## **1. ReactJS-HOL Objectives**

* Define SPA and its benefits
* Define React and identify its working
* Identify the differences between SPA and MPA
* Explain Pros & Cons of Single-Page Application
* Explain about React
* Define virtual DOM
* Explain Features of React

In this hands-on lab, you will learn how to:

* Set up a react environment
* Use create-react-app

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **30 minutes.**

Create a new React Application with the name “myfirstreact”, Run the application to print “welcome to the first session of React” as heading of that page.

1. To create a new React app, Install Nodejs and Npm from the following link:

<https://nodejs.org/en/download/>

1. Install Create-react-app by running the following command in the command prompt:



1. To create a React Application with the name of “myfirstreact”, type the following command:



1. Once the App is created, navigate into the folder of myfirstreact by typing the following command:



1. Open the folder of myfirstreact in Visual Studio Code
2. Open the App.js file in Src Folder of myfirstreact
3. Remove the current content of “App.js”
4. Replace it with the following:

A black and white text

AI-generated content may be incorrect.

1. Run the following command to execute the React application:



1. Open a new browser window and type “localhost:3000” in the address bar

A screenshot of a computer

AI-generated content may be incorrect.

**Implementation :**

**1. What is SPA (Single Page Application) and what are its benefits?**

A Single Page Application (SPA) is a type of web application that loads a single HTML page and dynamically updates the content as the user interacts with the app, without refreshing the whole page.

Benefits of SPA:

* Faster navigation: No need to reload the whole page every time.
* Better user experience: Feels smoother, like using a mobile app.
* Efficient data handling: Only loads the data that’s needed, not full pages.
* Reduces server load: Since fewer full-page requests are made.

**2. What is React and how does it work?**

React is a JavaScript library (created by Facebook) for building user interfaces, especially SPAs.

How it works:

* React builds UI using components — small, reusable blocks like buttons, cards, etc.
* It manages updates using a smart concept called the Virtual DOM.
* Instead of directly changing the real HTML, React creates a virtual copy (Virtual DOM) and figures out the difference (called diffing) to make updates fast and efficient.

**3. What is the difference between SPA and MPA (Multi Page Application)?**

|  |  |  |
| --- | --- | --- |
| **Feature** | **SPA** | **MPA** |
| Page Load | Loads once | Loads on every navigation |
| Navigation | Fast (no full reload) | Slower (page reloads happen) |
| User Experience | Smooth and app-like | Traditional and sometimes slower |
| Development Focus | Frontend-heavy (JS, APIs) | Backend-heavy (PHP, ASP.NET) |
| SEO | More complex | Easier |

**4. What are the Pros and Cons of Single Page Applications?**

**Pros:**

* Fast navigation and better performance after the first load
* Great for mobile or app-like experiences
* Seamless user interaction without page reloads

**Cons:**

* Not ideal for SEO (search engine indexing is harder)
* Initial load time can be slow
* May require more frontend logic and complexity

**5. Tell me more about React**

React is all about building user interfaces with **components** — you break your UI into small pieces and manage each independently.

* Reusability with components
* Super fast updates with Virtual DOM
* Easy to manage complex UI with state and props
* Large community and lots of libraries

**6. What is Virtual DOM in React?**

The Virtual DOM is a lightweight copy of the real DOM. React uses it to figure out what actually changed on the page and only updates that specific part — rather than reloading the whole page.

This makes React apps much faster and more efficient.

**7. Key Features of React**

* **Component-Based**: UI is built with small, reusable components
* **Virtual DOM**: Fast rendering and updates
* **Unidirectional Data Flow**: Data moves in one direction, which makes it easier to debug
* **JSX Syntax**: Allows HTML and JavaScript to work together smoothly
* **Rich Ecosystem**: With tools like Redux, React Router, etc.
* **Community Support**: Tons of open-source tools and learning resources

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## **2. ReactJS-HOL Objectives**

* Explain React components
* Identify the differences between components and JavaScript functions
* Identify the types of components
* Explain class component
* Explain function component
* Define component constructor
* Define render() function

In this hands-on lab, you will learn how to:

* Create a class component
* Create multiple components
* Render a component

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **30 minutes.**

Create a react app for Student Management Portal named StudentApp and create a component named Home which will display the Message “Welcome to the Home page of Student Management Portal”. Create another component named About and display the Message “Welcome to the About page of the Student Management Portal”. Create a third component named Contact and display the Message “Welcome to the Contact page of the Student Management Portal”. Call all the three components.

1. Create a React project named “StudentApp” type the following command in terminal of Visual studio:



1. Create a new folder under Src folder with the name “Components”. Add a new file named “Home.js”
2. Type the following code in Home.js

A black screen with text

AI-generated content may be incorrect.

1. Under Src folder add another file named “About.js”
2. Repeat the same steps for Creating “About” and “Contact” component by adding a new file as ”About.js”, “Contact.js” under “Src” folder and edit the code as mentioned for “Home” Component.
3. Edit the App.js to invoke the Home, About and Contact component as follows:

A screen shot of a computer program

AI-generated content may be incorrect.

1. In command Prompt, navigate into StudentApp and execute the code by typing the following command:



1. Open browser and type “localhost:3000” in the address bar:

A screenshot of a computer

AI-generated content may be incorrect.

**Implementation :**

**1. Explain React components**

React components are the building blocks of a React application. A component is essentially a reusable piece of code that defines how a part of the user interface should look and behave.Each component can manage its own state and props and can be reused multiple times within the application. Components help in organizing the UI into independent, modular pieces, which makes code easier to maintain and debug.

**2. Identify the differences between components and JavaScript functions**

|  |  |  |
| --- | --- | --- |
| **Feature** | **React Components** | **JavaScript Functions** |
| Purpose | Used to build user interfaces | General purpose reusable code blocks |
| Output | Returns JSX (UI elements) | Returns any data or performs an operation |
| Lifecycle Methods | Have access to lifecycle hooks (in classes) | No built-in lifecycle methods |
| JSX | Can contain JSX and render UI | Does not natively support JSX |
| Usage in React | Part of React rendering logic | May be used within or outside components |

**3. Identify the types of components**

React supports two main types of components:

* Class Components: Traditional way of writing components using ES6 classes. These components can have lifecycle methods and maintain internal state.
* Function Components: Modern and simpler way using JavaScript functions. With the introduction of Hooks, function components can now use state and other React features previously available only in class components.

**4. Explain Class Component**

A class component is a React component defined using a JavaScript class. It must extend React.Component and define a render() method that returns JSX. It can also use lifecycle methods and state.

Example:

import React, { Component } from 'react';

class Welcome extends Component {

render() {

return <h1>Hello, {this.props.name}</h1>;

}

}

Key Features:

* Uses ES6 class syntax
* Can hold and manage state
* Can use lifecycle methods like componentDidMount(), componentDidUpdate(), etc.

**5. Explain Function Component**

A function component is a simpler way to write components using JavaScript functions. These components take props as input and return JSX.

Example:

function Welcome(props) {

return <h1>Hello, {props.name}</h1>;

}

With React Hooks (like useState, useEffect), function components can now handle state and side effects, making them almost equivalent to class components in capability.

**6. Define Component Constructor**

The constructor is a special method in a class component used to:

* Initialize the component's state
* Bind event handler methods to the component

It's the first method that runs when the component is created.

Example:

class Welcome extends React.Component {

constructor(props) {

super(props);

this.state = { message: 'Welcome!' };

}

}

You must always call super(props) inside the constructor to access this.props in the constructor.

**7. Define render() Function**

The render() function is **mandatory** in class components. It describes what the UI should look like.

* It returns **JSX**, which React then renders to the DOM.
* The render() method should be **pure**, meaning it should return the same output each time it's called with the same inputs.

Example:

class Welcome extends React.Component {

render() {

return <h1>Hello, World!</h1>;

}

}

The render() function does not modify the DOM directly. Instead, React uses the output to update the Virtual DOM and then efficiently update the real DOM.

A computer screen shot of a program code

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## **3. ReactJS-HOL** **Objectives**

* Explain React components
* Identify the differences between components and JavaScript functions
* Identify the types of components
* Explain class component
* Explain function component
* Define component constructor
* Define render() function

In this hands-on lab, you will learn how to:

* Create a function component
* Apply style to components
* Render a component

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **30 minutes.**

Create a react app for Student Management Portal named scorecalculatorapp and create a function component named “CalculateScore” which will accept Name, School, Total and goal in order to calculate the average score of a student and display the same.

1. Create a React project named “scorecalculatorapp” type the following command in terminal of Visual studio:



1. Create a new folder under Src folder with the name “Components”. Add a new file named “CalculateScore.js”
2. Type the following code in CalculateScore.js

A black screen with white text

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

1. Create a Folder named Stylesheets and add a file named “mystyle.css” in order to add some styles to the components:

A screenshot of a computer program

AI-generated content may be incorrect.

1. Edit the App.js to invoke the CalculateScore functional component as follows:

A screen shot of a computer

AI-generated content may be incorrect.

1. In command Prompt, navigate into scorecalculatorapp and execute the code by typing the following command:



1. Open browser and type “localhost:3000” in the addressA screen shot of a computer

   AI-generated content may be incorrect.

**Implementation :**

**1.Explain React Components**

React components are the core building blocks of a React application. They are like small, self-contained pieces of code that manage their own logic and UI. Each component can be thought of as a section of the webpage — for example, a header, a button, or a form. Components help developers break down complex UIs into smaller, manageable parts, making development more organized and scalable.

**2. Identify the Differences Between Components and JavaScript Functions**

While both components and JavaScript functions can contain logic and take inputs, they serve different purposes:

|  |  |  |
| --- | --- | --- |
| **Feature** | **React Components** | **JavaScript Functions** |
| Purpose | Build UI elements | Perform calculations or reusable logic |
| Returns | JSX (HTML-like structure) | Data values or logic results |
| Integration | Tightly coupled with React UI rendering | Independent, used anywhere in JavaScript |
| State Management | Can manage internal state (if React component) | No concept of state in vanilla JS functions |
| React Hooks | Used only inside React function components | Not applicable |

**3. Identify the Types of Components**

React has two main types of components:

* **Class Components**: Created using ES6 classes, capable of using lifecycle methods and managing state.
* **Function Components**: Defined using regular JavaScript functions. Initially used for simple UI, but with React Hooks, they can now handle state and side effects too.

**4. Explain Class Component**

A class component is a way to define a component using JavaScript classes. It allows for more advanced features like managing **state** and using **lifecycle methods**. It must include a render() method that returns JSX to define what the UI should look like.

Example:

import React, { Component } from 'react';

class Greeting extends Component {

render() {

return <h1>Hello, {this.props.name}</h1>; }}

Key characteristics:

* Inherits from React.Component
* Must include a render() method
* Can manage its own state
* Can respond to lifecycle events

**5. Explain Function Component**

A function component is a simpler, more concise way to create components. It’s a plain JavaScript function that takes props as an argument and returns JSX.

Example:

function Greeting(props) {

return <h1>Hello, {props.name}</h1>;

}

Function components are now widely used because they are cleaner and, with the introduction of Hooks (like useState, useEffect), they can do almost everything class components can.

**6. Define Component Constructor**

The constructor method is used only in class components. It runs when the component is first created and is typically used to:

* Initialize state
* Bind methods to the class instance

Example:

constructor(props) {

super(props);

this.state = { count: 0 };

}

You must call super(props) to access this.props in the constructor.

**7. Define render() Function**

The render() function is required in every class component. It defines what gets displayed on the screen. It returns JSX, which React uses to update the virtual DOM and, eventually, the actual webpage.

Example:

render() {

return <p>Welcome to React!</p>;

}

It should not contain complex logic or side effects — just return the view.

A screenshot of a computer

AI-generated content may be incorrect.

Bottom of Form

A computer screen shot of a black screen

AI-generated content may be incorrect.A computer screen shot of a program

AI-generated content may be incorrect.

A computer screen shot of a black screen

AI-generated content may be incorrect.A computer screen shot of a program

AI-generated content may be incorrect.

**4. ReactJS-HOL  
Objectives**

* Explain the need and Benefits of component life cycle
* Identify various life cycle hook methods
* List the sequence of steps in rendering a component

In this hands-on lab, you will learn how to:

* Implement componentDidMount() hook
* Implementing componentDidCatch() life cycle hook.

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **60 minutes.**

1. Create a new react application using *create-react-app* tool with the name as “blogapp”
2. Open the application using VS Code
3. Create a new file named as **Post.js** in **src folder** with following properties

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 2: Post class

1. Create a new class based component named as **Posts** inside **Posts.js** file

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 3: Posts Component

1. Initialize the component with a list of Post in state of the component using the constructor
2. Create a new method in component with the name as **loadPosts()** which will be responsible for using Fetch API and assign it to the component state created earlier. To get the posts use the url (<https://jsonplaceholder.typicode.com/posts>)

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 4: loadPosts() method

1. Implement the **componentDidMount()** hook to make calls to **loadPosts()** which will fetch the posts

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 5: componentDidMount() hook

1. Implement the **render()** which will display the title and post of posts in html page using heading and paragraphs respectively.

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 6: render() method

1. Define a **componentDidCatch()** method which will be responsible for displaying any error happing in the component as alert messages.

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 7: componentDidCatch() hook

1. Add the Posts component to App component.
2. Build and Run the application using *npm start* command.

**Implementation :**

**1.Explain the Need and Benefits of Component Life Cycle**

The component life cycle in React refers to the series of phases a component goes through from its creation to its removal from the DOM. This includes mounting (creation), updating (re-rendering), and unmounting (deletion).

Need for Lifecycle Methods:

* To control when code runs at specific points in a component’s life (e.g., after it's rendered, when props/state change, or when it's about to be removed).
* To fetch data, set timers, or interact with external APIs only when appropriate.
* To clean up tasks like removing event listeners to avoid memory leaks.

Benefits:

* Enables efficient resource management.
* Provides greater control over what the component should do at different stages.
* Helps build more predictable and maintainable components.
* Essential for performance optimizations like preventing unnecessary renders or API calls**.**

**2. Identify Various Lifecycle Hook Methods**

React (particularly in class components) provides several lifecycle methods. These can be categorized based on the component phase:

Mounting (initial render):

* constructor(): Initializes state and binds methods.
* static getDerivedStateFromProps(): Updates state based on props before render.
* render(): Returns JSX to render UI.
* componentDidMount(): Runs once after the component is added to the DOM. Good for API calls.

Updating (on state/props change):

* static getDerivedStateFromProps(): Also called during updates.
* shouldComponentUpdate(): Controls whether the component should re-render.
* render(): Called again if re-render is allowed.
* getSnapshotBeforeUpdate(): Captures info (like scroll position) before DOM changes.
* componentDidUpdate(): Runs after re-render and DOM updates.

Unmounting (component removal):

* componentWillUnmount(): Clean up tasks like clearing timers or subscriptions.

Error Handling:

* componentDidCatch(): Catches errors in child components.

**3. List the Sequence of Steps in Rendering a Component**

Here is the typical order in which lifecycle methods are called during the initial render and updates:

Initial Mounting Sequence (Component Created):

1. constructor()
2. getDerivedStateFromProps()
3. render()
4. componentDidMount()

Updating Sequence (Component Re-rendered due to state/props change):

1. getDerivedStateFromProps()
2. shouldComponentUpdate()
3. render()
4. getSnapshotBeforeUpdate()
5. componentDidUpdate()

Unmounting Sequence (Component Removed):

1. componentWillUnmount()

**5. ReactJS-HOL  
Objectives**

* Understanding the need for styling react component
* Working with CSS Module and inline styles

In this hands-on lab, you will learn how to:

* Style a react component
* Define styles using the CSS Module
* Apply styles to components using className and style properties

## **Prerequisites**

The following is required to complete this hands-on lab:

* Node.js
* NPM
* Visual Studio Code

## **Notes**

Estimated time to complete this lab: **30 minutes.**

My Academy team at Cognizant want to create a dashboard containing the details of ongoing and completed cohorts. A react application is created which displays the detail of the cohorts using react component. You are assigned the task of styling these react components.

Download and build the attached react application.



1. Unzip the react application in a folder
2. Open command prompt and switch to the react application folder
3. Restore the node packages using the following commands



Figure 1: Restore packages

1. Open the application using VS Code
2. Create a new CSS Module in a file called “CohortDetails.module.css”
3. Define a css class with the name as “box” with following properties

*Width = 300px;*

*Display = inline block;*

*Overall 10px margin*

*Top and bottom padding as 10px*

*Left and right padding as 20px*

*1 px border in black color*

*A border radius of 10px*

1. Define a css style for html <dt> element using tag selector. Set the font weight to 500.
2. Open the cohort details component and import the CSS Module
3. Apply the box class to the container div
4. Define the style for <h3> element to use “green” color font when cohort status is “ongoing” and “blue” color in all other scenarios.
5. Final result should look similar to the below image

A close-up of a computer screen

AI-generated content may be incorrect.

Figure 2: Final Result

Implementation:

**A screenshot of a computer

AI-generated content may be incorrect.**