**Digital Nurture 4.0 Deep Skilling**

**Week 6**

1. ReactJS-HOL

**1.Define SPA and its benefits**  
 **SPA (Single Page Application)** is a web application that loads a single HTML page and dynamically updates content without reloading the page.  
**Benefits:**

* Faster user experience
* Less server load
* Smooth navigation

**2.Define React and identify its working**  
 **React** is a JavaScript library for building user interfaces, especially single-page applications.  
It works using a **component-based** structure and efficiently updates the user interface using a **Virtual DOM**.

**3.Identify the differences between SPA and MPA**

| **Feature** | **SPA** | **MPA** |
| --- | --- | --- |
| Page reloads | No | Yes |
| Speed | Faster | Slower |
| User Experience | Smooth | Disruptive |
| Technologies | React, Angular | PHP, JSP, etc. |

**4. Explain Pros & Cons of Single-Page Application**

**Pros of SPA:**

* Faster performance after initial load since it avoids full page reloads.
* Smooth user experience with dynamic content updates.
* Efficient use of bandwidth and resources through selective data loading.
* Can work offline with caching techniques like service workers.

**Cons of SPA:**

* Poor SEO (Search Engine Optimization) since content is loaded dynamically.
* Initial loading time can be longer due to loading the entire application at once.
* More complex client-side routing and state management.
* May face issues with browser history and navigation.

5.**Explain about React**  
 React is a JavaScript library developed by Facebook for building fast and interactive UIs. It uses components, JSX, and virtual DOM to make web apps highly efficient.

6.**Define Virtual DOM**  
The **Virtual DOM** is a lightweight JavaScript representation of the actual DOM. React uses it to track changes and updates only the necessary parts of the real DOM for better performance.

7.**Explain Features of React**

* Component-based architecture
* Virtual DOM for faster rendering
* One-way data binding
* Reusable components
* Support for hooks and functional programming

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

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

**Code: App.jsx**

A screenshot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generated**

2. ReactJS-HOL

### ****1.Explain React Components****

In React, **components** are the building blocks of a React application. They let you split the UI into independent, reusable pieces that can be handled separately.

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

| **Feature** | **JavaScript Function** | **React Component** |
| --- | --- | --- |
| Purpose | Performs a task or returns a value | Returns JSX (UI) to render on the screen |
| Return Type | Returns data types like string, number | Returns JSX (React elements) |
| Naming Convention | Can be lowercase | Must start with a capital letter |
| Lifecycle Methods | Not applicable | Used in class components (e.g., componentDidMount) |
| React Integration | Not used to define UI | Used to create UI elements in React |

### ****3.Identify the Types of Components****

There are two main types of components in React:

1. **Class Components**
2. **Function Components**

### ****4.Explain Class Component****

A **class component** is a JavaScript class that extends React.Component and has a render() method which returns JSX.

Example :

import React, { Component } from 'react';

class Welcome extends Component {

render() {

return <h1>Hello from Class Component</h1>;

}

}

### ****5.Define Component Constructor****

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

* Initialize state
* Bind methods to this

It’s called before the component is mounted.

Example :

class Welcome extends React.Component {

constructor(props) {

super(props);

this.state = { message: 'Hello' };

}

}

### ****6.Define render() Function****

The **render ()** function is a required method in class components. It defines what the UI should look like by returning JSX.

Example :

render() {

return (

<div>

<h1>Welcome to React</h1>

</div>

);

}

Hands-on lab

Code : App.js

A screen shot of a computer

Description automatically generated

Home.js

A screen shot of a computer

Description automatically generated

About.js

A screenshot of a computer program

Description automatically generated

Contact.js

A screenshot of a computer program

Description automatically generated

OUTPUT :

A screenshot of a computer

Description automatically generated

3. ReactJS-HOL

### 1. Explain React Components

React components are the core building blocks of a React application. They allow developers to break down the user interface into smaller, reusable pieces. Each component is an independent unit that returns JSX (JavaScript XML) to define how a section of the UI should appear. Components help in managing complex user interfaces by making the code modular, easier to maintain, and scalable.

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

| **Feature** | **JavaScript Function** | **React Component** |
| --- | --- | --- |
| Purpose | Performs logic or returns a value | Returns UI elements (JSX) |
| Return Type | Returns primitive values or objects | Returns JSX to render in the browser |
| Naming Convention | Function name can start with lowercase | Component name must start with an uppercase letter |
| Integration with React | Not inherently part of the UI rendering | Used to build and display user interface |
| State and Lifecycle Support | Not applicable | Class and function components support state and hooks |

### 3. Identify the Types of Components

There are two main types of components in React:

* **Class Components**: Traditional React components defined using ES6 classes. These have access to lifecycle methods and internal state.
* **Function Components**: Simpler components defined as JavaScript functions. They can use React hooks (like useState, useEffect) to manage state and lifecycle features.

### 4. Explain Class Component

A class component is a React component created using an ES6 class that extends React. Component. It must include a render () method which returns the JSX to be displayed on the browser. Class components support internal state and lifecycle methods such as componentDidMount, componentDidUpdate, and componentWillUnmount.

Example:

jsx

CopyEdit

import React, { Component } from 'react';

class Welcome extends Component {

render() {

return <h1>Hello from Class Component</h1>;

}

}

### 5. Explain Function Component

A function component is a simpler way to create React components using a JavaScript function. It takes props as input and returns JSX. With the introduction of hooks, function components can now use state and other features previously only available in class components.

Example:

jsx

CopyEdit

function Welcome() {

return <h1>Hello from Function Component</h1>;

}

### 6. Define Component Constructor

The constructor in a React class component is a special method used for initializing state and binding methods. It is called automatically when the component is created. The constructor is defined inside the class and typically calls super(props) to access the base class (React. Component) constructor.

Example:

jsx

CopyEdit

class Welcome extends React.Component {

constructor(props) {

super(props);

this.state = { message: 'Hello' };

}

}

### 7. Define render () Function

The render () function is a mandatory method in class components. It returns the JSX that describes what should be displayed on the screen. React calls this method every time the component’s state or props change.

Example:

jsx

CopyEdit

render() {

return (

<div>

<h1>Welcome to React</h1>

</div>

);

}

Hands-on lab

Code :

**A screen shot of a computer

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**OUTPUT :**

**A screenshot of a computer

Description automatically generated**

4. ReactJS-HOL

### 1. Explain the Need and Benefits of Component Lifecycle

In React, the **component lifecycle** refers to the series of phases a component goes through from creation to removal. Understanding the lifecycle is essential for performing operations at specific stages such as fetching data, updating the DOM, or cleaning up resources.

**Benefits:**

* Allows precise control over component behavior at different stages.
* Enables optimization of performance by controlling re-renders.
* Helps manage side effects like API calls, event listeners, or timers.
* Provides hooks for clean-up operations to prevent memory leaks.

### 2. Identify Various Lifecycle Hook Methods

React class components offer several lifecycle methods that can be grouped based on different phases:

**Mounting (when the component is created and inserted into the DOM):**

* constructor()
* static getDerivedStateFromProps()
* render()
* componentDidMount()

**Updating (when the component is re-rendered due to changes in props or state):**

* static getDerivedStateFromProps()
* shouldComponentUpdate()
* render()
* getSnapshotBeforeUpdate()
* componentDidUpdate()

**Unmounting (when the component is removed from the DOM):**

* componentWillUnmount()

**Error Handling:**

* componentDidCatch()
* getDerivedStateFromError()

Note: In function components, similar behavior can be achieved using **React Hooks** such as useEffect().

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

When a component is rendered, the following sequence of lifecycle methods is invoked (for class components):

**Initial Mounting Phase:**

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

**Updating Phase (when props or state change):**

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

Hands-on lab

Code :

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

OUTPUT :

A screenshot of a computer

Description automatically generated

5. ReactJS-HOL

### 1. Understanding the Need for Styling React Components

Styling React components is essential for creating visually appealing, consistent, and user-friendly interfaces. Just like in traditional web development, styles in React control the layout, colors, typography, spacing, and overall look and feel of the application. In React, styling helps in:

* Enhancing user experience
* Establishing visual hierarchy
* Making components reusable and theme-friendly
* Ensuring responsiveness across devices

React offers multiple styling approaches such as traditional CSS, CSS Modules, inline styles, styled components, and third-party libraries, giving developers flexibility based on project needs.

### 2. Working with CSS Modules and Inline Styles

**CSS Modules** and **inline styles** are two common methods to style components in React:

**CSS Modules:**

* CSS Modules are locally scoped CSS files that prevent class name collisions.
* They allow styles to be applied only to specific components by using a unique class naming convention.
* Useful for large-scale applications where component isolation is critical.

Example :

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

function Button() {

return <button className={styles.primary}>Click Me</button>;

}

**Inline Styles:**

* Inline styles are defined directly within the JSX using JavaScript objects.
* They are useful for dynamic styling and when styles depend on component logic.
* CSS properties must be written in camelCase syntax.

Example :

function Heading() {

const style = {

color: 'blue',

fontSize: '24px'

};

return <h1 style={style}>Welcome</h1>;

}

Hands-on lab

Code :

A screenshot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

OUTPUT :

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated