React Js Assignment

Introduction to React.js

Question 1: What is React.js? How is it different from other JavaScript frameworks and libraries?

=>

React.js is a JavaScript library for building user interfaces, primarily focusing on single-page applications. It uses a component-based architecture and a virtual DOM for efficient updates. Unlike frameworks like Angular or Vue, React is just the "view" layer, providing flexibility to integrate with other libraries or manage state using tools like Redux.

Question 2: Explain the core principles of React such as the virtual DOM and component-based architecture.

=>

React's core principles include the Virtual DOM, which updates the UI efficiently by minimizing real DOM changes, and a component-based architecture, enabling modular, reusable, and self-contained components. It uses a declarative approach for building predictable UIs and ensures a unidirectional data flow for easier state management.

Question 3: What are the advantages of using React.js in web development?

=>

React.js offers efficient UI updates with its Virtual DOM, reusable component-based architecture for modular development, and fast rendering. It supports a declarative syntax, seamless integration with other tools, and a strong community, making it ideal for scalable and dynamic web applications.

JSX (JavaScript XML)

Question 1: What is JSX in React.js? Why is it used?

=>

JSX (JavaScript XML) is a syntax extension for JavaScript used in React to write HTML-like code directly within JavaScript. It makes code more readable and expressive by combining UI structure and logic in one place, simplifying component creation.

Question 2: How is JSX different from regular JavaScript? Can you write JavaScript insideJSX?

=>

JSX looks like HTML but is actually JavaScript syntax that gets compiled into React elements. Unlike regular JavaScript, JSX allows embedding HTML-like code, and you can write JavaScript expressions inside JSX using {}.

Question 3: Discuss the importance of using curly braces {} in JSX expressions.

=>

In JSX, curly braces {} are used to embed JavaScript expressions within HTML-like syntax. They allow dynamic content, such as displaying variables, calling functions, or performing calculations, making components more interactive and flexible.

Components (Functional & Class Components)

Question 1: What are components in React? Explain the difference between functional components and class components.

=>

Components in React are reusable building blocks used to define and render UI elements. They manage their own structure, logic, and state.

- **Functional Components: Simplified, JavaScript functions that use hooks for state and lifecycle methods. Easier to read and write.**
- **Class Components: ES6 classes with render() method, using this.state and lifecycle methods for managing state. Less common in modern React.**

Question 2: How do you pass data to a component using props?

=>

Data is passed to a component using props by adding attributes to the component's tag in JSX. The parent component sends the data as attributes, and the child component accesses it using props.name.

```
<ChildComponent name="Madhav" />
// Access in ChildComponent
const ChildComponent = (props) => <p>{props.name}</p>;
```

Question 3: What is the role of render() in class components?

=>

In class components, the render() method is responsible for returning the JSX that defines the component's UI. It must be defined in every class component and is called automatically to update the component when its state or props change.

Props and State

Question 1: What are props in React.js? How are props different from state?

=>

Props in React.js are read-only inputs passed from a parent component to a child component, allowing data to be shared. They cannot be modified by the child component.

State is an internal data storage in a component that can be modified using setState(). Unlike props, state is managed within the component and can change over time, triggering re-renders.

Question 2: Explain the concept of state in React and how it is used to manage component data.

=>

In React, state is an object that holds data specific to a component and determines its behavior and rendering. Unlike props, state is mutable and can be updated using the setState() method. When state changes, React re-renders the component to reflect the updated data.

```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  Tabnine | Edit | Explain
  increment = () => {
    this.setState({ count: this.state.count + 1 });
  };

  Tabnine | Edit | Test | Explain | Document | Ask
  render() {
    return (
      <div>
        <p>{this.state.count}</p>
        <button onClick={this.increment}>Increment</button>
      </div>
    );
  }
}
```

Question 3: Why is this.setState() used in class components, and how does it work?

=>

In class components, this.setState() is used to update the component's state and trigger a re-render. It allows you to modify the state in a controlled way, ensuring React knows about the state change and re-renders the component with the new state values.

How it works:

- **Asynchronous: this.setState() does not immediately update the state but schedules an update. React batches multiple state updates for efficiency.**
- **Merges State: It merges the new state with the existing state, so only the specified properties are updated, and others remain unchanged.**

```
this.setState({ count: this.state.count + 1 });
```

Handling Events in React

Question 1: How are events handled in React compared to vanilla JavaScript? Explain the concept of synthetic events.

=>

In React, events are handled through JSX props (like onClick), while in vanilla JavaScript, events are handled using addEventListener. React uses Synthetic Events, which are cross-browser wrappers around native events, ensuring consistent behavior. These events are pooled for performance, meaning they are reused and cleared after the event handler executes.

Question 2: What are some common event handlers in React.js? Provide examples of onClick, onChange, and onSubmit.

=>

1. **onClick: Triggered when an element is clicked.**

```
<button onClick={() => alert('Clicked!')}>Click Me</button>
```

2. **onChange: Triggered when the value of an input element changes.**

```
<input type="text" onChange={(e) => console.log(e.target.value)} />
```

3. **onSubmit: Triggered when a form is submitted.**

```
<form onSubmit={(e) => { e.preventDefault(); console.log('Form submitted'); }}>
  <button type="submit">Submit</button>
</form>
```

Question 3: Why do you need to bind event handlers in class components?

=>

In class components, event handlers must be bound to the component instance (this) to ensure they have access to the component's context (e.g., this.setState). Without binding, this inside the handler would be undefined.

Conditional Rendering

Question 1: What is conditional rendering in React? How can you conditionally render elements in a React component?

=>

Conditional rendering in React is when elements are rendered based on a condition. You can use:

1. **if statements:**

```
if (isLoggedIn) return <h1>Welcome</h1>;
```

2. **Ternary operator:**

```
return isLoggedIn ? <h1>Welcome</h1> : <h1>Please log in</h1>;
```

3. **Logical &&:**

```
return isLoggedIn && <h1>Welcome</h1>;
```

Question 2: Explain how if-else, ternary operators, and && (logical AND) are used in JSX for conditional rendering.

=>

1. **if-else: Used in functions before the return statement to conditionally render different JSX.**

```
if (isLoggedIn) return <h1>Welcome</h1>;
else return <h1>Please log in</h1>;
```

2. Ternary operator: Inline conditional rendering within JSX.

```
return isLoggedIn ? <h1>Welcome</h1> : <h1>Please log in</h1>;
```

3. Logical &&: Renders an element only if the condition is true.

```
return isLoggedIn && <h1>Welcome</h1>;
```