Introduction to React.js

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

Answer 1:

1. **React.js** is a JavaScript library for building user interfaces.
2. It uses a component-based structure for reusable code.
3. React updates the UI efficiently using a Virtual DOM.
4. JSX lets you write HTML-like code in JavaScript.

**Different from other JavaScript frameworks and libraries:**

* **React is a library**, not a full framework, meaning it only handles the user interface, while other tools handle the rest.
* It uses **components** to break the UI into smaller, reusable pieces of code.
* React updates the UI **efficiently** with a Virtual DOM, making it faster than regular JavaScript.
* It’s **flexible** and can be combined with other libraries or frameworks.

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

Answer 2:

**Virtual DOM:**

* The Virtual DOM is a lightweight copy of the actual DOM which make DOM manipulation easier.

**Component based architecture:**

* Components are like function that return HTML element.

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

Answer 3:

**The advantages of using React.js in web development:**

1. **Fast Performance**: React uses a Virtual DOM, which makes updates and rendering faster.
2. **Reusable Components**: You can create components that can be reused, saving time and effort.
3. **Easy to Learn**: React’s syntax, especially with JSX, is simple and intuitive for developers.
4. **Large Community**: React has a big community, so you get lots of support and resources.
5. **Scalability**: React makes it easier to scale up your application as it grows.
6. **Cross-Platform**: React Native lets you build mobile apps for iOS and Android using the same concepts.

JSX (JavaScript XML)

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

Answer 1:

* JSX means JavaScript XML (Extensible Markup Language)
* JSX stands for JavaScript XML.
* JSX allows us to Write HTML in React.
* JSX makes it easier to write and add HTML in React.
* JSX code is copied into JavaScript by React.

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

Answer 2:

* "JavaScript needs to be written inside the <script> tag in an HTML file, but JSX can't be written directly in HTML files. JSX is written in React, and React converts it into JavaScript. JSX code is written in a JavaScript file."
* "Yes, you can write JavaScript inside JSX. To do this, you use curly braces {} around JavaScript expressions, like {name} or {2 + 2}, which inserts dynamic values into JSX."

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

Answer 3:

* In JSX, {} are used to include JavaScript expressions.
* They allow you to display variables, functions, or calculations.
* Example: <p>Hello, {name}! </p> will show dynamic content.
* Without {}, JSX treats content as plain text.
* Curly braces help mix logic and UI together.

Components (Functional & Class Components)

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

Answer 1:

* + Components are **independent** and **reusable blocks of code**.
  + They define specific parts of the **User Interface (UI)**, like a button, a header, or a form.

**Difference between Functional Components and Class Components:**

|  |  |
| --- | --- |
| **Function Components** | **Class Components** |
| Defined using a regular JavaScript Function. | Defined using ES6 Class. |
| Managed using the useState hook. | Managed using this.state. |
| Uses useEffect for lifecycle behavior. | Uses lifecycle methods like componentDidMount. |
| Can use hooks like useState, useEffect. | Not applicable. |

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

Answer 2:

* To pass data to a component using **props** in React:
* **Props** are like inputs that you give to a component.
* The **parent component** sends data to the **child component** using props.
* The **child component** receives this data and can use it to render content.
* In simple terms:
* The parent gives data to the child.
* The child uses that data by referring to it as props.

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

Answer 3:

* In class components, the **render ()** method is responsible for showing what the UI should look like. It returns the JSX (the HTML-like code) that gets displayed on the screen.
* Every class component must have a **render ()** method.
* It defines how the component should appear based on its data (like props or state).
* The render () method must return JSX.
* In simple terms, **render ()** is the function that **creates the UI** for your class component.

Props and State

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

Answer 1:

* Props are inputs passed from a parent component to a child component.
* They allowing it to receive data and behave dynamically.
* They are read-only and cannot be modified by the child component.

**Props Different from State:**

* Props come from the parent component while state is managed inside the component.
* Props are read-only, but state can change over time.
* Props are used for passing data While state is used for handling dynamic changes.

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

Answer 2:

* In React, **state** is a way to store and manage data that can change over time in a component. It is used to track things like user inputs, button clicks, or any information that changes during the lifecycle of the component.

**How State Works:**

1. **State is initialized** in a component, usually in the constructor (for class components) or with the useState hook (for functional components).
2. **State can change** based on actions like clicking a button, typing in a form, etc.
3. **When state changes**, React re-renders the component to reflect the updated data on the screen.

**In simple terms:**

* State allows your component to "remember" things and update the UI when something changes.

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

Answer 3:

* this.setState() updates the state in React class components.
* It tells React to re-render the component with the updated state.
* The state update happens asynchronously for better performance.
* React merges the new state with the existing one automatically.
* It ensures the UI stays in sync with the component's data.