Introduction :-

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

Ans :-

- A JavaScript library for building user interfaces.
- Developed by Meta (Facebook).
- Used for single-page applications (SPAs).
- Works with a virtual DOM for fast updates.
- Uses component-based architecture for reusability.

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

Ans :-

1.Virtual DOM (VDOM)

- React uses a Virtual DOM, which is a lightweight copy of the real DOM.
- It updates only the changed parts instead of re-rendering the entire UI.
- Improves performance by minimizing direct DOM manipulations.

2.Component-Based Architecture

- UI is built using independent, reusable components.
- Each component has its own state and logic.
- Makes development modular, scalable, and easier to maintain.

3.One-Way Data Binding

- Data flows in one direction (from parent to child components).
- Ensures better control over data and prevents unexpected changes.

4.Declarative UI

- React describes what the UI should look like, not how to update it.

- Makes code easier to read, debug, and manage.

---

5. State Management

- Uses useState, useReducer, or tools like Redux for managing component state.
- Helps in handling dynamic data efficiently.

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

Ans :-

1. Fast Performance

- Uses Virtual DOM for efficient rendering.
- Updates only the changed parts instead of reloading the entire page.

2. Component-Based Architecture

- UI is built with reusable components.
- Makes code modular, maintainable, and scalable.

3. One-Way Data Binding

- Ensures better control over data flow.
- Prevents unwanted UI changes and improves debugging.

4. Declarative UI

- React updates UI automatically when state changes.
- Code is easier to read, understand, and debug.

5. Rich Ecosystem & Community Support

- Large community with many third-party libraries.
- Constant updates and improvements.

6. Easy Integration

- Can be used with other frameworks (Angular, Vue, etc.).
- Works well with backend technologies like Node.js.

7. SEO-Friendly

- Faster page load time improves SEO ranking.
- Can be used with Next.js for server-side rendering (SSR).

8. Mobile App Development with React Native

- Code can be reused for mobile apps using React Native.
- Saves development time and effort.

JSX (JavaScript XML) :-

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

Ans :- JSX (**JavaScript XML**) is a syntax extension for JavaScript that allows writing HTML-like code inside JavaScript. It makes React components easier to read and write.

Why is JSX Used?

Readable & Clean Code – Looks like HTML, making UI structure clear.
Mixes HTML & JavaScript – Allows inserting JavaScript expressions inside {}.
Better Performance – Compiles to optimized JavaScript (React.createElement).
Prevents Security Risks – Escapes harmful inputs to prevent XSS attacks.
Component-Based UI – Simplifies building and managing UI components.

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

Ans :- **1. Syntax:** JSX looks like HTML inside JavaScript, while regular JS follows pure JS syntax.
2. Usage: JSX is used in React for UI components, whereas JS is used for logic.
3. Compilation: JSX is converted into `React.createElement()`, while JS runs directly in the browser.
4. Attributes: JSX uses `className` instead of `class` for styling.
5. Readability: JSX makes UI code cleaner and more readable**.**

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

Ans:-

1. Embeds JavaScript Inside JSX

- Allows inserting variables, functions, and expressions in JSX.
- Eg:-

```
const name = "Avani";

<h2>Hello, {name}!</h2>;
```

2. Supports Dynamic Rendering

- You can use **conditions, loops, and calculations** inside {}.
- Eg:-

```
const age = 23;

<p>{age >= 18 ? "Adult" : "Minor"}</p>;
```

3. Works with Functions

- Calls functions inside JSX for **dynamic content**.
- Eg:-

```
function getGreeting() {

  return "Welcome to React!";

}

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

4. Allows Inline Styling

- Uses {} to pass **JavaScript objects** for styling.
- Eg:-

```
<h2 style={{ color: "blue", fontSize: "20px" }}>Hello!</h2>;
```

Components (Functional & Class Components) :-

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

Ans:- 1.Components :-

1.Reusable UI blocks – Components help break down a UI into smaller, manageable parts.
2. Two types – React has functional and class components.
3. Returns JSX – Components return JSX (HTML-like syntax).

4. **Modular & Maintainable** – Makes the code structured and easy to debug.
5. **Used to Build UI** – React apps consist of multiple components working together.

2.How is JSX different from Regular JavaScript?

1.JSX looks like HTML but works inside JavaScript.
2. JSX is used in React, whereas JavaScript is standalone.
3. JSX requires `{}` for inserting JavaScript expressions.
4. JSX uses `className` instead of `class` for styling.
5. JSX needs Babel to convert it into `React.createElement()` calls.

3. Can You Write JavaScript Inside JSX?

1. **Embeds JavaScript** – Allows using variables, functions, and expressions inside JSX.
2. **Enables Dynamic Content** – Supports conditions, loops, and calculations.
3. **Calls Functions** – Functions can be executed inside JSX.
4. **Allows Inline Styling** – CSS styles can be applied using objects inside `{}`.
5. **Makes JSX More Powerful** – `{}` adds flexibility to React components.

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

Ans:-

1. Props (short for "properties") are used to pass data from a parent component to a child component.
2. Props are read-only (immutable) and cannot be modified by the child component.
3. Props are passed as attributes in the component tag.
4. The child component receives props as a parameter.
5. Props help make components reusable and dynamic.

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

Ans :-

- **Required in Class Components** – Every class component must have a render() method.
- **Returns JSX** – It returns the UI (JSX) that will be displayed on the screen.
- **Runs Automatically** – React calls render() when the component is first loaded and when state/props change.
- **Cannot Modify State** – The render() method should not directly update the component's state.
- **Pure Function** – It should return the same UI for the same state/props.

Props and State :-

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

Ans :-

1. **Definition** – Props (short for "properties") are used to pass data from a parent component to a child component.
2. **Read-Only** – Props cannot be modified by the child component.
3. **Passed as Attributes** – Props are passed as attributes when using a component.
4. **Makes Components Reusable** – Props allow components to be dynamic and reusable.

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

Ans :-

Definition – State is an object in React that holds dynamic data and controls a component's behavior.

Managed Inside Component – Unlike props, state is controlled and updated within the component.

Changes Trigger Re-rendering – When the state updates, React automatically re-renders the component.

Use in Functional Components – Managed using the useState hook in functional components.

Use in Class Components – Managed using this.state and updated with this.setState in class components.

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

Ans :-

Updates Component State – this.setState() is the only way to update state in class components.

Triggers Re-render – When state changes, React automatically re-renders the component.

Ensures Batch Updates – React groups multiple setState() calls to optimize performance.

Does Not Mutate State Directly – Direct state modification (this.state.value = newValue) does not trigger re-rendering.