前端網站開發人員證書課程
(二) 進階網絡程式設計--專業React.js應用

3. React JSX and Components

Presented by Krystal Institute

Lesson Outline

- Introduce the JSX and components
- The steps to create a class and functional component
- How to include CSS and Bootstrap to React

3.1 What is JSX?

3.1.1 What is JSX?

- JSX is a JavaScript Extension Syntax used in React to write HTML and JavaScript together easily.
- Syntax:

```
const element = <h1>Hello, world!</h1>;
```

- This is simple JSX code in React. But the browser does not understand this JSX because it's not valid JavaScript code.
- We're assigning an HTML tag to a variable that is not a string but just HTML code.
- So to convert it to browser understandable JavaScript code, we use a tool like Babel, a JavaScript compiler/transpiler.

3.1.2 Babel

- Babel is a JavaScript Compiler.
- Babel is a toolchain that is mainly used to convert ECMAScript 2015+ code into a backward-compatible version of JavaScript in current and older browsers or environments.
- Here are the main things Babel can do for you:
  - Transform syntax.
  - Polyfill features that are missing in your target environment (through a third-party polyfill such as core-js.
  - Source code transformations (code mods)

3.1.3 React JSX

- React JSX is an extension to JavaScript.
- It enables developers to create virtual DOM using XML syntax.
- It compiles down to pure JavaScript (React.createElement function calls).
- Since it compiles JavaScript, it can be used inside any valid JavaScript code.
- Assign a variable:

```
var greeting = <h1>Hello React!</h1>
```

- Assign a variable based on a condition:

```
var canGreet = true;
if(canGreet) {
greeting = <h1>Hello React!</h1>
}
```

3.1.3 React JSX

- To return value of function:

```
function Greeting() {
return <h1>Hello React!</h1>

}
greeting = Greeting()
```

- The argument of a function:

```
function Greet(message) {
    ReactDOM.render(message, document.getElementById('react-app')
    }
Greet(<h1>Hello React!</h1>)
```

3.2 JSX and JavaScript Expressions

3.2.1 JSX and JavaScript Expressions

- JSX supports expression in pure JavaScript syntax.
- The expression has to be enclosed inside the curly braces, { }.
- The expression can contain all variables available in the context, where the JSX is defined.
- Example:

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;
```

- In the example above, we declare a variable called name and then use it inside JSX by wrapping it in curly braces.
- Inside the curly braces, any valid JavaScript expression can be included.

3.2.2 JSX and JavaScript Expressions

- An expression is any valid unit of code that resolves to a value.
- Every syntactically valid expression resolves to some value but conceptually, there are two types of expressions:
- Assign the value to a variable. Example: x = 7;
- Resolve to a value. Example: 2 + 3;

- JavaScript expression categories:
  - **Arithmetic:** Evaluates a number, for example, 3.14159.
  - **String:** Evaluates a character string, for example, "Fred" or "234".
  - **Logical:** Evaluates to true or false.
  - **Primary expression:** Basic keywords and general expressions in JavaScript.(this)
  - **Left-hand-side expressions:** Left values are the destination of an assignment. (new)

3.3 Components

3.3.1 Components

- React component is the building block of a React application.
- A React component represents a small chunk of the user interface in a webpage.
- The primary job of a React component is to render its user interface and update it whenever its internal state is changed.
- In addition to rendering the UI, it manages the events belonging to its user interface.
- React component provides the below functionalities:
  - Initial rendering of the user interface.
  - Management and handling of events.
  - Updating the user interface whenever the internal state is changed.

3.3.2 Components Features

React components accomplish these features using three concepts:

- **Properties –** Enables the component to receive input.
- **Events –** Enable the component to manage DOM events and end-user interaction.
- **State –** Enable the component to stay stateful. A stateful component updates its UI with respect to its state.

3.3.3 Types of React Components

React library has two component types. The types are categorized based on the way it is being created.

- ES6 class component – Uses ES6 class.
- Function component – Uses plain JavaScript function.

3.4 Create a Class Component

3.4 Create a Class Component

Outline:

To create a class component for an expense manager app. Expense Entry Item to showcase an expense entry item. Expense entry item consists of name, amount, date, and category.

Steps:

- Create a new react app.

```
C:\workspace\react tutorial\create react app>create-react-app class-component

Creating a new React app in C:\workspace\react tutorial\create react app\class-component.
```

3.4 Create a Class Component

Steps:

```
Success! Created class-component at C:\workspace\react tutorial\create react app\class-component
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can’t go back!

We suggest that you begin by typing:

  cd class-component
  npm start

Happy hacking!
```

3.4 Create a Class Component

Steps:

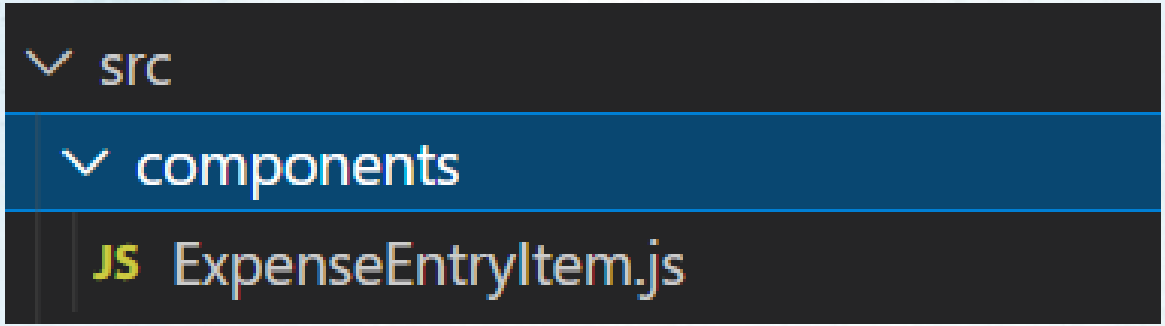
- Get into the app created and start your app.

```
C:\workspace\react tutorial\create react app>cd class-component

C:\workspace\react tutorial\create react app\class-component>npm start

> class-component@0.1.0 start
> react-scripts start
```

3.4 Create a Class Component

Steps:

- Open your app in VS code. Create a new folder “components” inside the “src” folder and create a new JavaScript file.



3.4 Create a Class Component

Steps:

- Import React, and create a class component by extending React. Component, and export the component created.

```
import React from 'react';
class ExpenseEntryItem extends React.Component {
    render() {

    }
}

export default ExpenseEntryItem;
```

3.4 Create a Class Component

Steps:

- Create the user interface using JSX and return it from the render method.

```
import React from 'react';
class ExpenseEntryItem extends React.Component {
    render() {
        return (
            <div>
                <div><b>Item:</b> <em>Mango Juice</em></div>
                <div><b>Amount:</b> <em>30.00</em></div>
                <div><b>Spend Date:</b> <em>2020-10-10</em></div>
                <div><b>Category:</b> <em>Food</em></div>
            </div>
        );
    }
}

export default ExpenseEntryItem;
```

3.4 Create a Class Component

Steps:

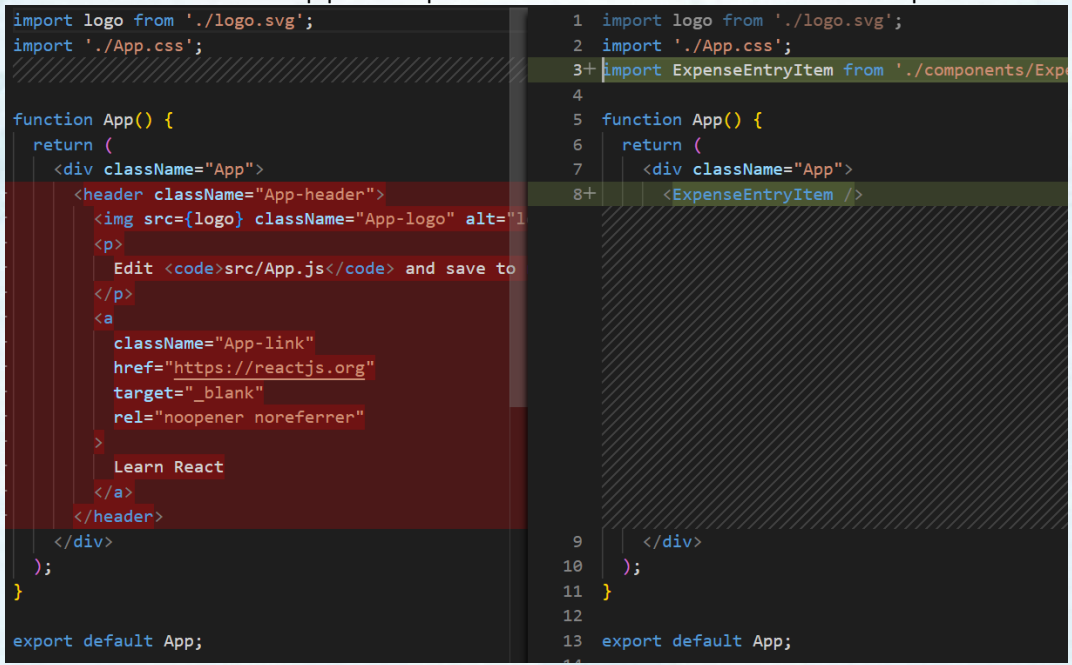
- Switch to the App.js file and import the class component created.

```
import ExpenseEntryItem from './components/ExpenseEntryItem';
```

3.4 Create a Class Component

Steps:

- Remove the content inside the App component and call the class component created.



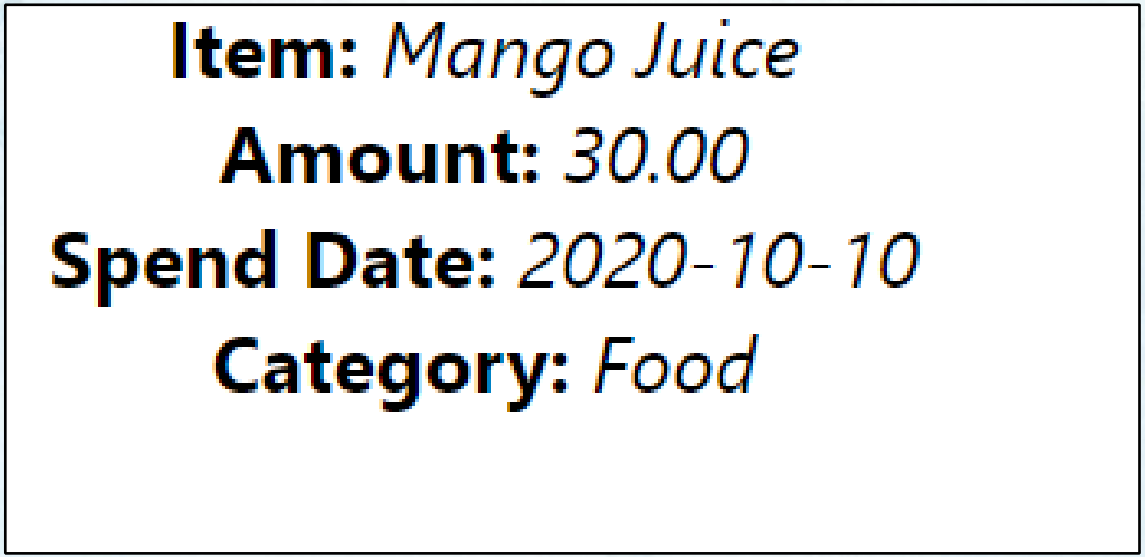
```
import logo from './logo.svg';
import './App.css';


function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="l
        <p>
          Edit <code>src/App.js</code> and save to
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```

```
1  import logo from './logo.svg';
2  import './App.css';
3+ import ExpenseEntryItem from './components/Exp
4
5  function App() {
6    return (
7      <div className="App">
8+       <ExpenseEntryItem />
9      </div>
10   );
11 }
12
13 export default App;
```

3.4 Create a Class Component

Steps:

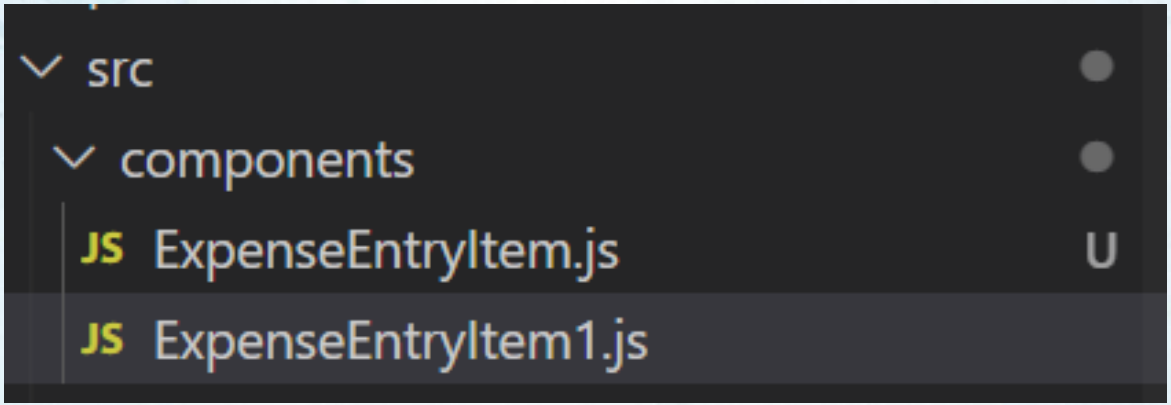
- Check the output:



3.5 Function Component

3.5.1 Function Component

- React components can also be created using plain JavaScript functions but with limited features.
- Let's create a second entry item as a functional component.
- Create a new file inside the folder "components".



3.5.1 Function Component

- Create a simple function that returns the JSX and export the component.

```
function ExpenseEntryItem1() {
    return (
        <div>
            <div><b>Item:</b> <em>Apple Juice</em></div>
            <div><b>Amount:</b> <em>50.00</em></div>
            <div><b>Spend Date:</b> <em>2020-10-10</em></div>
            <div><b>Category:</b> <em>Food</em></div>
        </div>
    );
}

export default ExpenseEntryItem1;
```

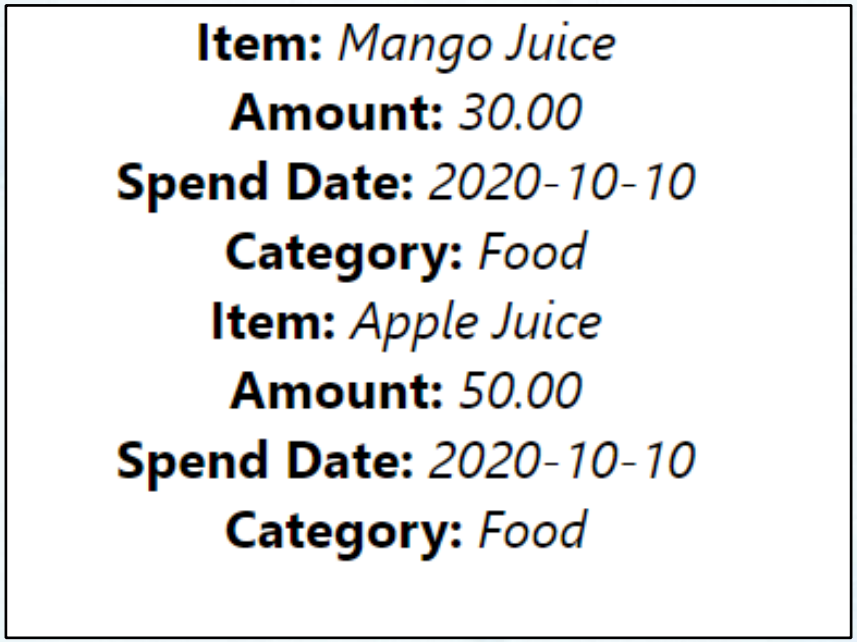
3.5.1 Function Component

- Switch to the *App.js* file and import the function component created and call the component inside the App function.

```
import logo from './logo.svg';
import './App.css';
import ExpenseEntryItem from './components/ExpenseEntryItem';
import ExpenseEntryItem1 from './components/ExpenseEntryItem1';

function App() {
  return (
    <div className="App">
      <ExpenseEntryItem />
      <ExpenseEntryItem1 />
    </div>
  );
}

export default App;
```

3.5.1 Function Component

- Check the output:



3.5.2 Difference Between Class and Functional Components

- Function components are very minimal in nature. It's only requirement is to return a React element.
- Class components support state management out of the box whereas function components do not support state management. But, React provides a hook, useState() for the function components to maintain its state.
- Class component have a life cycle and access to each life cycle event through dedicated callback APIs. The function component does not have a life cycle. Again, React provides a hook, useEffect() for the function component to access different stages of the component.

3.6 Styling and CSS

3.6 Styling and CSS

CSS can be added by using two methods:

- Inline style.
- CSS classes to components.

Inline style:

- The style attribute accepts a JavaScript object with camelCased properties rather than a CSS string.

```
const divStyle = {
  color: 'blue',
  backgroundImage: 'url(' + imgUrl + ')',
};

function HelloWorldComponent() {
  return <div style={divStyle}>Hello World!</div>;
}
```

3.6 Styling and CSS

- React will automatically append a “px” suffix to certain numeric inline style properties. If you want to use units other than “px”, specify the value as a string with the desired unit.

```
// Result style: '10px'
<div style={{ height: 10 }}>
  Hello World!
</div>

// Result style: '10%'
<div style={{ height: '10%' }}>
  Hello World!
</div>
```

3.6 Styling and CSS

CSS Classes to components:

- Pass a string as the *className* prop.

```
render() {
  return <span className="menu navigation-menu">Menu</span>
}
```

3.7 Enhance your Expense Manager App

3.7 Enhance your Expense Manager App

- Switch to the App.css file and remove all the existing style classes.
- Add a class name in the function and class component created.

```
function ExpenseEntryItem1() {
    return (
        <div className="expense-item-container">
```

```
import React from 'react';
class ExpenseEntryItem extends React.Component {
    render() {
        return (
            <div className="expense-item-container">
```

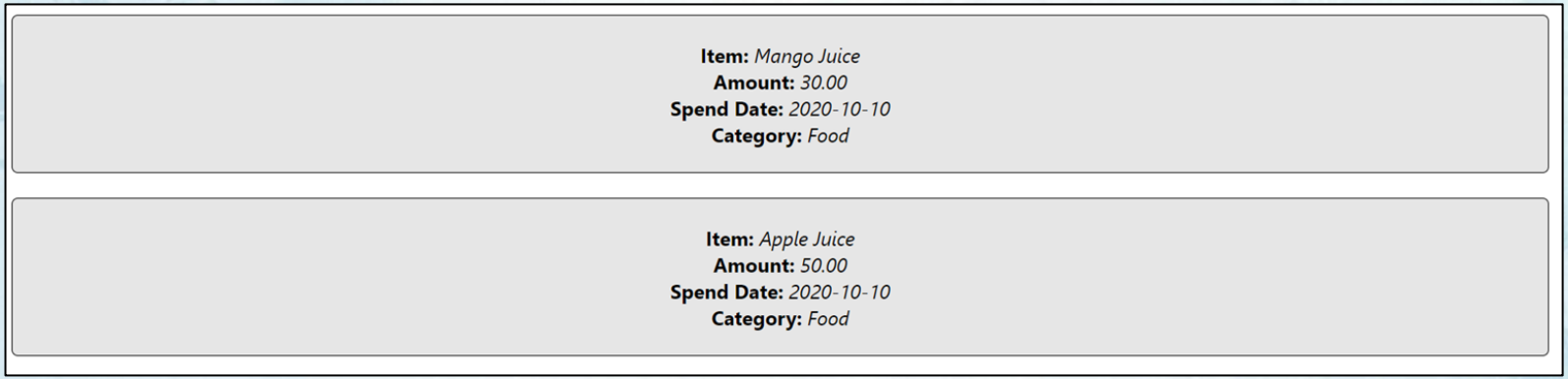
3.7 Enhance your Expense Manager App

- Define the style for the class in App.css, you can add your own styles to make the entry item attractive.

```
.expense-item-container {
  background-color: #e6e6e6;
  margin: 20px;
  padding: 20px;
  border: 1px solid #000000;
  border-radius: 5px;
  text-align: center;
}
```

3.7 Enhance your Expense Manager App

- Verify the output:



3.7 Enhance your Expense Manager App

- We know that both components are called inside the in the App.js file.

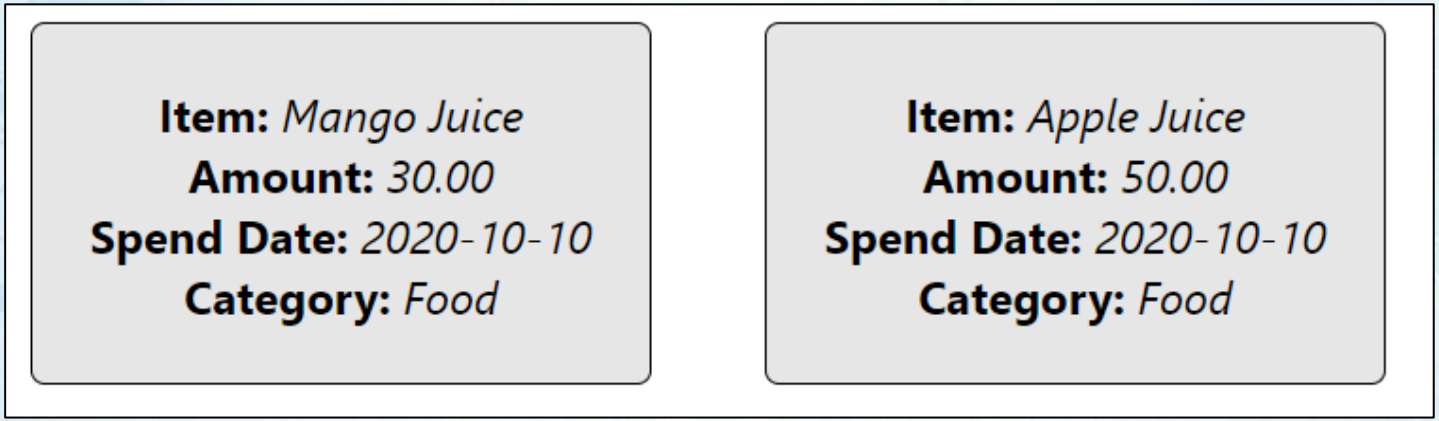
```
<div className="App">
  <ExpenseEntryItem />
  <ExpenseEntryItem1 />
</div>
```

- We can add styles to the *App* class to make the webpage more attractive.

```
.App {
  display: flex;
}
```

3.7 Enhance your Expense Manager App

- Verify the output:

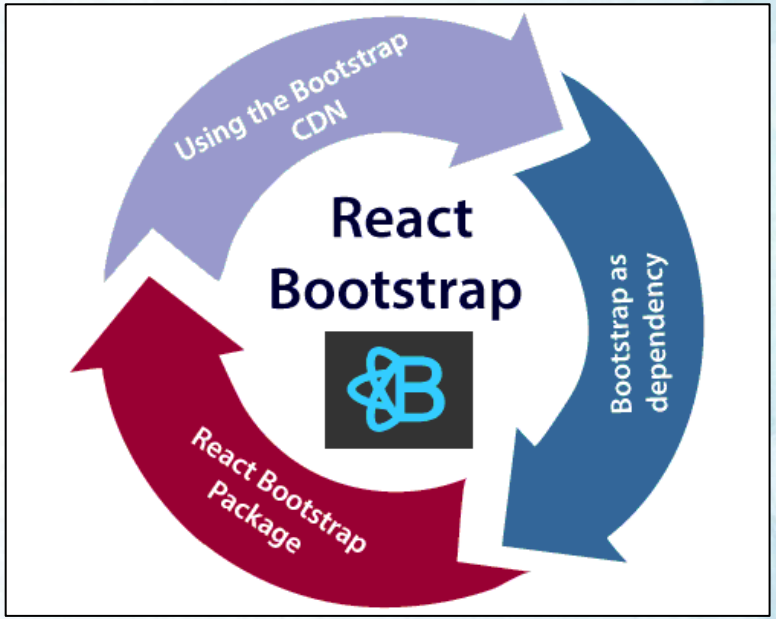


3.8 Bootstrap in React

3.8 Bootstrap in React

There are multiple ways to add bootstrap in react project.

- Using bootstrap CDN
- Installing bootstrap dependency
- Using react bootstrap packages



3.8 Bootstrap in React

Using bootstrap CDN

- This is the simplest way to add bootstrap. Like other CDN, we can add bootstrap CDN in index.html of the react project.

- CDN URL:
  - <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
  - <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
- The class names and their description can be referred from the Link.

3.8 Bootstrap in React

Installing bootstrap dependency

- Bootstrap dependency can be added to the local by using the following commands:

  npm install bootstrap

  npm install jquery popper.js

- We can add these through import in the *index.js* file of react project.

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

- The class names and their description can be referred from the Link.

3.8 Bootstrap in React

Using React bootstrap packages

There are two options to include react bootstrap packages

- react-bootstrap
- reactstrap

3.8 Bootstrap in React

React-bootstrap

- Install it with npm.

  npm install –save react-bootstrap

- This package currently serves bootstrap 3 only. Once installed include them in App.JSX file.
- Once installed we can directly use the bootstrap components in any components file. Where the component and their description can be referred from the link.

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

3.8 Bootstrap in React

reactstrap

- Install it with npm.

  npm install –save reactstrap

- reactstrap supports the bootstrap 4 version which means it is more latest than react-bootstrap
- We can import the components from reactstrap which are similar to other react components. Where the component and their description can be referred from the link.

```
import {
  Button,UncontrolledAlert,Card,CardImg,CardBody,
  CardTitle,CardSubtitle,CardText
} from'reactstrap';
```

3.8 Bootstrap in React

Other important CSS Frameworks

Not only bootstrap you can include any CSS Frameworks with React. Some of the interesting CSS Frameworks links are given below:

- Material UI - Link
- Tailwind - Link

3.9 Assignment

3.9 Assignment

Outline:

Take your Expense Manager app to next level by styling them with bootstrap. You can use either of the ways to add bootstrap in React.