1.What is React.js? How is it different from other JavaScript frameworks and Libraries?

React.js is a JavaScript library developed by Facebook in 2013.
It's mainly used for building user interfaces especially single-page applications where the web page updates dynamically without reloading.

- Fast → Uses Virtual DOM, updates only what's needed.
- Reusable → Component-based.
- Flexible → Just a library, not strict like Angular.
- Easier → Learning curve is lighter than Angular.
- Popular → Huge community, ecosystem.

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

React works on a few core principles that make it powerful and efficient. It follows a component-based architecture, where the user interface is broken into small, reusable building blocks like buttons, cards, or navbars. This makes applications easier to build, maintain, and scale. To improve performance, React uses a Virtual DOM, which is a lightweight copy of the real DOM. When something changes, React compares the new virtual DOM with the previous one and updates only the parts of the real DOM that actually changed, instead of re-rendering the whole page. React also provides a declarative approach to UI, meaning you just describe what the interface should look like for a given state, and React automatically updates it when the state changes. Finally, React follows a one-way data flow, where data moves from parent components down to child components. This predictable flow of information makes debugging and managing large applications much easier.

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

- Fast Performance
  Updates only the necessary parts of the page, making apps smoother and faster.
- Reusable Components
  Build UI components and reuse them across the app saves time and effort.
- Declarative UI
  You describe *what* the UI should look like, React handles the updating automatically.
- Easy to Learn & Use
  Focuses mainly on the UI. Easier than learning full frameworks like Angular.
- Strong Community & Ecosystem
  Huge support, many libraries, and lots of job opportunities.
- Flexibility
  Since it's just a library, you can choose your own tools for routing, state management, etc.
- One-Way Data Flow
  Makes apps more predictable and easier to debug.

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

JSX (JavaScript XML) is a syntax extension in React.js that allows developers to write HTML-like code directly within JavaScript. It makes the code more readable and easier to visualize because the UI structure looks similar to HTML. JSX also lets developers embed JavaScript expressions inside curly braces, combining logic and design in one place. Although browsers cannot understand JSX . It is used in React because it simplifies UI development, improves readability, and speeds up coding.

5. **How is JSX different from regular JavaScript? Can you write JavaScript inside JSX?**

- JSX looks like HTML inside JavaScript, while regular JavaScript uses functions to build UI.
- Browsers don't understand JSX it gets compiled into JavaScript.
- You can write JavaScript expressions inside JSX using `{ }` (e.g., `{name}`, `{2+2}`).
- Only expressions work, not full statements like `if` or `for`.

6. Discuss the importance of using curly braces `{}` in JSX expressios

In React JSX, curly braces `{}` are used to embed JavaScript expressions directly inside the markup. They are important because they allow developers to display dynamic content such as variables, calculations, and function results within the UI. For example, writing `<h1>Hello, {name}</h1>` will render the value of the variable `name` instead of plain text. Without curly braces, JSX treats everything as static text, but with them, the UI becomes dynamic and interactive. Thus, curly braces act as a bridge between JavaScript logic and the HTML-like structure in React components.

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

In React, **components** are the fundamental building blocks of a web application. A component is a small, reusable piece of code that defines how a part of the user interface should look and behave, such as a button, header, or form. Components allow developers to break down complex UIs into smaller, manageable parts, which makes the application easier to develop, maintain, and reuse. React supports two main types of components: functional components, class components.

Functional Components

- A JavaScript function that returns JSX.
- Simple, easy to write, mostly used in modern React.
- Can use Hooks to manage state and lifecycle.

Class Components

- An ES6 class that extends `React.Component`.
- Must include a `render()` method that returns JSX.
- Before Hooks, they were used for state and lifecycle methods.

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

- Props are used to send data from a parent component to a child component.
- They make components reusable by allowing different inputs.
- Props are read-only a child component cannot change the props it receives

9. **What is the role of `render()` in class components?**

The `render()` method in a class component defines the UI structure by returning JSX, and React calls it whenever the component needs to be displayed or updated.

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

Feature	Props	State
Definition	Data passed from parent to child	Data managed inside a component
Mutability	Can not be updated	Can be updated
Who owns it?	Parent component	The component itself
Usage	To send data and customize child components	To manage dynamic data that changes over time

11. **Explain the concept of state in React and how it is used to manage component Data**.

In React, state is used to store and manage a component's dynamic data. It is local to the component and changes over time, such as user input or counters. When the state updates using `setState` in class components `React automatically re-renders the` component, making the UI interactive and responsive.

12. **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.
- You cannot change `this.state` directly because React will not know the state has changed.
- `this.setState()` tells React that the state has been updated, so React can re-render the component with the new data.

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

In vanilla JS, you attach event listeners directly to DOM elements with `addEventListener`, and you get a native event object.

In React, you attach handlers via JSX props (`onClick`, onChange) and React uses event delegation (a single listener at the root) instead of attaching many listeners.

14. What are some common event handlers in React.js? Provide examples of `onClick`, `onChange`, and `onSubmit`.

1. On click

```
function App() {
  function handleClick() {
    alert("Button was clicked!");
  }

  return <button onClick={handleClick}>Click Me</button>;
}
```

2.onchange

```
import { useState } from "react";

function App() {
  const [name, setName] = useState("");

  function handleChange(event) {
    setName(event.target.value);
  }

  return (
    <div>
      <input type="text" onChange={handleChange} />
      <p>Typed: {name}</p>
    </div>
  );
}
```

3. on submit

```
function App() {
  function handleSubmit(event) {
    event.preventDefault(); // prevent page reload
    alert("Form submitted!");
  }

  return (
    <form onSubmit={handleSubmit}>
      <input type="text" placeholder="Enter something" />
      <button type="submit">Submit</button>
    </form>
  );
}
```

15. Why do you need to bind event handlers in class components?

In class components, methods don't automatically bind to the component instance.
When passed as event handlers, `this` becomes `undefined` (or refers to something else).
Without binding, you can't access `this.state` or `this.props`.
Binding fixes this → ensures `this` always refers to the component.
Alternatives: use arrow functions in class fields or switch to function components with hooks

16. **What is conditional rendering in React? How can you conditionally render elements in a React component?**

Conditional rendering in React = deciding what to show in the UI based on conditions.
You can do it using:

- `if/else` statements
- Ternary (`condition ? A : B`)
- Logical AND (`condition && A`)
- `switch` statements

17. **Explain how `if-else`, ternary operators, and `&&` (logical AND) are used in JSX for conditional rendering**

- if-else → Used outside JSX, good for complex conditions.
- Ternary (`? :`) → Used inside JSX, best for choosing between two elements.
- Logical AND (&&) → Used inside JSX, best for rendering only when a condition is true.

18. **How do you render a list of items in React? Why is it important to use keys when rendering lists?**

In React, lists are usually rendered using `.map()`.
Each list item needs a unique `key` prop.
Why keys matter:

- Help React identify items during re-renders.
- Make updates efficient
- Prevent UI bugs

19. What are keys in React, and what happens if you do not provide a unique key?

What Are Keys in React?

- Keys are special props (`key`) that uniquely identify elements in a list.
- They help React track which items have changed, been added, or removed during re-rendering.
- Keys should be unique among siblings but don't need to be globally unique

What Happens If You Don't Provide a Unique Key?

1. React shows a warning:
   "Each child in a list should have a unique 'key' prop."
2. Performance issues:
   React cannot efficiently determine which items changed, so it may re-render the entire list.
3. Potential UI bugs:
   Elements may be reused incorrectly, causing unexpected behavior (e.g., wrong input values in dynamic lists).

20. **How do you handle forms in React? Explain the concept of controlled components**

In React, forms are handled using state and event handlers, making inputs controlled components whose values are managed by React, allowing easy validation and dynamic updates

21. **What is the difference between controlled and uncontrolled components in React?**

Controlled Components

- Definition: Form elements whose value is controlled by React state.
- How it works: `value` is set from state, and `onChange` updates the state.
- Advantages: Easy to validate, manipulate, and respond to user input.

Uncontrolled Components

- Definition: Form elements that manage their own state in the DOM.
- How it works: React uses a ref to access the value when needed.
- Advantages: Less code for simple forms, easier to integrate non-React code.

22. **What are React hooks? How do `useState()` and `useEffect()` hooks work in functional components?**

- Hooks are special functions that let you use state and lifecycle features in functional components.
- Before hooks, only class components could have state and lifecycle methods.
- Common hooks: `useState`, `useEffect`, `useRef`, `useContext`, etc.

23. **What problems did hooks solve in React development? Why are hooks considered an important addition to React?**

Hooks solved key React issues:

- State in functional components → useState adds state without classes.
- Lifecycle management → useEffect handles side effects without lifecycle methods.
- Logic reuse → custom hooks allow clean, reusable stateful logic.
- Class complexity → hooks reduce boilerplate, `this` binding, and verbosity.

Why Hooks Are Important

- Simplify code → no need for class components for state or lifecycle.

- Enable logic reuse → custom hooks make code modular and reusable.
- Consistent patterns → all components can now use state, effects, context, etc., in the same way.
- Better readability and maintainability → cleaner, less error-prone code.

24. What is useRef ? How to work in react app?

- `useRef` is a React hook that provides a mutable reference that persists across re-renders.
- It can be used to access DOM elements directly or store mutable values without causing a re-render.
- Think of it as a way to "remember" a value between renders.

25. What is React Router? How does it handle routing in single-page applications?

- React Router is a library for React that enables client-side routing in single-page applications (SPAs).
- It allows you to navigate between different components (pages) without reloading the browser.
- Provides features like nested routes, route parameters, and redirects.
- In a single-page app, the browser normally doesn't reload when navigating.
- React Router intercepts URL changes and renders the corresponding React component instead of fetching a new HTML page.
- It uses components like `<BrowserRouter>`, `<Routes>`, and `<Route>` to manage routing.

26. Explain the difference between `BrowserRouter`, `Route`, `Link`, and `Switch` components in React Router.

Component	Purpose
BrowserRouter	Provides routing context for the app
Route	Maps a URL path to a React component
Link	Navigates to different routes without page reload
Switch/Routes	Renders only the first matching route (v5) / All routes (v6)