**React.js Notes**

**1. Introduction to React.js**

* **React.js is a JavaScript library for building user interfaces.**
* **Developed and maintained by Facebook.**
* **Focuses on creating reusable UI components and provides an efficient way to manage the view layer of web applications.**

**2. Single Page Application (SPA)**

* **Definition: An SPA dynamically updates the content on a single web page without reloading the entire page.**
* **Benefits:** 
  + **Improves user experience with faster navigation.**
  + **Reduces server load as only required data is fetched.**

**3. DOM and Virtual DOM**

* **DOM (Document Object Model): Represents the structure of a web page as a tree of nodes.**
* **Virtual DOM: A lightweight copy of the DOM used by React to optimize updates by comparing the current and previous versions of the Virtual DOM (diffing).**
* **Benefits of Virtual DOM: React updates only the necessary parts of the DOM, improving performance.**

**4. React Components**

**Function Components**

* **Definition: Simple JavaScript functions that return JSX, used to define UI elements.**
* **Example:**
* **function Greeting() {**
* **return <h1>Hello, World!</h1>;**
* **}**

**Class Components**

* **Definition: ES6 classes that extend React.Component, used for more complex components.**
* **Features: Must include a render() method to return JSX.**
* **Example:**
* **class Greeting extends React.Component {**
* **render() {**
* **return <h1>Hello, World!</h1>;**
* **}**
* **}**

**5. Elements and Element Rendering**

**What is an Element?**

* **Definition: The smallest building block of React apps. Represents what you want to see on the screen.**
* **const element = <h1>Hello, React!</h1>;**

**What is Rendering?**

* **Definition: The process of converting React elements into DOM nodes and displaying them in the browser.**
* **Example: Use ReactDOM.render to render elements to the DOM.**
* **ReactDOM.render(element, document.getElementById('root'));**

**6. Creating and Rendering Elements**

* **React.createElement: A function used to create React elements.**
* **Example:**
* **const element = React.createElement('h1', null, 'Hello, React!');**
* **ReactDOM.render(element, document.getElementById('root'));**

**7. Class Component Lifecycle Methods**

**What are Lifecycle Methods?**

* **Definition: Special methods in class components that get called at different stages of a component's lifecycle.**
* **Stages:** 
  + **Mounting: constructor, componentDidMount.**
  + **Updating: shouldComponentUpdate, componentDidUpdate.**
  + **Unmounting: componentWillUnmount.**

**8. State**

**What is State?**

* **Definition: A built-in object used to store data that affects a component's rendering.**
* **Example:**
* **class Counter extends React.Component {**
* **constructor() {**
* **super();**
* **this.state = { count: 0 };**
* **}**
* **render() {**
* **return <h1>{this.state.count}</h1>;**
* **}**
* **}**

**9. Function Components and Hooks**

**What are Hooks?**

* **Definition: Special functions that let you "hook into" React features (like state and lifecycle) from function components.**
* **Example:**
* **const [count, setCount] = useState(0);**

**Common Hooks:**

1. **useState: Manages state.**
2. **const [count, setCount] = useState(0);**
3. **useEffect: Manages side effects (e.g., data fetching).**
4. **useEffect(() => {**
5. **console.log('Component mounted!');**
6. **}, []);**
7. **useContext: Consumes context values.**
8. **useReducer: Manages complex state logic.**
9. **useRef: Creates references to DOM elements or variables.**
10. **useMemo: Optimizes performance by memoizing values.**

**10. Routing with React Router DOM**

**What is Routing?**

* **Definition: The process of navigating between different views or components in a React application without reloading the page.**
* **Features of React Router DOM:** 
  + **Dynamic routing.**
  + **Easy integration with React components.**

**Common Features:**

* **<BrowserRouter>: Wraps the app to enable routing.**
* **<Routes> and <Route>: Define paths and corresponding components.**
* **<Routes>**
* **<Route path="/" element={<Home />} />**
* **<Route path="/about" element={<About />} />**
* **</Routes>**
* **<Link>: Provides navigation without reloading.**
* **<Link to="/about">About</Link>**

**11. Props**

**What are Props?**

* **Definition: Short for "Properties," used to pass data from a parent to a child component.**
* **Example:**
* **function Welcome(props) {**
* **return <h1>Hello, {props.name}!</h1>;**
* **}**

**Props Drilling:**

* **Definition: Passing props through multiple layers of components.**
* **Solution: Use Context API or Redux.**

**12. React Redux**

**What is Redux?**

* **Definition: A state management library for managing global application state.**
* **Core Concepts: Actions, Reducers, Store, Dispatch.**
* **Example:**
* **const reducer = (state = {}, action) => {**
* **switch (action.type) {**
* **case 'INCREMENT':**
* **return { count: state.count + 1 };**
* **default:**
* **return state;**
* **}**
* **};**

**13. Map Method**

**What is the Map Method?**

* **Definition: A JavaScript array method used to iterate over arrays and return a new array based on the callback function.**
* **Example:**
* **const items = ['Apple', 'Banana', 'Cherry'];**
* **const list = items.map(item => <li key={item}>{item}</li>);**

**14. API Calling with Axios**

**What is Axios?**

* **Definition: A popular library for making HTTP requests in JavaScript.**
* **Common Methods:** 
  1. **GET: Retrieve data.**
  2. **axios.get('/api/data').then(response => console.log(response.data));**
  3. **POST: Send data.**
  4. **axios.post('/api/data', { name: 'John' });**
  5. **DELETE: Remove data.**
  6. **axios.delete('/api/data/1');**
  7. **UPDATE (PUT/PATCH): Modify data.**
  8. **axios.put('/api/data/1', { name: 'Jane' });**

**15. Inline If-Else**

**What is Inline If-Else?**

* **Definition: A way to use conditional logic directly within JSX using ternary operators or &&.**
* **Example:**
* **const isLoggedIn = true;**
* **return (**
* **<div>{isLoggedIn ? <h1>Welcome back!</h1> : <h1>Please log in.</h1>}</div>**
* **);**