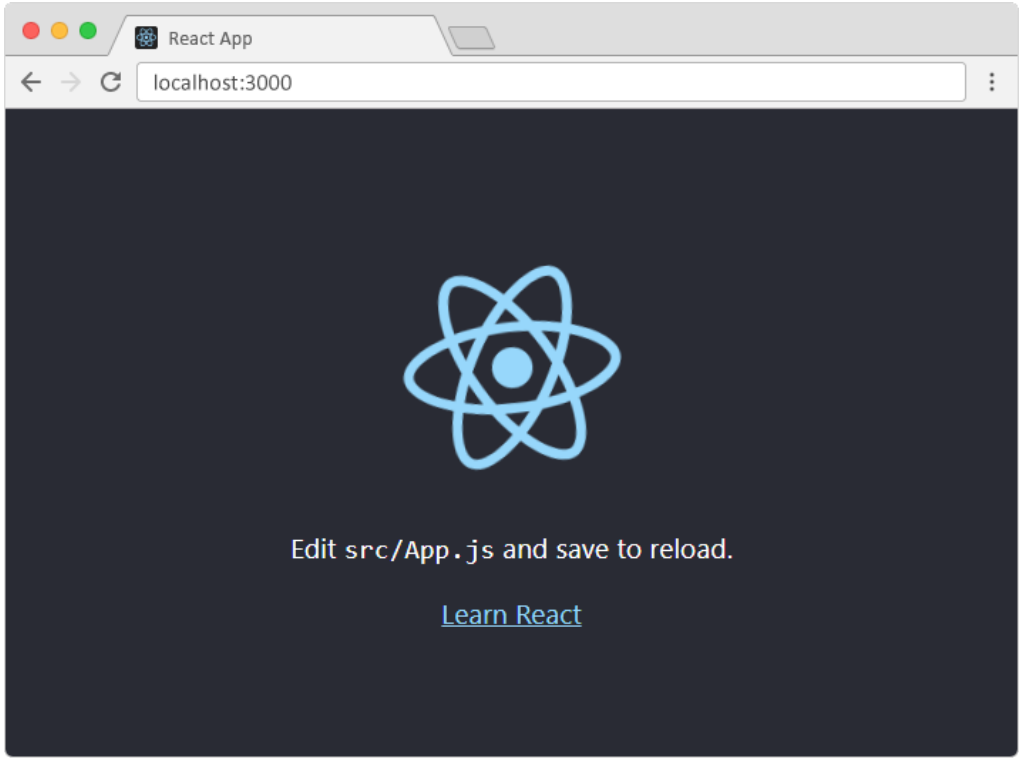
- In order to learn and use React, you should set up a React Environment on your computer.
  - The create-react-app is an officially supported way to create React applications.

npm install -g create-react-app

React

- The create-react-app will set up everything you need to run a React application.

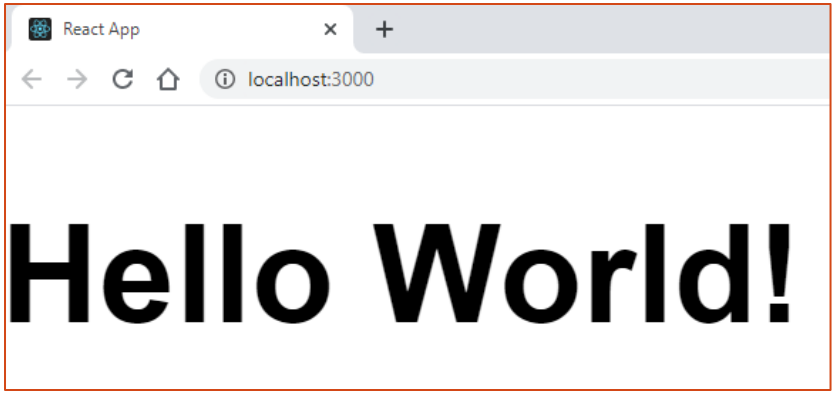
```
npx create-react-app myfirstreact
cd myfirstreact
npm start
```



React

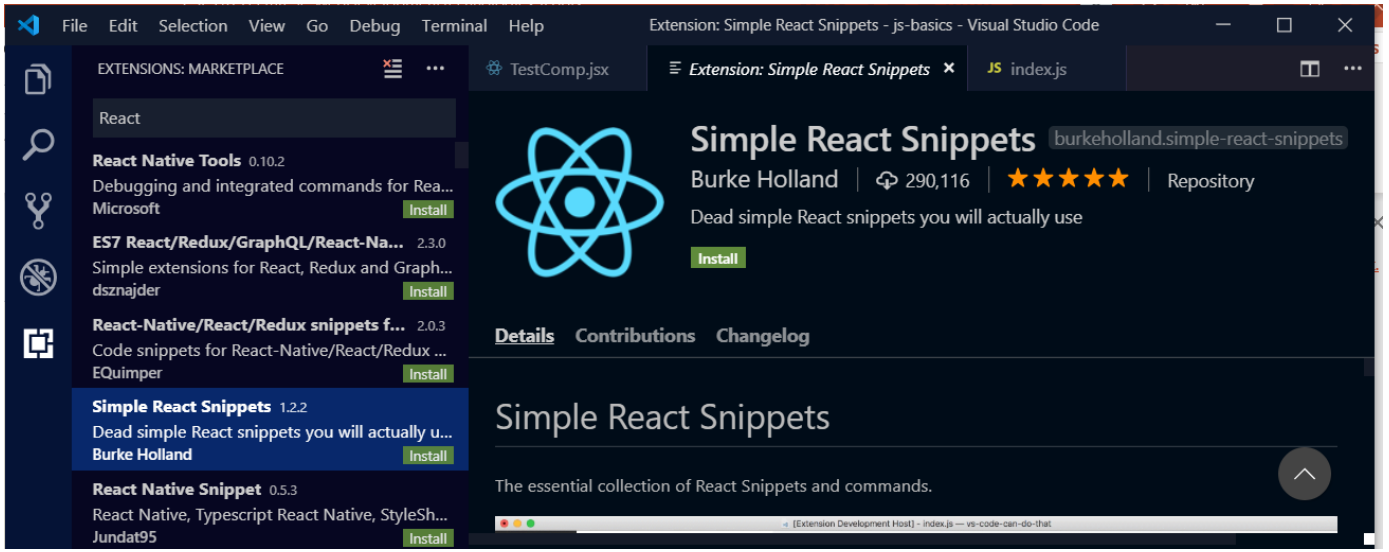
- Edit App.js:

```
import React from 'react';
import ReactDOM from 'react-dom';
class App extends React.Component {
  render() {
    return <h1>Hello World!</h1>;
  }
}
ReactDOM.render(<App />, document.getElementById('root'));
```



React

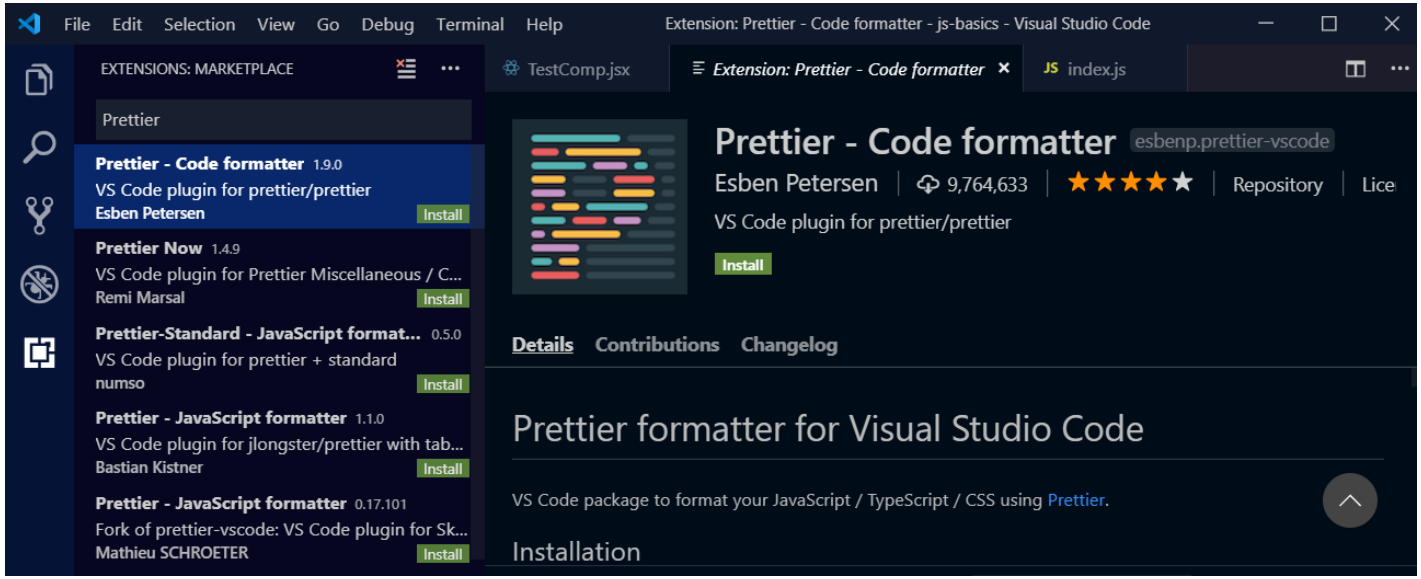
- Install Simple React Snippets in VSCode



Click 'Install'

React

- Install Prettier - Code Formatter



Click 'Install'
Exit and re-enter Visual Studio Code

React – First Application

- Install bootstrap – (a CSS library for consistent look and feel)

  npm i bootstrap
- Create a development folder
- Drop the development folder in Visual Studio Code
- Create a new application.
  - In terminal window:
    - Navigate to development folder created above and run:

      create-react-app myfirstreact

React – First Application

- Open Visual Studio code. Navigate/cd to folder <myfirstreact> inside your development folder
- App should have 3 folders
  - node_modules
  - public
  - src
- Open 'index.js' inside of the src folder and add a line to import boostrap

```
import 'bootstrap/dist/css/bootstrap.css'
```

React – First Component

- In src folder:
  - Create a folder called components
  - Create a .jsx file. Pick a name suggestive of its function
    - <componentname>.jsx
- Open the file. It will be empty
- Use Simple React Snippets to quickly write some template code
  - Select Simple React Snippets from Extensions menu
  - Type 'imrc<tab>' – This will generate import Component statement
  - Type 'cc<tab>' – This will create a class

React – First Component

```
import React, { Component } from 'react';

class TestApp extends Component {
    state = { }
    render() {
        return ( <H1>Test</H1>);
    }
}
export default TestApp;
```

Add App name in these two places!

React – First Component

```
import React, { Component } from 'react';

class TestApp extends Component {
    state = { }
    render() {
        return ( <div>Test</div>);
    }
}
export default TestApp;
```

This holds state information!

This holds code to render the page. All of the code is placed in the return statement as XML.

Return value can only contain 1 top-level element. Best to use a

React – Additions to index.js

```
import React from "react";
import ReactDOM from "react-dom";
import "./index.css";
import App from "./App";
import * as serviceWorker from "./serviceWorker";
import "bootstrap/dist/css/bootstrap.css";                    // bootstrap css library (already added
                                                                                                  earlier)
import TestApp from './components/TestApp';      // Add this line

// Now change 'App' to 'TestApp'
ReactDOM.render(<App />, document.getElementById("root"));

ReactDOM.render(<TestApp />, document.getElementById("root"));
```

This is what renders the content into a div in the html file!

React – Index.html

- Basic html file in which document is rendered

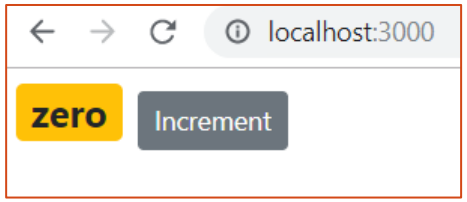
```
<html>
   <head>
   ….

     <title>React App</title>
   </head>
   <body>
      <noscript>You need to enable JavaScript to run this app.</noscript>
      <div id="root"></div>
      …..
   </body>
</html>
```

React – Example – Counter app

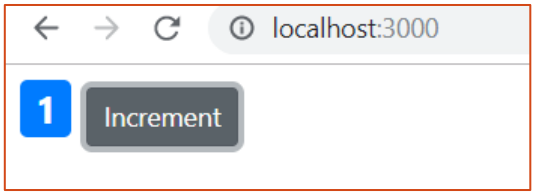
```
import React, { Component } from "react";
class Counter extends Component {

  state = {
    count: 0
  };

  handleIncrement = () => {
    this.setState({ count: this.state.count + 1 });
  };
  render() {
    return (
      <div>
        <span style={{ fontSize: 20 }} className={this.getBadgeClasses()}>{this.formatCount()}</span>
        <button
          className="btn btn-secondary btn-sm"
          onClick={this.handleIncrement}
        >
          Increment
        </button>
      </div>
    );
  }

  getBadgeClasses() {
    let classes = "badge m-2 badge-";
    classes += this.state.count === 0 ? "warning" : "primary";
    return classes;
  }

  formatCount() {
    return this.state.count === 0 ? "zero" : this.state.count;
  }
}

export default Counter;
```

Note: This is jsx (Javascript XML). It should NOT be quoted! It is compiled by 'Babel' into javascript code like calls to createElement(), etc.

React



Initial state



After 1 click on 'Increment' button

React - Events

- React supports Javascript events
  - Events are written in camelCase (onClick= rather than onclick=)
  - Target functions do not need parens () but are placed inside braces {}
    - onClick={this.handleIncrement}

React – Forms

- React provides access to HTML forms
- Similar to Events, handler names are coded in camelCase
  - onChange – When content of an input has changed
  - onSubmit – When a form is submitted

React – Forms - Example

```
import React, { Component } from "react";
class MyForm extends React.Component {
 constructor(props) {
   super(props);
   this.state = { username: '' };
  }

mySubmitHandler = (event) => {
  event.preventDefault();
  alert("You are submitting " + this.state.username);
}

myChangeHandler = (event) => {
  this.setState({username: event.target.value});
}

render() {
  return (
    <form onSubmit={this.mySubmitHandler}>
    <h1>Hello {this.state.username}</h1>
    <p>Enter your name, and submit:</p>
    <input
      type='text'
      onChange={this.myChangeHandler}
    />
    <input
      type='submit'
    />
    </form>
  );
 }
}

export default MyForm;
```

Methods have no parens but are enclosed braces {}

Fires when text input field is changed

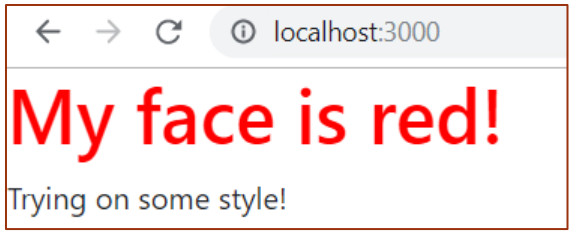
Fires when submit button is pressed

React – CSS

- React supports CSS style information inside jsx
- Since Javascript expressions are encased in braces {} and Javascript objects also use braces, style information will be in 2 sets of braces
- Style attributes use camelCase rather than hyphen separated words
  - background-color => backgroundColor
  - font-family => fontFamily

React – CSS Example

```
import React, { Component } from "react";

class CSSApp extends Component {
  state = {};
  render() {
    return (
      <div>
        <h1 style={{ color: "red" }}>My face is red!</h1>
        <p>Trying on some style!</p>
      </div>
    );
  }
}

export default CSSApp;
```



React - Functions

- React functions can be defined two ways
  - Similar to Javascript:

```
changeColor () {
    this.setState(color: 'blue');
}
```

This code will fail unless you ***bind*** *this in a constructor.*

  - With 'Arrow' notation:

```
changeColor = () => {
    this.setState(color: 'blue');
}
```

- Arrow notation allows access to this keyword representing the component

React – binding 'this'

```
class Car extends React.Component {
  constructor() {
    super()
    this.changeColor = this.changeColor.bind(this)
  }
changeColor () {
    this.setState(color: 'blue');
}
```