Instructor Notes:

Add instructor notes here.

LEARNING FACEBOOK'S

React.JS

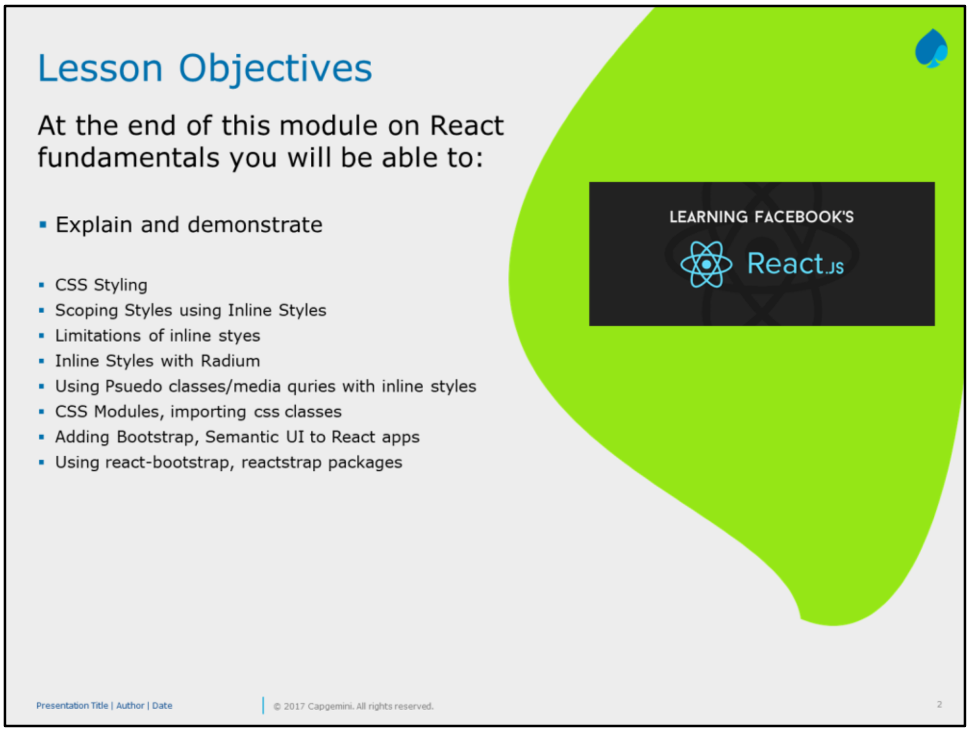
Styling Components

Lesson 04

Capgemini

Instructor Notes:

Add instructor notes here.



Lesson Objectives

At the end of this module on React fundamentals you will be able to:

- Explain and demonstrate
- CSS Styling
- Scoping Styles using Inline Styles
- Limitations of inline styes
- Inline Styles with Radium
- Using Psuedo classes/media quries with inline styles
- CSS Modules, importing css classes
- Adding Bootstrap, Semantic UI to React apps
- Using react-bootstrap, reactstrap packages

LEARNING FACEBOOK'S React.JS

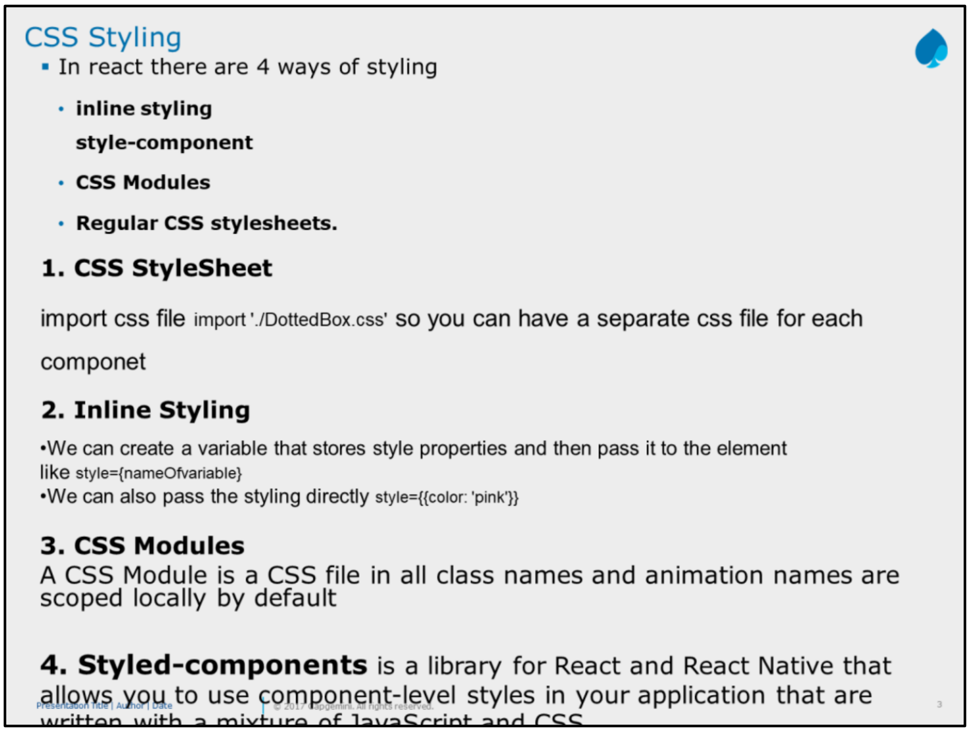
Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

2

Instructor Notes:

Slide explains that communication between service provider and consumer happen via SOAP messages



CSS Styling

- In react there are 4 ways of styling
  - **inline styling**
    style-component
  - **CSS Modules**
  - **Regular CSS stylesheets.**

1. CSS StyleSheet

import css file import './DottedBox.css' so you can have a separate css file for each componet

2. Inline Styling

•We can create a variable that stores style properties and then pass it to the element like style={nameOfvariable}
•We can also pass the styling directly style={{color: 'pink'}}

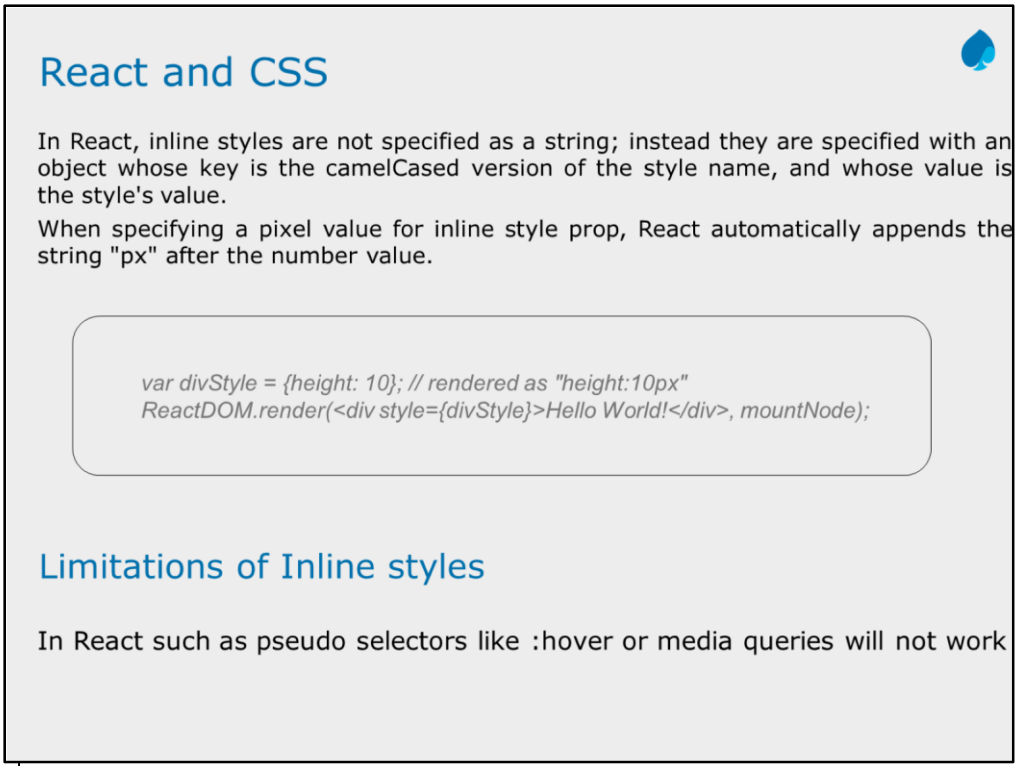
3. CSS Modules
A CSS Module is a CSS file in all class names and animation names are scoped locally by default

4. Styled-components is a library for React and React Native that allows you to use component-level styles in your application that are written with a mixture of JavaScript and CSS

Presentation Title | Author | Date © 2017 Capgemini. All rights reserved. 3

Instructor Notes:

Add instructor notes here.



React and CSS

In React, inline styles are not specified as a string; instead they are specified with an object whose key is the camelCased version of the style name, and whose value is the style's value.

When specifying a pixel value for inline style prop, React automatically appends the string "px" after the number value.

```
var divStyle = {height: 10}; // rendered as "height:10px"
ReactDOM.render(<div style={divStyle}>Hello World!</div>, mountNode);
```

Limitations of Inline styles

In React such as pseudo selectors like :hover or media queries will not work

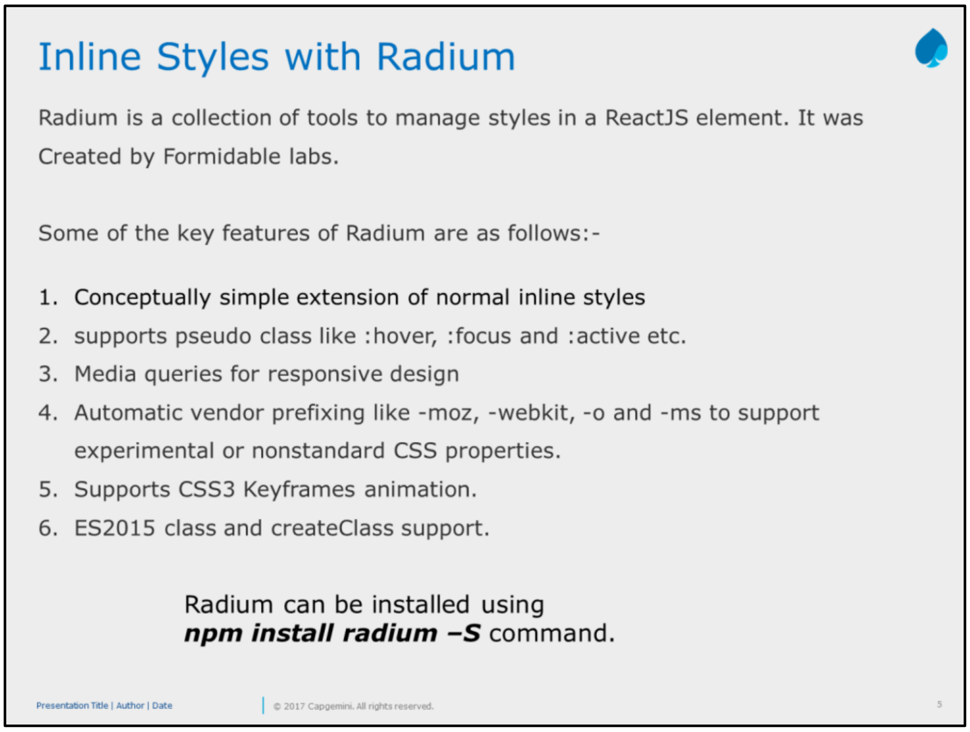
There are a number of attributes that work differently between React and HTML:

className is used instead of class attribute in normal html
The style attribute accepts a JavaScript object with camelCased properties rather than a CSS string.

List of properties that won't get the automatic "px" suffix:

- animationIterationCount
- boxFlex
- boxFlexGroup
- boxOrdinalGroup
- columnCount
- fillOpacity
- flex
- flexGrow
- flexPositive
- flexShrink
- flexNegative
- flexOrder
- fontWeight
- lineClamp
- lineHeight
- opacity
- order
- orphans
- stopOpacity
- strokeDashoffset
- strokeOpacity
- strokeWidth
- tabSize
- widows
- zIndex
- zoom

Instructor Notes:



Inline Styles with Radium

Radium is a collection of tools to manage styles in a ReactJS element. It was Created by Formidable labs.

Some of the key features of Radium are as follows:-

1. Conceptually simple extension of normal inline styles
2. supports pseudo class like :hover, :focus and :active etc.
3. Media queries for responsive design
4. Automatic vendor prefixing like -moz, -webkit, -o and -ms to support experimental or nonstandard CSS properties.
5. Supports CSS3 Keyframes animation.
6. ES2015 class and createClass support.

Radium can be installed using
npm install radium –S command.

Presentation Title | Author | Date © 2017 Capgemini. All rights reserved. 5

Problem in using css style sheet and inline style:

Radium resolves nested style objects into a flat object that can be applied directly to a React element.

If we use inline style we assign values through javascript objects and there we cannot use pseudo-selectors. So we move on to external css file and use className in code, which again will have impact on total application.for example All the button or text boxes will be affected instead of a particular button.

It was created by FormidableLabs.

Its like radium = inlineStyles++;

Use npm install radium,

After installing,

wrap Radium() around your component, like module.exports = Radium(Component), or Component = Radium(Component).
//this is supported in class, stateless createClass

Pass the style object to your component via style={...}

Rest taken care by Radium.

Example:

```
var React = require('react'); var Radium = require('radium');
 var studentList = [
 {name:"Sandeep", subject:"Physics", roll:4},
{name:"John", subject:"Chemistry", roll:1},
{name:"Smith", subject:"Physics", roll:2},
{name:"Ramesh", subject:"Chemistry", roll:3} ];

var styles = { header: { color: '#999', ':hover':{ color:'#00f' }, ':focus':{ color:'#f00' }, ':active':{ color:'#000' } } };

 var StudentListComponent = React.createClass({ render: function() { var results = this.props.results;
 return ( <ol> {studentList.map(function(student) { return <li key={student.roll}> <a key={student.roll}
style={[styles.header]} href="#"> {student.name} </a> </li>; })} </ol> ); } });

module.exports = Radium(StudentListComponent);
```

Radium is activated by wrapping your component:
class Button extends React.Component { // ... } export default Radium(Button); // or class Button extends React.Component { // ... } Button = Radium(Button);

Instructor Notes:

Add instructor notes here.

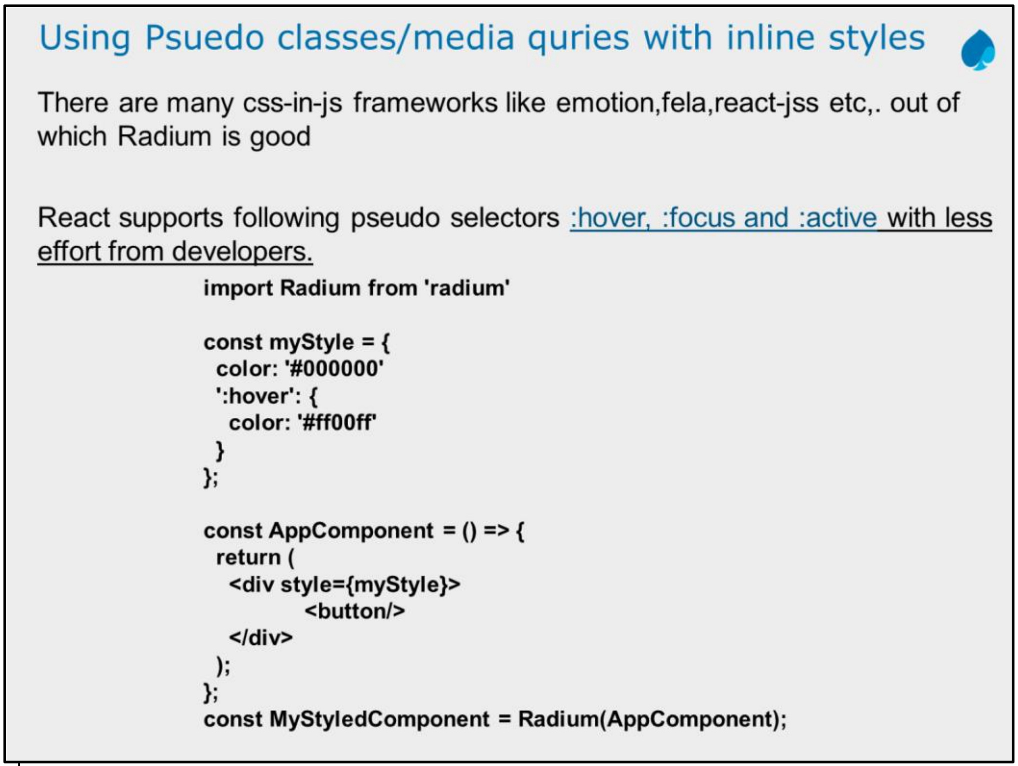
Demo

Inline-Styles

- create-react-styles

Instructor Notes:

Add instructor notes here.



Using Psuedo classes/media quries with inline styles

There are many css-in-js frameworks like emotion,fela,react-jss etc,. out of which Radium is good

React supports following pseudo selectors :hover, :focus and :active with less effort from developers.

```
import Radium from 'radium'

const myStyle = {
  color: '#000000'
  ':hover': {
    color: '#ff00ff'
  }
};

const AppComponent = () => {
  return (
    <div style={myStyle}>
          <button/>
    </div>
  );
};
const MyStyledComponent = Radium(AppComponent);
```

Known issues with media queries

IE9 supports CSS media queries, but doesn't support the matchMedia API. You'll need a polyfill that includes addListener.

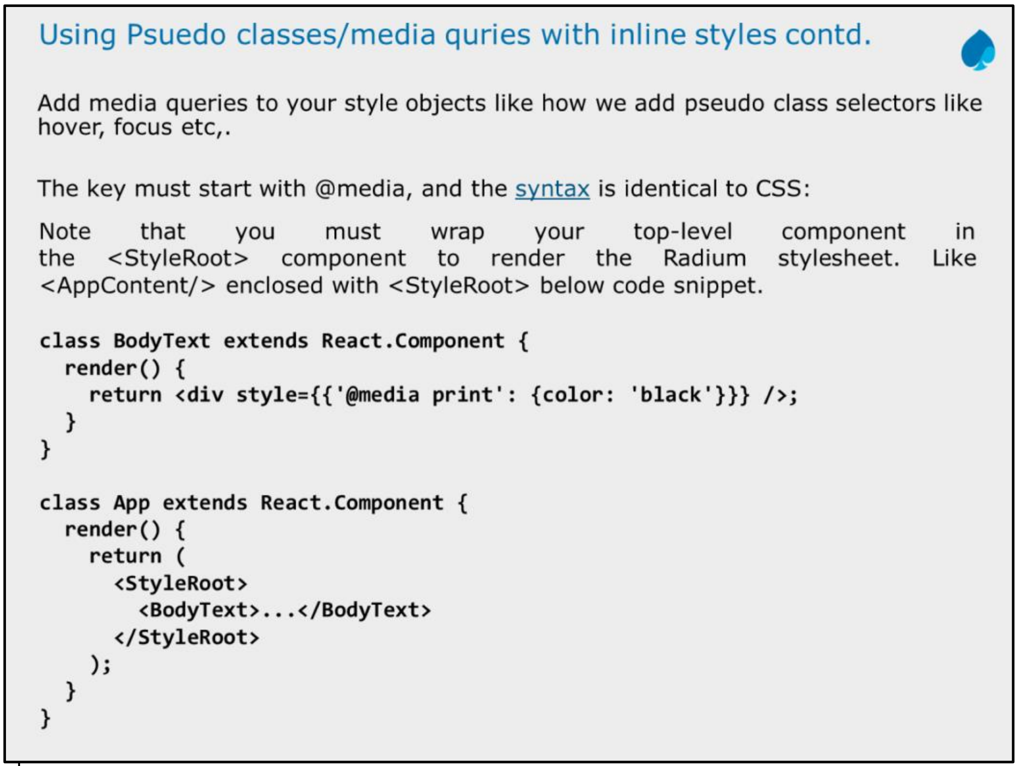
Media queries:

If you add any @media in the css code it will generate belwo error

Error: To use plugins requiring `addCSS` (e.g. keyframes, media queries), please wrap your application in the StyleRoot component.

Instructor Notes:

Add instructor notes here.



Using Psuedo classes/media quries with inline styles contd.

Add media queries to your style objects like how we add pseudo class selectors like hover, focus etc,.

The key must start with @media, and the syntax is identical to CSS:

Note that you must wrap your top-level component in the <StyleRoot> component to render the Radium stylesheet. Like <AppContent/> enclosed with <StyleRoot> below code snippet.

```
class BodyText extends React.Component {
  render() {
    return <div style={{'@media print': {color: 'black'}}} />;
  }
}

class App extends React.Component {
  render() {
    return (
      <StyleRoot>
        <BodyText>...</BodyText>
      </StyleRoot>
    );
  }
}
```

Known issues with media queries

IE9 supports CSS media queries, but doesn't support the matchMedia API. You'll need a polyfill that includes addListener.

Media queries:

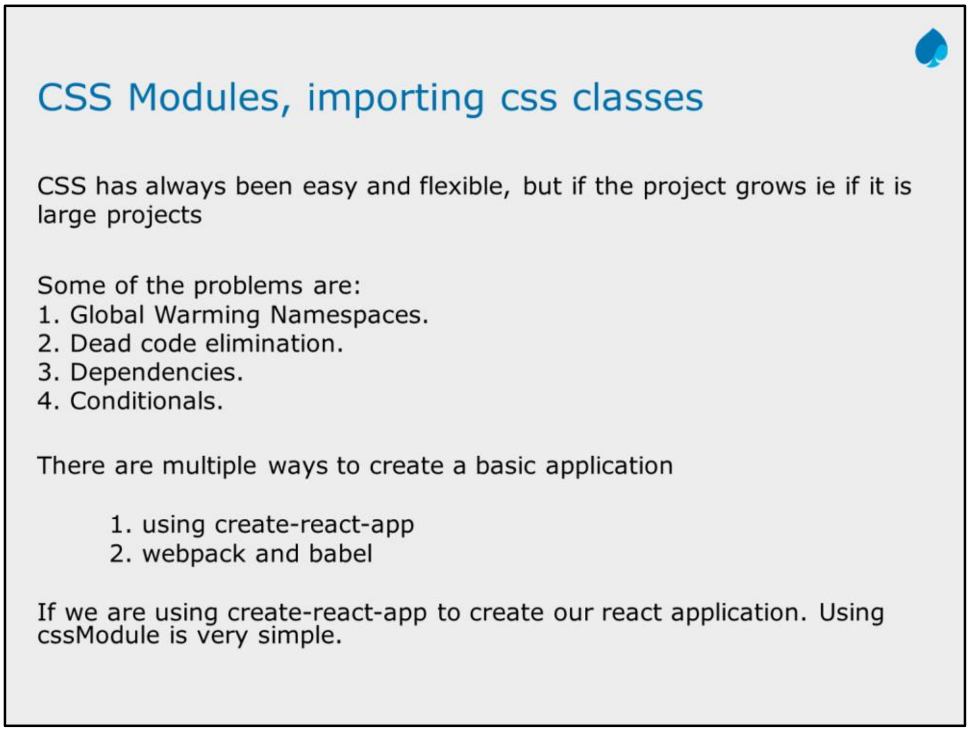
If you add any @media in the css code it will generate belwo error

Error: To use plugins requiring `addCSS` (e.g. keyframes, media queries), please wrap your application in the StyleRoot component.

Top level CSS rules in your media queries will be converted to CSS and rendered in an actual <style> element with !important appended instead of being applied inline so they will work with server-side rendering.

To sole above problem we should wrap component by <StyleRoot>

Instructor Notes:



CSS Modules, importing css classes

CSS has always been easy and flexible, but if the project grows ie if it is large projects

Some of the problems are:
1. Global Warming Namespaces.
2. Dead code elimination.
3. Dependencies.
4. Conditionals.

There are multiple ways to create a basic application

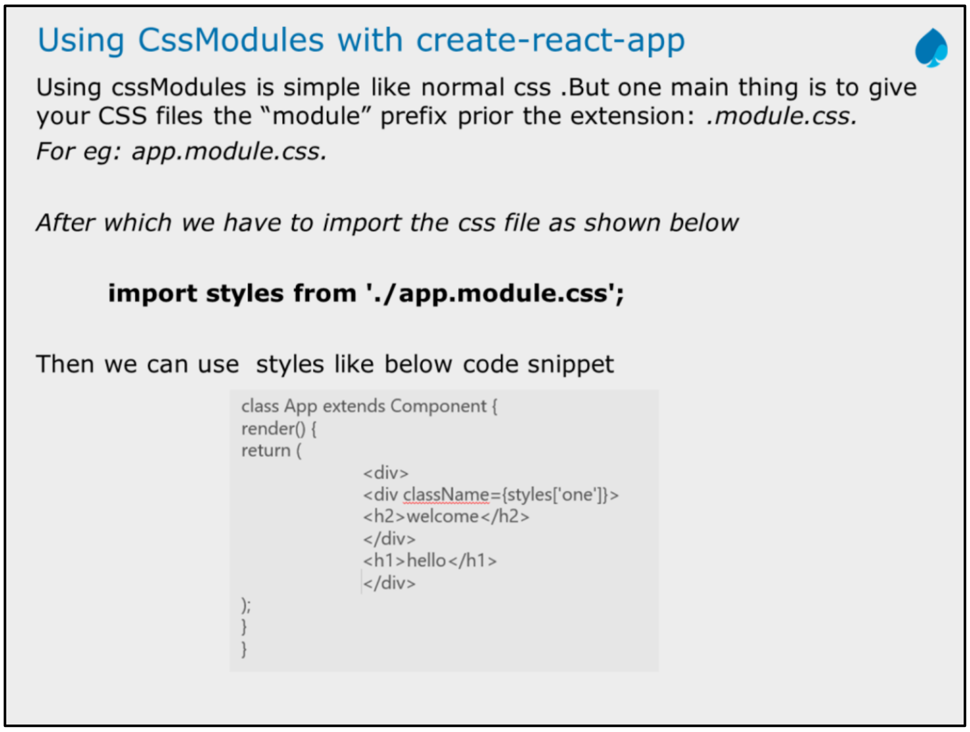
1. using create-react-app
2. webpack and babel

If we are using create-react-app to create our react application. Using cssModule is very simple.

Global namespaces
CSS is a language of globals. In most languages, global variables are considered bad and taking into account we have CSS styles mostly specific to our components, it’s obvious that globals are bad (there are cases when using global CSS is okay though).
Global CSS can easily rewrite styling in other classes. A change in the style in one class can ultimately and silently change the UI on multiple components. Generally, no one wants that to happen when they develop apps.
Dead code elimination
Most times while building apps, we have styling that is redundant or no longer in use. Although they can be small at first, over time it builds up and we have rogue and unused CSS everywhere and it can cause unnecessarily large file sizes.
With CSS modules, we get to only import styles we use. therefore reducing redundancy and the increase in our ability to spot CSS that is not being used.
Dependencies
Working with dependencies in CSS is hazy. Most of the time we are used to just including the entire script directly into the page whether they are being used or not.
Conditionals
Sometimes you need your CSS class to style in a certain way in one section and in a different way in another section. In this case, it is usually better to have the CSS class variable modifiable in the component.
Modular CSS solves each of these problems in a clean structured manner by having our components import only the classes they actually use therefore getting rid of redundancy. This in itself also solves the problems of global namespaces.

From
https://blog.pusher.com/css-modules-react/

Instructor Notes:



Using CssModules with create-react-app

Using cssModules is simple like normal css .But one main thing is to give your CSS files the "module" prefix prior the extension: *.module.css.*

For eg: app.module.css.

After which we have to import the css file as shown below

import styles from './app.module.css';

Then we can use styles like below code snippet

```
class App extends Component {
render() {
return (
              <div>
              <div className={styles['one']}>
              <h2>welcome</h2>
              </div>
              <h1>hello</h1>
              </div>
);
}
}
```

We can also use stlyes using className like below

className={styles.one}

Instructor Notes:

Add instructor notes here.

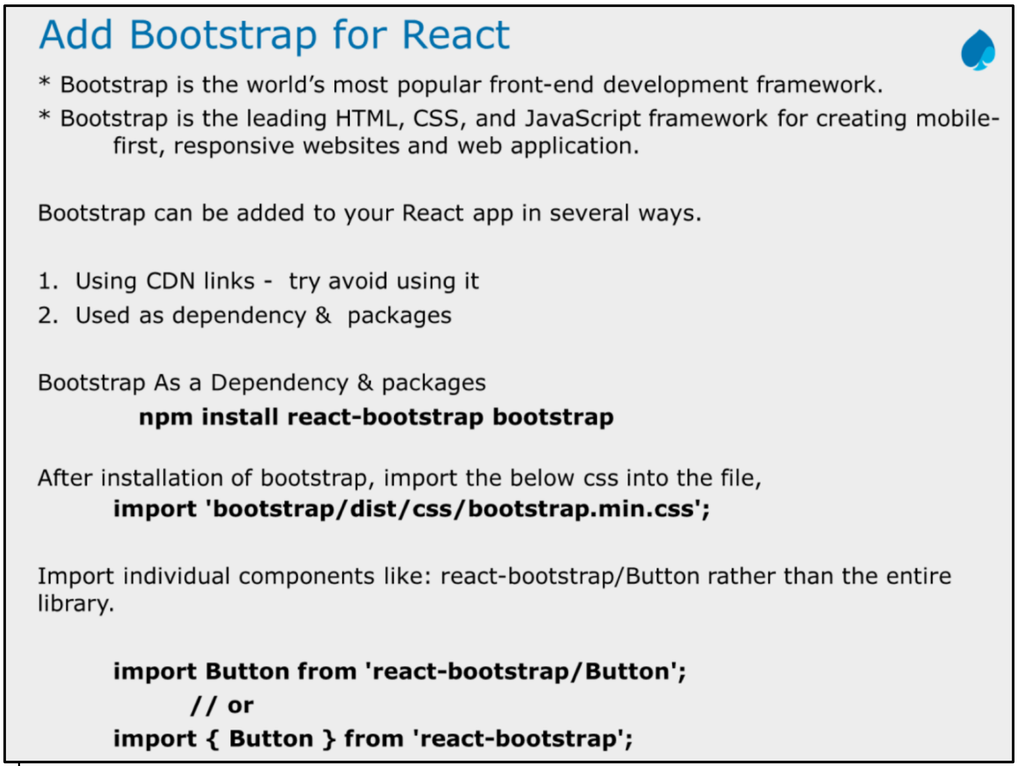
Demo

Styling

- react-style-pseudoselector
- react-radium-2019
- react-create-app-radium-cssmodule

Instructor Notes:

Add instructor notes here.



Add Bootstrap for React

* Bootstrap is the world's most popular front-end development framework.

* Bootstrap is the leading HTML, CSS, and JavaScript framework for creating mobile-first, responsive websites and web application.

Bootstrap can be added to your React app in several ways.

1. Using CDN links - try avoid using it
2. Used as dependency & packages

Bootstrap As a Dependency & packages

```
npm install react-bootstrap bootstrap
```

After installation of bootstrap, import the below css into the file,

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

Import individual components like: react-bootstrap/Button rather than the entire library.

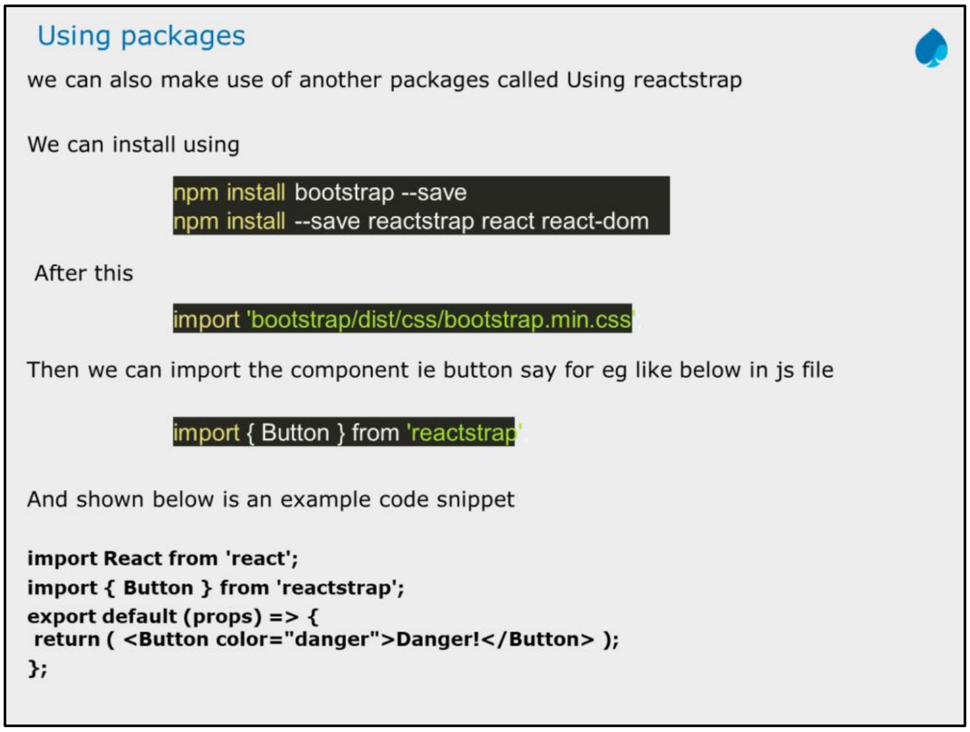
```
import Button from 'react-bootstrap/Button';
        // or
import { Button } from 'react-bootstrap';
```

For other components
https://react-bootstrap.github.io/components/

Bootstrap cdn:

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vb
dEjh4u" crossorigin="anonymous"> <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-
theme.min.css" integrity="sha384-
rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXw
l/Sp" crossorigin="anonymous"> <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384-
Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIc
PD7Txa" crossorigin="anonymous"></script>

Instructor Notes:



Using packages

we can also make use of another packages called Using reactstrap

We can install using

```
npm install bootstrap --save
npm install --save reactstrap react react-dom
```

After this

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

Then we can import the component ie button say for eg like below in js file

```
import { Button } from 'reactstrap'
```

And shown below is an example code snippet

```
import React from 'react';
import { Button } from 'reactstrap';
export default (props) => {
 return ( <Button color="danger">Danger!</Button> );
};
```

Other components can be taken from below link

https://reactstrap.github.io/components/

We can also use CDN links as given below into the project

https://cdnjs.cloudflare.com/ajax/libs/reactstrap/4.8.0/reactstrap.min.js

If you prefer to include Reactstrap globally by marking reactstrap as external in your application, the reactstrap library provides various single-file distributions, which are hosted on the following CDNs:
cdnjs

<!-- Main version --> <script src="https://cdnjs.cloudflare.com/ajax/libs/reactstrap/6.0.1/reactstrap.min.js"></script> <!-- All optional dependencies version --> <script src="https://cdnjs.cloudflare.com/ajax/libs/reactstrap/6.0.1/reactstrap.full.min.js"></script>
unpkg
<!-- Main version --> <script src="https://unpkg.com/reactstrap@6.0.1/dist/reactstrap.min.js"></script> <!-- All optional dependencies version --> <script src="https://unpkg.com/reactstrap@6.0.1/dist/reactstrap.full.min.js"></script>
Note: To load a specific version of Reactstrap replace 6.0.1 with the version number.
Versions
Reactstrap has two primary distribution versions:
reactstrap.min.js
This file **excludes** the optional dependencies – react-popper and react-transition-group. This is the recommended approach (similar approach in Bootstrap's JavaScript components) for including Reactstrap as it reduces the filesize and gives more flexibility in configuring needed dependencies.
Recommended use cases:
Small, medium, or large applications
Applications that do not use any transitions or popper components
Applications that directly use react-popper or react-transition-group – Reactstrap and your application will use the single global version included
reactstrap.full.min.js
This file **includes** the optional dependencies – react-popper and react-transition-group
Recommended use cases:
Small applications

<!doctype html> <html lang="en"> <head> <!-- Required dependencies --> <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/prop-types/15.6.1/prop-types.min.js"></script> <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/react/16.3.2/umd/react.production.min.js"></script> <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/react-dom/16.3.2/umd/react-dom.production.min.js"></script> <!-- Optional dependencies --> <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/react-transition-group/2.2.1/react-transition-group.min.js"></script> <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"></script> <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/react-popper/0.10.4/umd/react-popper.min.js"></script> <!-- Reactstrap --> <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/reactstrap/6.0.1/reactstrap.min.js"></script> <!-- Lastly, include your app's bundle --> <script type="text/javascript" src="/assets/bundle.js"></script> </head> <body> <div id="my-app" /> </body> </html>

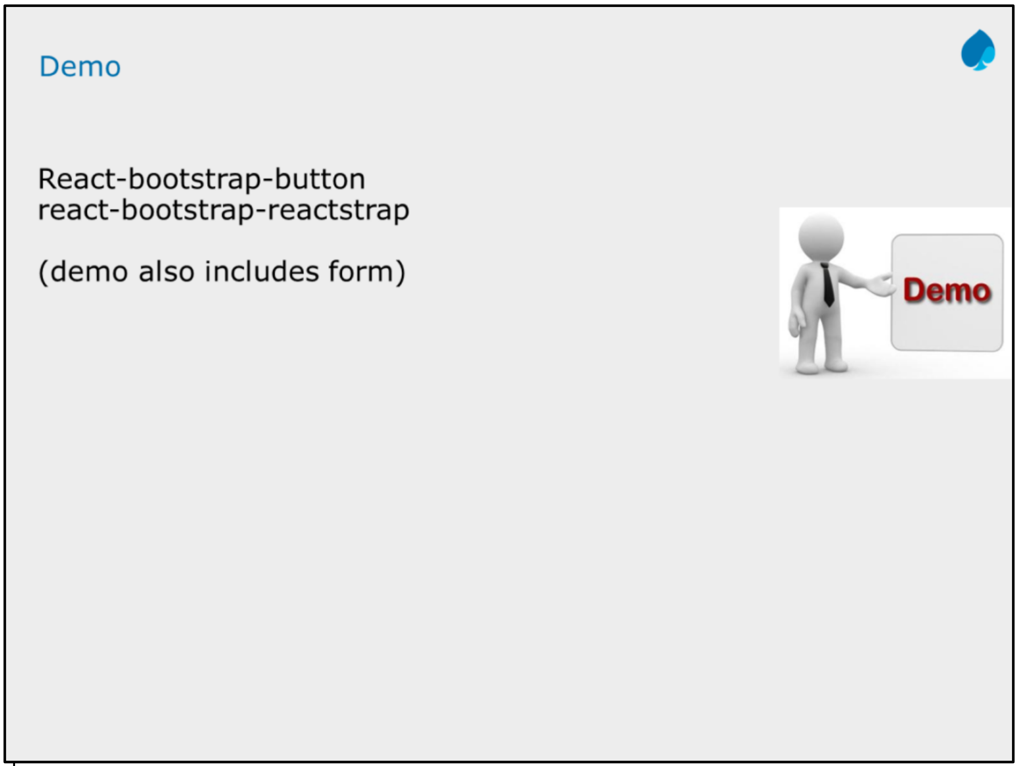
Advantages of bootstrap:

1. Responsive design
2. Consistency
3. Quick and efficient

Responsive design : Bootstrap works with almost any size of screen - perfect for the mobile era.
Consistency: The grid layout delivered by the JavaScript components and CSS elements means you can ensure consistency across all browser versions with Bootstrap.
Quick and efficient: Bootstrap can get your site up and running quickly, with each component configured to ensure performance.

Instructor Notes:

Add instructor notes here.



Demo

React-bootstrap-button
react-bootstrap-reactstrap

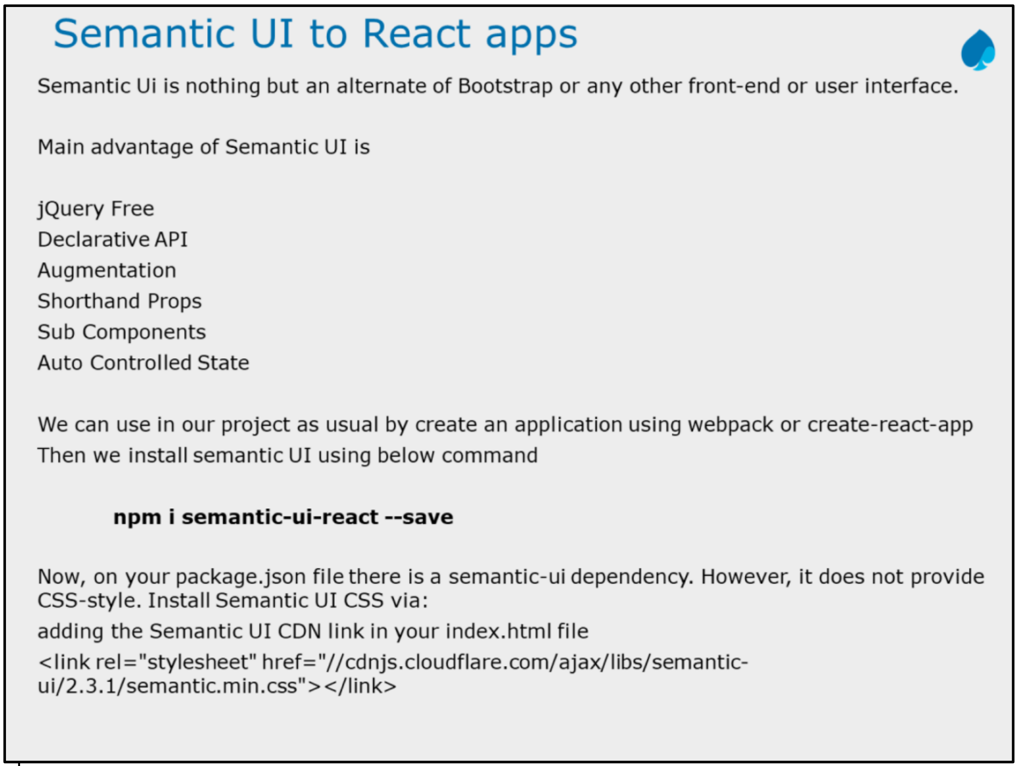
(demo also includes form)

If you are new to React and using **create-react-app cli** setup. Then run the below NPM command to include the latest version of bootstrap.

```
npm install --save bootstrap@4.0.0-alpha.6
```

Instructor Notes:

Add instructor notes here.



Semantic UI to React apps

Semantic Ui is nothing but an alternate of Bootstrap or any other front-end or user interface.

Main advantage of Semantic UI is

jQuery Free
Declarative API
Augmentation
Shorthand Props
Sub Components
Auto Controlled State

We can use in our project as usual by create an application using webpack or create-react-app
Then we install semantic UI using below command

```
npm i semantic-ui-react --save
```

Now, on your package.json file there is a semantic-ui dependency. However, it does not provide CSS-style. Install Semantic UI CSS via:
adding the Semantic UI CDN link in your index.html file

```
<link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/semantic-ui/2.3.1/semantic.min.css"></link>
```

jQuery Free

jQuery is a DOM manipulation library. It reads from and writes to the DOM. React uses a virtual DOM (a JavaScript representation of the real DOM). React only *writes* patch updates to the DOM, but *never reads* from it.
It is not feasible to keep real DOM manipulations in sync with React's virtual DOM. Because of this, all jQuery functionality has been re-implemented in React

Declarative API

easy for props validation

Augmentation

Control the rendered HTML tag, or render one component as another component. Extra props are passed to the component you are rendering as

Shorthand Props
Shorthand props generate markup for you, making many use cases a breeze. All object props are spread on the child components.
Child Object Arrays
Components with repeating children accept arrays of plain objects. Facebook is fond of this over using context to handle parent-child coupling and so are we.

Instru

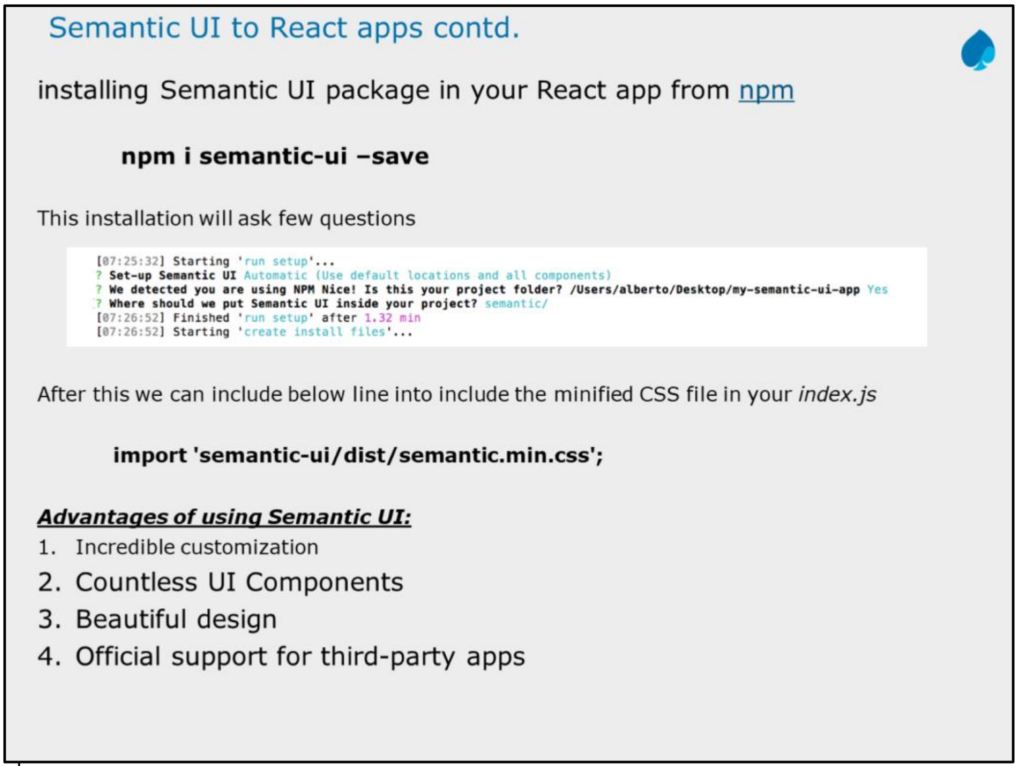
Sub Components
Sub components give you complete access to the markup. This is essential for flexibility in customizing components.

Auto Controlled State
React has the concept of controlled and uncontrolled components.

Our stateful components self manage their state out of the box, without wiring. Dropdowns open on click without wiring onClick to the openprop. The value is also stored internally, without wiring onChange to value.
If you add a value prop or an open prop, the Dropdown delegates control for that one prop to your value. The other props remain auto controlled. Mix and match any number of controlled and uncontrolled props. Add and remove control at any time by adding or removing props. Everything just works.

Instructor Notes:

Add instructor notes here.



Semantic UI to React apps contd.

installing Semantic UI package in your React app from npm

npm i semantic-ui –save

This installation will ask few questions

```
[07:25:32] Starting 'run setup'...
? Set-up Semantic UI Automatic (Use default locations and all components)
? We detected you are using NPM Nice! Is this your project folder? /Users/alberto/Desktop/my-semantic-ui-app Yes
? Where should we put Semantic UI inside your project? semantic/
[07:26:52] Finished 'run setup' after 1.32 min
[07:26:52] Starting 'create install files'...
```

After this we can include below line into include the minified CSS file in your *index.js*

import 'semantic-ui/dist/semantic.min.css';

Advantages of using Semantic UI:

1. Incredible customization
2. Countless UI Components
3. Beautiful design
4. Official support for third-party apps

jQuery Free

jQuery is a DOM manipulation library. It reads from and writes to the DOM. React uses a virtual DOM (a JavaScript representation of the real DOM). React only *writes* patch updates to the DOM, but *never reads* from it.
It is not feasible to keep real DOM manipulations in sync with React's virtual DOM. Because of this, all jQuery functionality has been re-implemented in React

Declarative API

easy for props validation

Augmentation

Control the rendered HTML tag, or render one component as another component. Extra props are passed to the component you are rendering as

Shorthand Props
Shorthand props generate markup for you, making many use cases a breeze. All object props are spread on the child components.
Child Object Arrays
Components with repeating children accept arrays of plain objects. Facebook is fond of this over using context to handle parent-child coupling and so are we.

Instru

Sub Components
Sub components give you complete access to the markup. This is essential for flexibility in customizing components.

Auto Controlled State
React has the concept of controlled and uncontrolled components.

Our stateful components self manage their state out of the box, without wiring. Dropdowns open on click without wiring onClick to the openprop. The value is also stored internally, without wiring onChange to value.
If you add a value prop or an open prop, the Dropdown delegates control for that one prop to your value. The other props remain auto controlled. Mix and match any number of controlled and uncontrolled props. Add and remove control at any time by adding or removing props. Everything just works.

Advantages of Semantic UI:

Incredible customisation: Although Bootstrap does offer lots of personalisation solutions, the extra elements in Semantic UI mean that you get far more options, even beyond the potential of Bootstrap.

Load the components you need: The complete Semantic UI package is very well organised, with each component established in its own file.

Countless UI Components: Semantic UI comes with almost any UI component you can think of for your project, including a ton of UI components for mobile and responsive solutions.

Well documented: Like Bootstrap, Semantic UI is very well documented, with lots of examples to follow and use.

Beautiful design: Semantic UI has a stunning and modern design, which makes it great for a range of different developers.
Official support for third-party apps: Finally, there are several implementations of Semantic UI for different libraries like WordPress, Angular, and so on.

Instructor Notes:

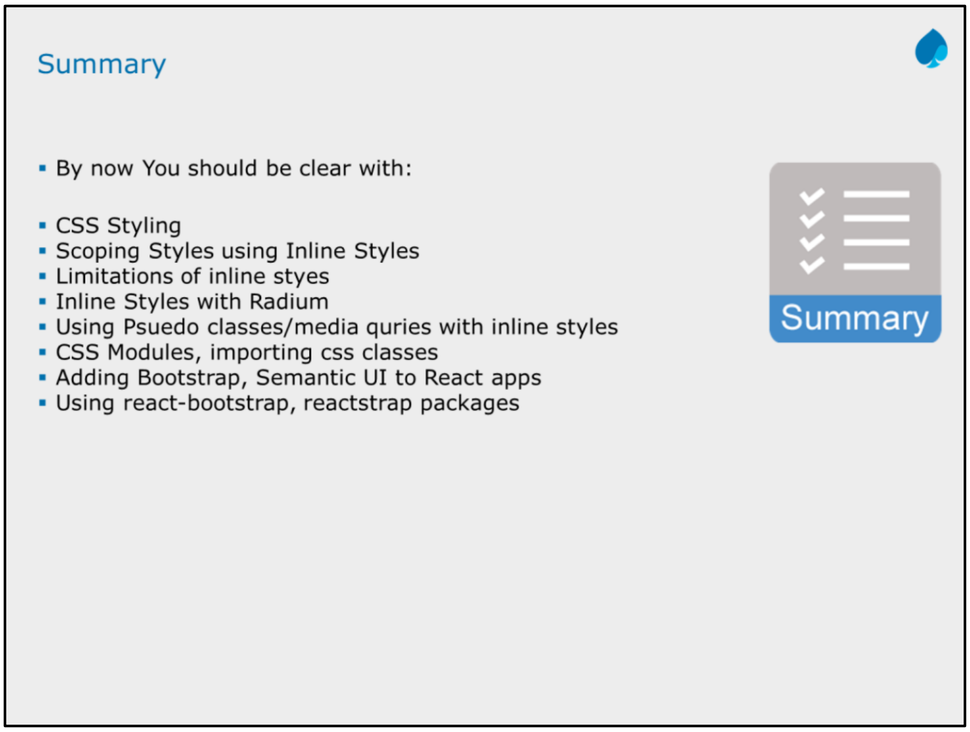
Add instructor notes here.

Demo

React-semantic-2019

Instructor Notes:

Add instructor notes here.



Summary

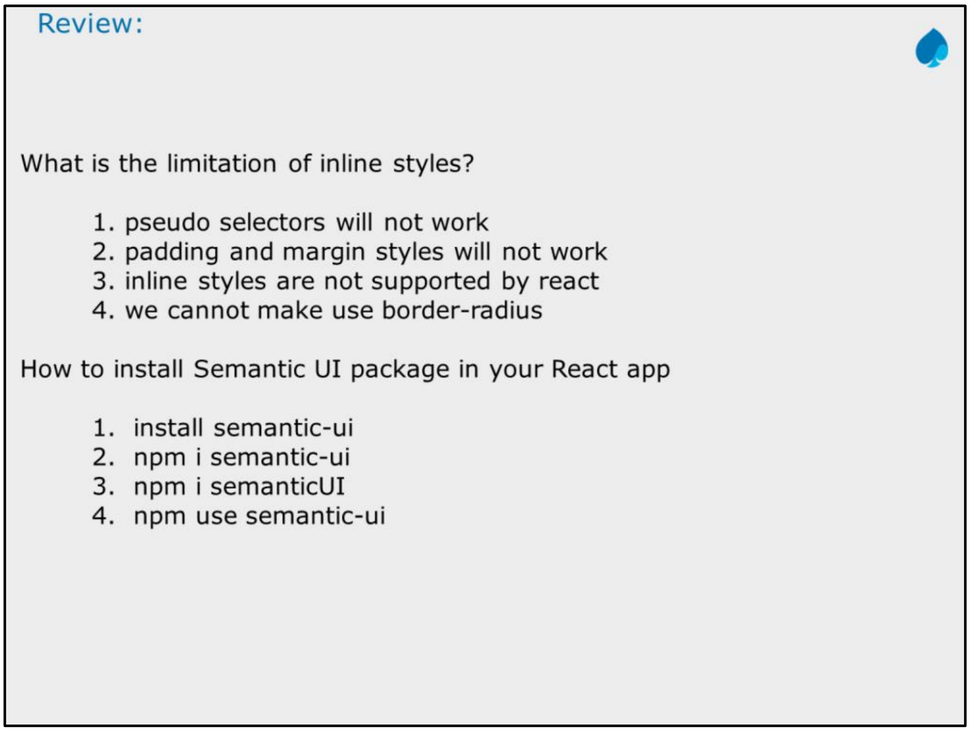
- By now You should be clear with:

- CSS Styling
- Scoping Styles using Inline Styles
- Limitations of inline styes
- Inline Styles with Radium
- Using Psuedo classes/media quries with inline styles
- CSS Modules, importing css classes
- Adding Bootstrap, Semantic UI to React apps
- Using react-bootstrap, reactstrap packages

Summary

Add the notes here.

Instructor Notes:



Review:

What is the limitation of inline styles?

1. pseudo selectors will not work
2. padding and margin styles will not work
3. inline styles are not supported by react
4. we cannot make use border-radius

How to install Semantic UI package in your React app

1. install semantic-ui
2. npm i semantic-ui
3. npm i semanticUI
4. npm use semantic-ui