Introduction to React.js

Question 1: What is React.js?

Ans.

- ✓ It is a JavaScript library created by Facebook.
- ✓ Use for building user interface, particularly for single page application.
- ✓ It is component based and allows developers to create reusable UI component.
- ✓ React-JS is faster as compare to JavaScript
- ✓ React-JS create a virtual DOM in Memory.

➔ **How is it different from other JavaScript frameworks and libraries?**

1. Library vs. Framework

   React.js is a library, not a full-fledged framework, focusing solely on the **view layer (UI)** of an application.

2. Virtual DOM
3. React-JS is faster as compare to other JS
4. It is totally based on components.
5. It use to create dynamic and user-friendly web.

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

Ans.

Core Principles of React

React is a powerful JavaScript library developed by Facebook for building user interfaces. Its design is based on a few core principles that make it efficient, flexible, and maintainable. Below are the key principles.

1. Virtual DOM

The Virtual DOM (Document Object Model) is a lightweight, in-memory representation of the real DOM that React uses to optimize UI rendering.

How the Virtual DOM Works:

- When a component's state or props change, React creates a new Virtual DOM tree to represent the updated UI.
- React then compares the new Virtual DOM tree with the previous one (a process called reconciliation).
- Only the differences (or "diffs") are identified and applied to the real DOM, minimizing expensive DOM operations.

Benefits of the Virtual DOM:

- Performance Optimization: Only the changed parts of the DOM are updated, improving efficiency.
- Better User Experience: Ensures fast updates, even for complex UIs.
- Abstraction: Developers don't have to manually handle DOM manipulations, as React manages them under the hood.

---

2. Component-Based Architecture

React applications are built as a collection of reusable and independent components. Each component is a small, self-contained unit responsible for rendering part of the UI.

Key Features of Components:

- Encapsulation: Components manage their own logic and state, making them reusable and isolated from other components.
- Reusability: Components can be used multiple times across the application, reducing redundancy.
- Composition: Components can be nested and combined to build complex UIs.

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

Ans.

- ✓ Flexibility [we can use HTML,CSS for customize this webpage]
- ✓ Reusable Component
- ✓ Virtual DOM
- ✓ It is faster hen other JS
- ✓ Declarative syntax
- ✓ Strong community support.

JSX (JavaScript XML)

Question 1: What is JSX in React.js?

Ans.

JSX (JavaScript XML) is a syntax extension for JavaScript, used in React to describe what the UI should look like. It allows developers to write HTML-like code directly within JavaScript.

➔ **Why is it used?**

- ✓ Improved Readability
- ✓ Integration with JavaScript Binds HTML with React Components
- ✓ Prevents Cross-Site Scripting (XSS)
- ✓ Enhanced Debugging

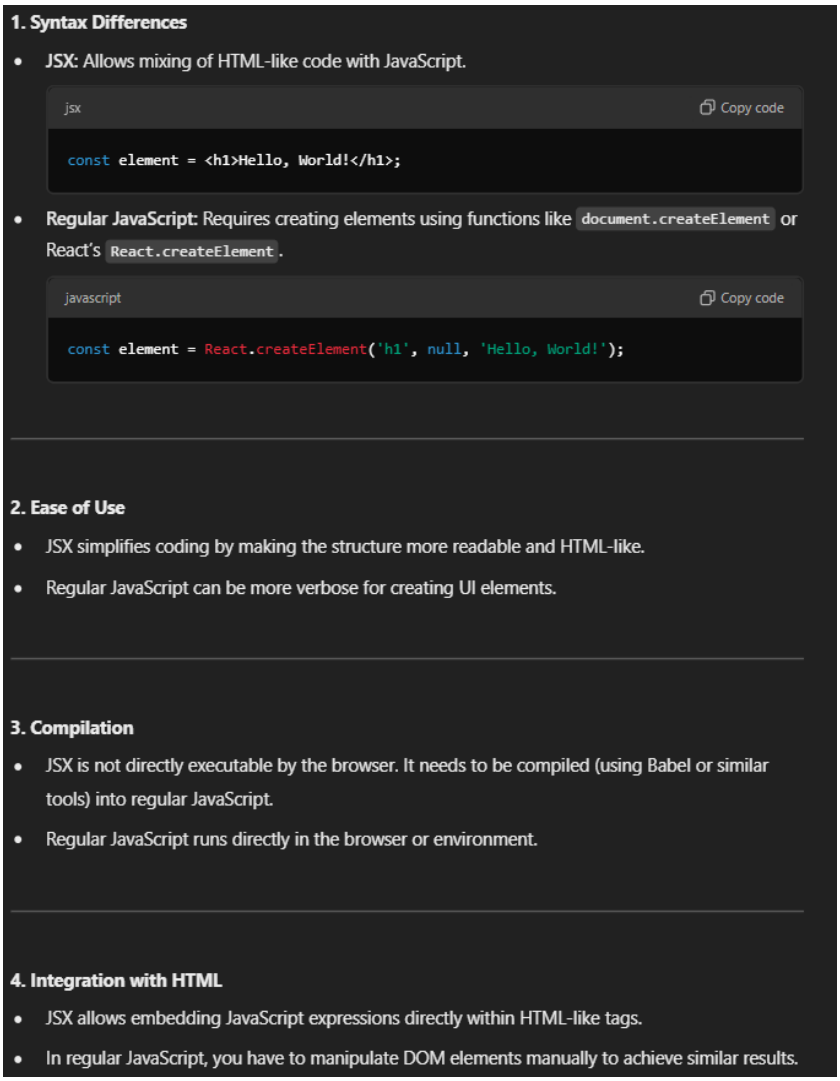
✓ Virtual DOM Optimization
✓ Custom Component Support

Question 2: How is JSX different from regular JavaScript?

Ans.

JSX is a syntax extension of JavaScript that allows you to write HTML-like code directly within JavaScript. While it looks like HTML, it is not valid JavaScript or HTML—it's transformed into JavaScript before execution. Below are the key differences:

➔ **Can you write JavaScript insideJSX?**
Yes, you can embed JavaScript expressions inside JSX using curly braces {}. These expressions can include variables, function calls, or any valid JavaScript expression.



1. Syntax Differences

- **JSX:** Allows mixing of HTML-like code with JavaScript.

jsx — Copy code

```jsx
const element = <h1>Hello, World!</h1>;
```

- **Regular JavaScript:** Requires creating elements using functions like `document.createElement` or React's `React.createElement`.

javascript — Copy code

```javascript
const element = React.createElement('h1', null, 'Hello, World!');
```

---

2. Ease of Use

- JSX simplifies coding by making the structure more readable and HTML-like.
- Regular JavaScript can be more verbose for creating UI elements.

---

3. Compilation

- JSX is not directly executable by the browser. It needs to be compiled (using Babel or similar tools) into regular JavaScript.
- Regular JavaScript runs directly in the browser or environment.

---

4. Integration with HTML

- JSX allows embedding JavaScript expressions directly within HTML-like tags.
- In regular JavaScript, you have to manipulate DOM elements manually to achieve similar results.

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

Ans.

In React, curly braces {} play a vital role when working with JSX (JavaScript XML). They allow you to embed JavaScript expressions directly into the JSX markup, making your components dynamic and flexible.

Also, we can include bellow items Inside {} :-

1. **Variables:** {name}
2. **Functions or Calculations:** {age > 18 ? "Adult" : "Minor"}
3. **Function Calls:** {getGreeting()}
4. **Array Mapping**

Here's why curly braces are important:

1. Embedding JavaScript in JSX

- Curly braces allow you to insert JavaScript expressions inside JSX.
- You can use variables, function calls, mathematical operations, and other expressions within curly braces.

Example:

jsx Copy code

```jsx
function Greeting(props) {
  return <h1>Hello, {props.name}!</h1>;
}
// If props.name = 'John', the output will be: Hello, John!
```

2. Dynamic Rendering

- JSX with curly braces enables **dynamic rendering** of data, such as:
  - Showing dynamic text.
  - Calculating values on the fly.
  - Rendering conditions (e.g., ternary operators).

Example:

jsx Copy code

```jsx
function Sum() {
  const a = 5, b = 10;
  return <p>The sum is: {a + b}</p>;
}
// Output: The sum is: 15
```

3. Conditional Rendering

- Curly braces allow **conditional rendering** using JavaScript logic.
- Commonly used with ternary operators or short-circuit evaluations.

Example:

jsx Copy code

```jsx
function Status({ isLoggedIn }) {
  return <p>{isLoggedIn ? 'Welcome Back!' : 'Please Log In'}</p>;
}
// Output changes based on the value of isLoggedIn.
```

4. Loops and Arrays

- Curly braces allow embedding JavaScript methods, such as `map()`, to dynamically generate elements.

Example:

jsx Copy code

```jsx
function ItemList({ items }) {
  return (
    <ul>
      {items.map((item, index) => (
        <li key={index}>{item}</li>
      ))}
    </ul>
  );
}
// Renders a list based on the items array.
```

5. Inline Styling

- When applying inline styles in JSX, curly braces are used to pass JavaScript objects.

Example:

jsx Copy code

```jsx
function StyledText() {
  const style = { color: 'blue', fontSize: '20px' };
  return <p style={style}>This is styled text.</p>;
}
```

What Curly Braces Cannot Do in JSX

- You cannot use **statements** like `if`, `for`, or `while` inside curly braces, as JSX supports **expressions** only.

Invalid:

jsx Copy code

```jsx
{if (condition) { return value; }}  // This will throw an error.
```

Components (Functional & Class Components)

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

Ans.

- Component are the building blocks of a react application.
- They allow developers to split the user interface into independent, reusability, pieces, making it easier to manage and develop.
- A component can be thought of as a JavaScript function or class that accept inputs(props) and returns a react element that describes what should appear on the screen.

Types of components:-

1) Functional component
2) Class component

Difference between functional component and class component:-

Feature	Functional Components	Class Components
Definition	JavaScript functions	ES6 classes extending `React.Component`
State Management	Uses **React Hooks** (`useState`, `useEffect`, etc.)	Uses `this.state` and `this.setState()`
Lifecycle Methods	Hooks like `useEffect` handle lifecycle events	Lifecycle methods like `componentDidMount`, etc.
Code Complexity	Simpler and more concise	More verbose due to `this` binding and boilerplate
Performance	Slightly faster due to less overhead	Slightly slower due to class-related overhead
Usage	Preferred in modern React applications	Legacy approach, mostly used in older projects

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

Ans.

In React, **props (properties)** are used to pass data from a parent component to a child component. Props are immutable, meaning they cannot be changed by the child component.

How to pass data using props..

1. Define Props in the Parent Component

- You specify the data as attributes of the child component when rendering it inside the parent component.

2. Access Props in the Child Component

- The child component receives the **props as an argument in functional components** or through **this.props in class components.**

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

Ans.

- The render() method is a key component of React's class-based architecture. Its primary purpose is to define the structure and content of the component's user interface (UI).
- It serves as the bridge between the component's data (state/props) and the virtual DOM.
- The render() method is crucial in React class components for defining and updating the user interface.
- It encapsulates the logic for displaying UI in a declarative and predictable way, ensuring efficient updates and maintaining React's core philosophy of component-based architecture.

Props and State

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

Ans.

- In React, props (properties) are a way to pass data from a parent component to a child component. They are used to make components reusable and dynamic by allowing a parent component to provide inputs data or functions to its children.

- State is a built-in object in React used to store and manage the dynamic data of a component. Unlike props, which are external and passed from a parent, the state is internal to a component and can be modified within that component.

Aspect	Props	State
Definition	External data passed to a component.	Internal data managed by a component.
Mutability	Immutable (read-only).	Mutable (can be updated).
Responsibility	Controlled by the parent component.	Controlled by the component itself.
Purpose	To pass data and configure child components.	To manage dynamic and interactive data.
Triggers Re-Render	Changing props does not directly trigger a re-render.	Changing state triggers a re-render.
Accessibility	Passed to child components.	Local to the component where it is defined.
Examples	Text content, themes, or configurations.	User input, toggles, or counters.

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

Ans.

- State in React is a built-in object that allows components to store, update, and manage dynamic data.
- It is an essential concept for building interactive and responsive user interfaces.
- State enables React components to "remember" information across renders and update their output dynamically when user interactions or events occur.

1. Initializing State

State is initialized when a component is created, typically to hold default values for the component's data.

- **Class Components**: Use the this.state object.
- **Functional Components:** Use the **useState** hook.

2. Updating State

State can be updated using:

- Class Components: this.setState method.
- Functional Components: The setter function from useState.

3. Binding State to UI

State values are often used to dynamically render UI elements.

4. Managing Dynamic Data

State allows components to handle dynamic data, such as:

- **Form Inputs**: Tracking user input in forms.

5. Handling Asynchronous Updates

State is often used to store data fetched from an API.

6. Conditional Rendering with State

State enables conditional rendering of UI elements.

7. Synchronizing State Between Components

State in a parent component can be passed as props to child components to synchronize data.

In short:-

- State is a core feature of React that provides a way to manage and update dynamic data within components.
- It helps in creating interactive, responsive, and dynamic user interfaces by ensuring the component data and UI are always synchronized.
- By using state effectively, developers can build applications that handle user interactions, manage data updates, and create dynamic views efficiently.

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

Ans.

- In React class components, this.setState() is the method used to update the state of a component.

- Directly modifying the state (e.g., this.state.someValue = newValue) is not allowed, as React relies on state changes to trigger re-renders and maintain the virtual DOM efficiently.
- The setState() method ensures state updates are handled correctly and the component re-renders as needed.

How this.setState() Works

1. **Schedules a State Update**:
    - this.setState() schedules a change to the component's state. The update is merged with the current state rather than replacing it entirely.
    - React processes this update asynchronously for performance optimization.
2. **Triggers Re-rendering**:
    - When the state is updated using setState(), React re-evaluates the component's render() method.
    - This ensures the user interface reflects the latest state.
3. **Partial State Updates**:
    - Only the properties specified in setState() are updated. Other properties in the state remain unchanged.

Example of `this.setState()`

Initial State and Update:

jsx Copy code

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

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

  render() {
    return (
      <div>
        <p>Count: {this.state.count}</p>
        <button onClick={this.increment}>Increment</button>
      </div>
    );
  }
}
```