**React JS**

This course will help you create the user interface of web applications where, when the application data changes, UI updates happen automatically and efficiently. Also helps understanding its best features like Virtual DOM, Components, JSX, Lifecycle methods, Isomorphism, and so on

**By the end of the course, you will be able to:**

* Create user interface of application through interactive components
* Build components using the special syntax of JavaScript (JSX)
* Handle data efficiently using state and props
* Describe lifecycle phases and implement hooks
* Performing Http get, post, put and delete requests
* Configure routes in the application
* Create Single Page Application
* Style React application using CSS, bootstrap, react-bootstrap and Material UI
* Manage state efficiently using Redux architecture

Angular is one of the popular frameworks used for UI development. Below is the comparison of React with Angular:

| **React** | **Angular** |
| --- | --- |
| React is a small view library | Angular is a full framework |
| React covers only the rendering and event handling part | Angular provides the complete solution for front-end development |
| Presentation code in JavaScript powered by JSX | Presentation code in HTML embedded with JavaScript expressions |
| React's core size is smaller than Angular, so bit fast | Angular being a framework contains a lot of code, resulting in longer load time |
| React is very flexible | Angular has less flexibility |
| Great performer, since it uses Virtual DOM | Angular uses actual DOM which affects its performance |

About React JS: -

React JS was created by Jordan Walke, a software engineer at Facebook and open sourced to the world by Facebook and Instagram.

ReactJS, a JavaScript library for creating user interfaces, makes the development of UI components easy and modular.

Key Features of React JS

The key features of React are:

* **Component-based architecture:** A Component is the smallest unit in a React application. Anything that is to be rendered on the browser can be rendered through components. Components help in maintainability and re-usability.
* **Virtual DOM:** React uses virtual DOM concept for DOM manipulation which improves the performance of the application.
* **Unidirectional data flow:**  React’s one-way data flow (also called one-way binding) keeps everything modular and fast and easy to debug.
* **JSX syntax:** React used JSX syntax which is similar to XML and HTML syntax that makes it easy for writing markup and binding events in components.
* **SEO performance:** The SEO performance can be improved using the server-side rendering concept.  Isomorphic applications can be developed by using React which increases the SEO performance.

Introduction To React: -

1. **What is React**  
   1. React JS is a JavaScript library for creating user interfaces  
   2. ReactJS Optimizes the DOM manipulation by writing very simple code  
   3. React is used for applications where the data keeps changing very frequently
2. **Features​​​​​​​**  
   1. Component-based architecture  
   2. Virtual DOM  
   3. Unidirectional data flow  
   4. JSX   
   5. SEO performance

The **create-react-app** tool can be used as it provides a modern build setup allowing to create and run React applications with minimal configuration. The create-react-app is a command-line interface (CLI).

To install create-react-app, following are the steps:

1. Install node.js with version 14+ from Software House or Node.js official site.

You can check the node version installed in your machine by running the following command that displays the node version as follows: node -v

2. Installs create-react-app by running the following command:

1. npm install -g create-react-app

3. Once the installation is done, create a React app using the below command:

1. create-react-app my-app

​​​​​​​​​​​​​​If you do not want to install create-react-app then you can use the below command to create a React application

1. D:\>npx create-react-app my-app

Below is the description of the folder structure:

|  |  |
| --- | --- |
| **Files** | **Purpose** |
| node\_modules | All the node module dependencies are created in this folder |
| public | This folder contains the public static assets of the application |
| public/index.html | This is the first page that gets loaded when you run the application |
| src | All application related files/folders are created in this folder |
| src/index.js | This is the entry point of the application |
| package.Json | Contains the dependencies of the React application |

To Run the application, need to run “npm start” command in command line interface.

Your project does not have a start script in its [package.json](vscode-file://vscode-app/c:/Users/CHAKRAVARTHY%20TUMMA/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ").  
To run your React app, use the dev script instead:

-- npm run dev

Summary: React Components

1. React allows developers to create 2 types of components  
     a. Class components  
     b. Functional components  
2. Functional components are javascript functions which return HTML elements using React.createElement method.  
3. The component can be rendered to the DOM using the ReactDOM.render method.

In React creating form by using JS and JSX

Let us understand the need of JSX by taking the below example:

Code using pure JavaScript:

1. function App() {
2. return (
3. React.createElement('form', {},
4. React.createElement("h1", {}, "Login"),
5. React.createElement('input', {type: 'text',placeholder:'Name',
6. value:'',}),
7. React.createElement('br', {}),React.createElement('br', {}),
8. React.createElement('input', {type: 'password', placeholder:'password',
9. value:'',}),
10. React.createElement('br', {}), React.createElement('br', {}),
11. React.createElement('button', {type: 'submit'}, "Login"))
12. )
13. }
14. export default App;

From the above code, it is observed that more lines of JavaScript code need to be written to implement the Login component. The Code looks difficult to understand and hence productivity goes down. JSX has been introduced in React to create elements that are very easy to read and write, which makes the component's code simple and understandable. The above Login component can be written using JSX in an easier way:

1. function App() {
2. return (<form><h2>Login</h2>
3. <input type="text" placeholder="Name" /><br/><br/>
4. <input type="password" placeholder="password" /> <br/><br/>
5. <input type="submit" nvalue="log" />
6. </form>);
7. }
8. export default App;

**How to create React JS element?**

While working with React components, multiple React elements are required to be rendered. For example, the below code needs to be render multiple JSX elements:

function App() {

    return

    <h3>ReactJS:</h3>

     <img src="./image/react.PNG" width="120" height="120"/>

    <p> React is a JavaScript library for creating User Interfaces.</p>

}

export default App;

The above code logs an error saying "Adjacent JSX elements must be wrapped in an enclosing tag".  As per the JSX syntax, all the adjacent elements must be wrapped in an enclosing tag i.e., there should be only one outermost element. Hence, the above code can be modified as follows:

function App() {

  return (

    <div>

      <h3>ReactJS:</h3>

      <img src="./image/react.PNG" width="120" height="120" />

      <p> React is a JavaScript library for creating User Interfaces.</p>

    </div>

  );

}

export default App;  
  
**React Fragments**

By adhering to JSX syntax, the <div> tag can be used for grouping the elements and this introduces an extra and unnecessary node into the DOM. As a solution to this, React Fragments are introduced which is a common pattern in React for a component to return multiple elements. React Fragments allows to group a list of React elements without adding any extra node to the DOM.

Modifying the previous code as follows using React Fragments:

function App() {

  return (

    <React.Fragment>

      <h3>ReactJS:</h3>

      <img src="./image/react.PNG" width="120" height="120" />

      <p> React is a JavaScript library for creating User Interfaces.</p>

    </React.Fragment>

  );

}

export default App;

**Empty Tags**

You can use empty tags instead of React. Fragment.

function App() {

  return (

    <>

      <h3>ReactJS:</h3>

      <img src="./image/react.PNG" width="120" height="120" />

      <p> React is a JavaScript library for creating User Interfaces.</p>

    </>

  );

}

export default App;

Fragments can be passed with key attributes if required. Empty tags don’t support any attributes.

**Highlights:**

1. Creating elements in JSX
2. How JSX code will be converted to JavaScript

**Demo steps:**

1. Modify the file App.js by writing the following code using JSX:

1. function App () {
2. return <h1>Hello World</h1>
3. }
4. export default App;

Render App component in index.js file as shown below:

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

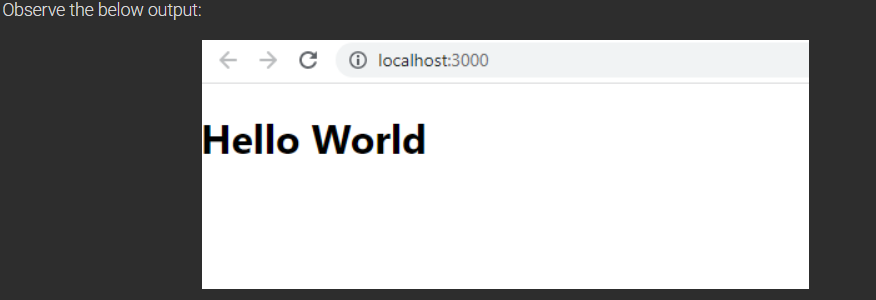
const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<App/>

);

reportWebVitals();

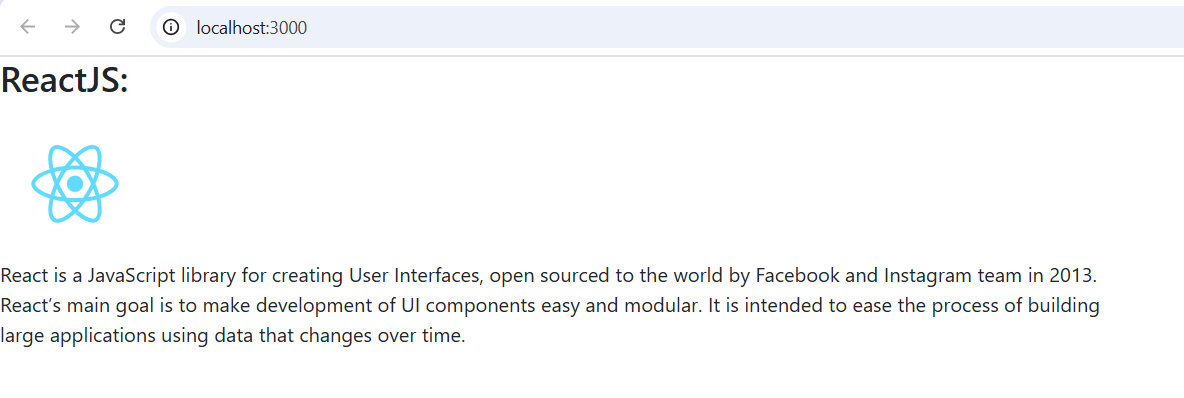


**Highlights:**

1. How to render multiple elements
2. Nesting JSX elements

**Demo steps:**

Create a component to render an image with its description as follows:



1. Modify App.js as shown below:

function App() {

  return (

    <>

      <h3>ReactJS:</h3>

      <img src={logo} width="120" height="120" alt='' />

      <p>

        {" "}

        React is a JavaScript library for creating User Interfaces, open sourced

        to the world by Facebook and Instagram team in 2013.

        <br />

        React’s main goal is to make development of UI components easy and

        modular. It is intended to ease the process of building

        <br /> large applications using data that changes over time.

      </p>

    </>

  );

}

To use an image, the image must be imported and the imported value should be provided to the src tag as an expression.

The logo.svg used in the component would be present in the src folder by default.

2. Render the App Component in the index.js file as shown below:

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

**JavaScript Expressions in JSX**

​So far, you have learnt, how to use JSX to create React elements, but there are times when JavaScript expressions also is required in React elements. Therefore, let's understand, how to write JavaScript expressions in JSX.

JavaScript expressions to be evaluated should be wrapped within curly braces as follows:

<h1> { Expression to be evaluated } </h1>

Content which must be displayed as is will be written in double quotes, wrapped within curly braces as follows:

<h1> {" Content to be displayed "} </h1>

**Accessing a variable:**

The values of any variable can be accessed within curly braces as shown below:

**Usage of map function to render JSX elements**map() method in React

The map() function in React is used to iterate over an array and return a new array with the results of the function call. The function passed to map() is called once for each element in the array, and the result of the function call is used as the element in the new array.

For example, the following code creates an array of numbers and then uses map() to double the values in the array:

const numbers = [1, 2, 3, 4, 5];

const doubled = numbers.map((number) => number \* 2);

console.log(doubled); // [2, 4, 6, 8, 10]

The map() function can also be used to iterate over objects and return a new array with the values of the object's properties. For example, the following code creates an object with some properties and then uses map() to return an array with the values of the properties:

const object = {

name: "John Doe",

age: 30,

address: "123 Main Street"

};

const properties = Object.keys(object).map((property) => object[property]);

console.log(properties); // ["John Doe", 30, "123 Main Street"]

The map() function is a powerful tool that can be used to iterate over data and transform it into a new form. It is a commonly used function in React components, and it can be used to create a variety of different UI elements.

**Here are some additional tips for using map() in React components:**

Always use a unique key for each element in the array that is being mapped. This will help React to update the UI correctly when the data changes.

If the function passed to map() is too complex, it may be a good idea to extract it into a separate component. This will make the code more readable and maintainable.

Use map() to create reusable components that can be used in multiple places in your application. This will help to keep your code DRY (not repetitive).

Map Sample Example:

function App() {

var employeeList = [

    { id: 101, name: "Anil", salary: 50000 },

    { id: 102, name: "Sunil", salary: 60000 },

    { id: 103, name: "Kiran", salary: 70000},

  ];

  return (

    <>

<table className='table table-bordered table-striped'>

        <thead className='bg-dark text-white text-center'>

          <tr>

            <th>Emp ID</th>

            <th>Emp Name</th>

            <th>Emp Salary</th>

          </tr>

        </thead>

        <tbody>

          {

            employeeList.map( (emp) => (

              <tr key={emp.id} >

                <td>{emp.id}</td>

                <td>{emp.name}</td>

                <td>{emp.salary}</td>

              </tr>

            ))

          }

        </tbody>

      </table>

</>

}

}