ECAP472

Web Technologies

Dr. Pritpal Singh

Associate Professor

Learning Outcomes

After this lecture, you will be able to

- Understand concept of creating and rendering React Components.

The Render Function

- React's goal is in many ways to render HTML in a web page.
- React renders HTML to the web page by using a function called ReactDOM.render().

The Render Function

The ReactDOM.render() function takes two arguments:

- HTML code and an
- HTML element.

The purpose of the function is to display the specified HTML code inside the specified HTML element.

But render where?

There is another folder in the root directory of your React project, named "public". In this folder, there is an index.html file.

You'll notice a single <div> in the body of this file. This is where our React application will be rendered.

Example

Display a paragraph inside an element with the id of "root":

```
ReactDOM.render(<p>Hello</p>,
document.getElementById('root’));
```

The result is displayed in the <div id="root"> element:

```
<body>
  <div id="root"></div>
</body>
```

The Root Node

- The root node is the HTML element where you want to display the result.
- It is like a container for content managed by React.
- It does NOT have to be a <div> element and it does NOT have to have the id='root':

What is JSX?

- JSX stands for JavaScript XML.
- JSX allows us to write HTML in React.
- JSX makes it easier to write and add HTML in React.

Coding JSX

- JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods.
- JSX converts HTML tags into react elements.
- You are not required to use JSX, but JSX makes it easier to write React applications.

Example 1

- Here are two examples. The first uses JSX and the second does not:
  - const myelement = <h1>I Love JSX!</h1>;
  - ReactDOM.render(myelement,document.getElementById('root'));

Example 2

- const myelement = React.createElement('h1', {}, 'I do not use JSX!');
  - ReactDOM.render(myelement, document.getElementById('root'));

Observation

As you can see in the first example, JSX allows us to write HTML directly within the JavaScript code.

JSX is an extension of the JavaScript language based on ES6, and is translated into regular JavaScript at runtime.

Expressions in JSX

- With JSX you can write expressions inside curly braces { }.
- The expression can be a React variable, or property, or any other valid JavaScript expression. JSX will execute the expression and return the result

Example

- Execute the expression 5 + 5:
- const myelement = <h1>React is {5 + 5} times better with JSX</h1>;

React Components

- Components are like functions that return HTML elements.
- Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML.
- Components come in two types, Class components and Function components.

Create Your First Component

- When creating a React component, the component's name MUST start with an upper case letter.
- A class component must include the extends React.Component statement. This statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.
- The component also requires a render() method, this method returns HTML.

Create Your First Component

- When creating a React component, the component's name MUST start with an upper case letter.
- A class component must include the extends React.Component statement. This statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.
- The component also requires a render() method, this method returns HTML.

Create Your First Component

- When creating a React component, the component's name MUST start with an upper case letter.
- A class component must include the extends React.Component statement. This statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.
- The component also requires a render() method, this method returns HTML.

Example

Create a Class component called Car

```
class Car extends React.Component
 {
  render() {
    return <h2>Hi, I am a Car!</h2>;
  }
}
```

Function Component

A Function component also returns HTML, and behaves much the same way as a Class component, but Function components can be written using much less code, are easier to understand.

Example

Create a Function component called Car

```
function Car()

{

  return <h2>Hi, I am a Car!</h2>;

}
```

Rendering a Component

- Now your React application has a component called Car, which returns an <h2> element.
- To use this component in your application, use similar syntax as normal HTML: <Car />

Why Should You Use React?

React is Flexible:

- React is remarkably flexible. Once you have learned it, you can use it on a vast variety of platforms to build quality user interfaces. React is a library, NOT a framework. Its library approach has allowed React to evolve into such a remarkable tool.
- React was created with a single focus: to create components for web applications. A React component can be anything in your web application like a Button, Text, Label, or Grid.

React Has a Great Developer Experience

- You will fall in love with React when they start coding in it. Rapid development and React's small API combined creates a fantastic developer experience.
- React's API is very simple to learn. It has very few concepts to learn.

React Has Facebook's Support/Resources

- React is heavily used in the Facebook app, website, and in Instagram. That's why Facebook is deeply committed to it. They use over 50k React components in their production environment. The top four React contributors on GitHub are full-time Facebook employees.

React Has Facebook's Support/Resources

- Also, the React team maintains a blog that consistently gives you details for each release.
- Because of the deep commitment by Facebook to React in production, when breaking change occur in React, Facebook consistently provides Codemod that automates the change.

React Has Broader Community Support, Too

- Since 2015, React's popularity has grown steadily. It has a massive active community and its GitHub Repository has over 164k Stars. It is one of the Top 5 Repositories on GitHub.

React Has Great Performance

- The React team realized that JavaScript is fast, but updating the DOM makes it slow. React minimizes DOM changes. And it has figured out the most efficient and intelligent way to update DOM.
- Before React, most frameworks and libraries would update the DOM unintelligently to reflect a new state. This resulted in changes to a significant portion of the page.

React is Easy to Test

- React's design is very user friendly for testing.
- Traditional UI browser testing is a hassle to setup. On the other hand, you require very little or no configuration for testing in React.
- Traditional UI browser requires browsers for testing, but you can test React components quickly and easily using the node command-line.

React is Easy to Test

- Traditional UI browser testing is slow. But command-line testing is fast, and you can run a considerable amount of test suites at a time.
- Traditional UI browser testing is often time consuming and challenging to maintain. React test can be written quickly using tools like Jest & Enzyme.

React

- React is an excellent tool with which to create interactive applications for mobile, web, and other platforms.
- React's popularity and usage are increasing day by day for good reason. As a developer, coding in React makes you better at JavaScript, a language that holds nearly 90% of the web development share today.

That’s all for now…