Java Script

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

**Ans**.

**What is React.js?**

React.js is a **JavaScript library** used to build the parts of a website or app you interact with, like buttons, forms, or entire pages. It helps developers create dynamic and fast websites by splitting them into small, reusable pieces called

**components**.

**How is React.js Different from Other Tools?**

| **React.js** | **Other Tools (like Angular or jQuery)** |
| --- | --- |
| **Type**: A library focused on building UIs. | Frameworks like Angular do more but are bigger and stricter. |
| **Speed**: Uses a **Virtual DOM**, making updates fast. | jQuery and others update the whole page, which can be slower. |
| **Flexibility**: Works with any project or tool. | Frameworks like Angular control more of how your app is built. |
| **Data Flow**: Uses **one-way data flow** (easier to manage). | Some, like Angular, use two-way data flow, which can get complex. |

**Why Use React?**

* It’s **simple** and helps organize your code.
* Makes apps **faster** by updating only what changes.
* **Reusable components** save time when building.

React is great for creating dynamic websites, like Facebook or Instagram, and is easier to learn and more flexible than many other tools.

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

**Ans**.

1. **Virtual DOM**
   * The **DOM** is like a map of everything on your webpage.
   * React creates a **virtual DOM**, a lightweight copy of the real DOM, to make updates faster.
   * Instead of changing the whole page when something updates, React only changes the part that actually needs it.
   * This makes your app run smoother and faster.
2. **Component-Based Architecture**
   * A **component** is like a Lego block—it’s a small piece of your website, such as a button, form, or menu.
   * You can build a whole app by combining these components.
   * Each component is **reusable**, so you can use the same "block" in different places without building it from scratch again.
   * Components also manage their own data (called **state**) and can talk to each other using **props** (short for properties).

**Why These Principles Matter:**

* **Virtual DOM** makes updates efficient and fast.
* **Components** keep the code clean, organized, and easy to maintain.

By using these principles, React makes building apps quicker and less complicated!

Top of Form

Bottom of Form

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

**Ans**.

 **Reusable Components**

* You can create small building blocks (like buttons or forms) and use them in different parts of your app, saving time.

 **Fast Performance**

* React updates only the parts of the page that change using the **Virtual DOM**, making your app faster and smoother.

 **Easy to Learn**

* React is simple if you know JavaScript, and its syntax (JSX) is like writing HTML inside JavaScript.

 **Great Flexibility**

* You can use React with other tools or libraries and even add it to existing projects.

 **One-Way Data Flow**

* React makes it easier to track and manage data by moving it in one direction, reducing bugs.

 **Strong Community Support**

* React has lots of tutorials, libraries, and tools because many developers and companies use it.

 **Works Everywhere**

* With **React Native**, you can use the same React principles to build mobile apps for iOS and Android.

 **Rich Ecosystem**

* Tools like React Router (for navigation) and Redux (for state management) make development even easier.

**Q 4** **: What is JSX in React.js? Why is it used?**

**Ans**.

* 1. Extension to java script language syntax
  2. Jsx = java script and xml
  3. Jsx also contain all the html like attribute
  4. Jsx make the code simple and look elegant
  5. It is not necessary to write jsx to render html tags

JSX (JavaScript XML) is a syntax used in **React.js** to write HTML-like code inside JavaScript. It makes it easier to create and visualize user interfaces.

**Why is JSX used?**

1. **Readability**: JSX looks like HTML, so it’s easier to read and understand the UI structure.
2. **Integration with JavaScript**: You can use JavaScript expressions (e.g., variables, loops) inside JSX to make your UI dynamic.
3. **Simplicity**: JSX combines the structure (HTML) and logic (JavaScript) in one place, making the code more organized.

**Example:**

Without JSX:

React.createElement('h1', null, 'Hello, World!');

With JSX:

<h1>Hello, World!</h1>

**Q 5** **: : How is JSX different from regular JavaScript? Can you write JavaScript insideJSX?**

**Ans**.

**How is JSX different from regular JavaScript?**

1. **HTML-like Syntax**: JSX looks like HTML, but it’s not. It lets you write HTML-style tags in JavaScript code.
2. **Requires Compilation**: JSX isn’t valid JavaScript, so it needs to be converted (using tools like Babel) into plain JavaScript before the browser can understand it.
3. **Easier UI Creation**: With JSX, you can define what your user interface should look like in a more intuitive way compared to regular JavaScript.

**Can you write JavaScript inside JSX?**

Yes! You can write JavaScript inside JSX by wrapping it in **curly braces {}**.

**Example:**

const name = "John";

// Using JavaScript inside JSX

<h1>Hello, {name}!</h1>

Hello, John!

**Q 6: Discuss the importance of using curly braces {} in JSX expressions.**

**Ans**.

In JSX, **curly braces {}** are very important because they let you use **JavaScript code** inside your HTML-like JSX.

**Why are curly braces {} important?**

1. **Dynamic Content**: You can insert JavaScript variables, calculations, or expressions into your JSX.  
   Example:

const name = "Alice";

<h1>Hello, {name}!</h1>

Output: Hello, Alice!

1. **Logic and Conditions**: You can use JavaScript logic, like if-else or ternary operators, to show different things.  
   Example:

const isLoggedIn = true;

<p>{isLoggedIn ? "Welcome back!" : "Please log in."}</p>

Output: Welcome back!

1. **Function Calls**: You can call functions inside curly braces to show their results.  
   Example:

function getGreeting() {

return "Good morning!";

}

<h2>{getGreeting()}</h2>

Output: Good morning!

1. **Dynamic Styling**: Curly braces let you pass JavaScript objects for inline styles.  
   Example:

const style = { color: "blue", fontSize: "20px" };

<p style={style}>This is styled text.</p>

Without curly braces, JSX would treat everything as plain text, and you couldn’t use JavaScript logic or dynamic content.

**Q 7** **: What are components in React? Explain the difference between functional components and class components.**

**Ans**.

**What are Components in React:-**

Components in React are like building blocks of a React application. They are **reusable pieces of code** that define how a part of the user interface (UI) should look and behave. Each component works independently, managing its own logic and UI.

| **Aspect** | **Functional Components** | **Class Components** |
| --- | --- | --- |
| **Syntax** | Simple JavaScript functions. | ES6 classes that extend React. Component. |
| **State Handling** | Use React Hooks (useState, useEffect). | Manage state using this.state and setState(). |
| **Code Complexity** | Cleaner, easier to read, and less code. | More verbose and complex. |
| **Lifecycle** | Handle lifecycle using Hooks like useEffect. | Use built-in lifecycle methods like componentDidMount. |
| **Performance** | Slightly better in some cases due to simplicity. | May have additional overhead. |

**Q 8** **: How do you pass data to a component using props?**

**Ans**.

In React, **props** (short for "properties") are a way to send data from one component (usually a parent) to another component (usually a child).

**Full Example:**

function App() {

return <Greeting name="Alice" />;

}

function Greeting(props) {

return <h1>Hello, {props.name}!</h1>;

}

**Output:**  
Hello, Alice!

**Q 9** **: What is the role of render() in class components?**

**Ans**.

**What is the Role of render() in Class Components?**

In React **class components**, the render() method is used to define **what the component should display on the screen**.

**Key Points About render()**

1. **Renders JSX**:  
   The render() method returns JSX (HTML-like code) that tells React what UI should look like.
2. **Called Automatically**:  
   React automatically calls render() whenever the component needs to be shown or updated.
3. **Part of the React Lifecycle**:  
   It is a required method in every class component and is re-executed whenever the component's state or props change.

**When is render() Called?**

* When the component **first loads**.
* Whenever the **state** or **props** of the component are updated.

**Q 10:**  **What are props in React.js? How are props different from state?**

**Ans**.

**What are Props in React.js?**

* **Props** (short for "properties") are used to **pass data from a parent component to a child component** in React.
* They are like **function arguments** and help make components reusable by customizing them with different data.
* Props are **read-only**, meaning the child component cannot change them

**What is State in React.js?**

* **State** is used to manage data **within a component**.
* It represents values that can change over time, like a counter or user input.
* State is **mutable**, meaning you can update it, and React will re-render the component to reflect the changes.

**Key Differences Between Props and State**

| **Aspect** | **Props** | **State** |
| --- | --- | --- |
| **Definition** | Data passed **from parent to child**. | Data managed **within the component**. |
| **Mutability** | **Immutable** (cannot be changed). | **Mutable** (can be updated). |
| **Usage** | Customize or configure child components. | Track and manage dynamic data. |
| **Ownership** | Controlled by the **parent component**. | Controlled by the **component itself**. |

**Q 11** **: Explain the concept of state in React and how it is used to manage componentdata**

**Ans**.

**What is State in React?**

In React, **state** is an object that holds **dynamic data** for a component.  
It is like a **memory** for a component that remembers information between renders and allows the component to respond to user actions, changes, or events.

**Key Features of State**

1. **Managed Locally**:  
   State is private and belongs to the component where it is created. Other components cannot access it directly.
2. **Mutable**:  
   Unlike props, state can be **changed** using specific methods (e.g., setState in class components or useState in functional components).
3. **Triggers Re-rendering**:  
   When the state is updated, React automatically re-renders the component to reflect the changes in the UI.

**Q 11** **: Why is this.setState() used in class components, and how does it work?**

**Ans**.

**Why is this.setState() Used in Class Components?**

In React **class components**, this.setState() is the method used to **update the component's state**. It allows you to change the state and tells React to **re-render the component** to reflect the updated data in the UI.

**How this.setState() Works**

1. **Updates State**:  
   When you call this.setState(), it merges the new state values with the existing state.
2. **Triggers Re-render**:  
   After updating the state, React automatically re-runs the render() method to update the component's UI.
3. **Asynchronous Behavior**:  
   this.setState() works asynchronously, meaning updates might not happen immediately. React batches updates for performance

Ex:-

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

{ console.log("State updated:", this.state.count); });