1. What is React?

Hinglish (Samjhne ke liye):
React ek **JavaScript library** hai jo UI banane ke liye use hoti hai. Ye component-based hai aur Virtual DOM ka use karti hai taaki UI fast aur efficient update ho.

English (Interview Style):
React is a **JavaScript library for building user interfaces**. It follows a **component-based architecture** and uses the **Virtual DOM** to efficiently update and render components, making applications fast and scalable.

---

2. What is JSX?

Hinglish:
JSX ek syntax extension hai jo JavaScript ke andar HTML-like code likhne deta hai. Browser ise directly samajhta nahi, isliye Babel ise normal JavaScript me transpile karta hai.

English:
JSX is a **syntax extension for JavaScript** that allows us to write HTML-like code inside JavaScript. Browsers cannot read JSX directly, so it is transpiled into plain JavaScript using tools like Babel.

---

3. What are Components in React?

Hinglish:
React me **Components building blocks** hote hain. Do type ke hote hain:

- **Class Components** – lifecycle methods + `this.state` use karte hain.
- **Functional Components** – simple functions hote hain aur Hooks ke through state aur lifecycle handle karte hain.

English:
Components are the **building blocks of a React application**. There are two types:

- **Class Components** – use lifecycle methods and `this.state`.
- **Functional Components** – are simpler and use **Hooks** to manage state and lifecycle.

---

4. Difference between Class and Functional Components.

Hinglish:
Class components heavy hote hain, lifecycle methods ke saath aate hain. Functional components light hote hain, aur Hooks use karte hain. Ab industry me mostly functional components prefer hote hain.

English:
Class components are ES6 classes that extend `React.Component` and use lifecycle methods and `this.state`.
Functional components are plain JavaScript functions that use **Hooks** to manage state and lifecycle. They are preferred today because they are **simpler, cleaner, and easier to test.**

---

5. What are Props in React?

Hinglish:
Props ek component se dusre component tak **data pass karne ka tareeka** hai. Ye read-only hote hain, component inhe modify nahi kar sakta.

English:
Props are **inputs to components**. They allow data to be passed from a parent to a child component. Props are **immutable**, meaning they cannot be modified by the receiving component.

---

6. What is State in React?

Hinglish:
State ek **mutable object** hai jo component ke andar data hold karta hai. Agar state change hoti hai toh component re-render hota hai.

English:
State is a **mutable object managed within a component**. It holds information that may change over time, and when the state changes, React re-renders the component to reflect the updated data.

---

7. What are Hooks in React?

Hinglish:
Hooks special functions hote hain jo functional components me state aur lifecycle features laate hain. Example:

- `useState` (state management)
- `useEffect` (side effects like API calls)
- `useContext` (global data access)

English:
Hooks are **special functions** in React that let functional components use state and lifecycle features.
For example:

- `useState` manages local state.
- `useEffect` handles side effects such as data fetching.
- `useContext` provides access to global data without prop drilling.

---

8. Explain State Management in React.

Hinglish:
Chhoti applications ke liye hum `useState` ya `useReducer` use karte hain. Agar data multiple components me share karna ho toh Context API ya external libraries (Redux, Zustand) use hote hain.

English:
State management in React can be handled in different ways:

- **Local state** – with `useState` or `useReducer`.
- **Global state** – using **Context API**.
- **Complex state management** – using external libraries like **Redux, MobX, or Zustand**.

---

9. What is the Virtual DOM?

Hinglish:
Virtual DOM ek lightweight copy hota hai real DOM ka. Jab state change hoti hai toh React Virtual DOM update karta hai aur fir diffing algorithm se sirf wohi part real DOM me update hota hai jo change hua hai.

English:
The Virtual DOM is a lightweight copy of the actual DOM. When state changes, React updates the Virtual DOM first, then uses a **diffing algorithm** to update only the necessary parts of the real DOM, improving performance.

---

10. What is useEffect Hook and its use cases?

Hinglish:
`useEffect` ek hook hai jo side effects ke liye use hota hai jaise API call, subscriptions, ya DOM manipulation.

- Agar dependency array na do → har render pe chalega.
- Agar empty array `[]` do → sirf initial render pe chalega.
- Agar specific dependency do → sirf us value ke change hone par chalega.

English:
`useEffect` is a Hook used for handling **side effects** such as data fetching, subscriptions, or manually changing the DOM.

- Without a dependency array → runs after every render.
- With an empty array `[]` → runs only once after initial render.
- With specific dependencies → runs only when those dependencies change.

---

11. Key prop in lists: Why is it needed? Best practices?

Hinglish (samjhne ke liye):
React me jab hum list of elements render karte hain (e.g. `map` use karke), toh `key` prop dena zaruri hai.

- Ye ek unique identifier hota hai, jisse React ko pata chalta hai kaunsa element change/update/remove hua hai.
- Agar `key` nahi doge ya galat doge (e.g. index use karna), toh unnecessary re-render ya bugs aa sakte hain.

Best Practices:

- Always use a unique id (like database ID) as a key.
- Avoid using array index as key (except in static lists with no updates).

English (Interview Style):
In React, the `key` prop is required when rendering lists because it helps React identify which items have changed, been added, or removed. This makes the rendering process more efficient.

Best practices:

- Use a unique identifier from your data (such as an ID).
- Avoid using array indexes as keys unless the list is static and never changes.

🔹 12. Conditional Rendering in React. What ways can we do it?

Hinglish:
Conditional rendering ka matlab hai ki hum kuch UI sirf specific condition pe dikhayenge. React me ise karne ke common tareeke hain:

1. if / else statements – normal JS logic use karke return different components.
2. Ternary operator (`condition ? A : B`) – short syntax.
3. Logical AND (`&&`) operator – agar condition true hai toh UI render hoga.
4. Switch statements – multiple cases ke liye.

English:
Conditional rendering in React means displaying UI based on certain conditions. Common approaches include:

1. Using if/else statements to return different elements.
2. Using the ternary operator (`condition ? ComponentA : ComponentB`).
3. Using logical AND (`&&`) for short-circuit rendering.
4. Using switch statements for multiple conditions.

🔹 13. Controlled vs Uncontrolled Components

Hinglish:
React me forms ke input handle karne ke do tareeke hain:

- Controlled Components:
  - Input value React ke state ke through control hoti hai.
  - `value` aur `onChange` dono React state se linked hote hain.
  - Example: `useState` ka use karke input ki value manage karna.
- Uncontrolled Components:
  - Input ki value directly DOM handle karta hai.
  - React state ke through control nahi hoti.
  - Access karne ke liye `ref` use karna padta hai.

English:

- Controlled Components:
  In controlled components, form data is handled by React state. The input's `value` is set from state and updated via `onChange`.

- Uncontrolled Components:
  In uncontrolled components, the form data is handled by the DOM itself, not React. We use `ref` to access the value directly.

👉 Controlled components are preferred in most cases because they keep the form state in sync with the React component’s state.

---

14. What is the Virtual DOM? How does React use it?

🔹 **Hinglish:**
Virtual DOM ek lightweight copy hota hai real DOM ka. Jab bhi React app me state/props change hota hai, React pehle Virtual DOM ko update karta hai, fir efficient diffing algorithm se changes calculate karke sirf required jagah real DOM update karta hai. Isse app fast rehta hai.

🔹 **English (Interview):**
Virtual DOM is a lightweight representation of the real DOM. When the state or props change, React updates the Virtual DOM first, then compares it with the previous version using a diffing algorithm. Finally, it updates only the necessary parts of the real DOM, which improves performance significantly.

---

15. What are React Lifecycle Methods?

🔹 **Hinglish:**
Lifecycle methods Class Components me hoti hain jaise:

- `componentDidMount` (initial render ke baad run hota hai)
- `componentDidUpdate` (jab props/state change ho)
- `componentWillUnmount` (cleanup ke liye).

Functional components me ye sab **useEffect Hook** se handle karte hain.

🔹 **English (Interview):**
React lifecycle methods in class components include `componentDidMount`,

`componentDidUpdate`, and `componentWillUnmount`. In functional components, these lifecycles are replaced by the `useEffect` hook, which can handle mounting, updating, and cleanup.

---

16. Difference between useEffect and useLayoutEffect

🔹 **Hinglish:**

- `useEffect` async tarike se run hota hai render ke baad, non-blocking hai.

- `useLayoutEffect` render ke turant baad, paint hone se pehle run hota hai (synchronous).
  Generally UI changes ke liye `useEffect` use karte hain, aur DOM measurement/layout adjustments ke liye `useLayoutEffect`.

🔹 **English (Interview):**
`useEffect` runs asynchronously after the render is committed to the screen, while `useLayoutEffect` runs synchronously after render but before the browser paints. `useLayoutEffect` is useful for DOM measurements or synchronizing layout, whereas `useEffect` is for general side effects like data fetching.

---

17. What is Context API? How is it different from prop drilling?

🔹 **Hinglish:**
Agar parent → child → grandchild me data bhejna ho to har level pe props pass karne padte hain (prop drilling). Context API isse bachata hai, ek global store create karta hai jisme se directly data consume kar sakte ho kisi bhi component me.

🔹 **English (Interview):**
The Context API allows you to share state globally across components without passing props manually through every level (prop drilling). It provides a Provider that supplies data and a Consumer that reads it, simplifying state sharing in deep component trees.

---

18. What are React Fragments? Why do we use them?

🔹 **Hinglish:**
React Fragment ek wrapper hota hai (`<> </>`) jo extra DOM nodes add kiye bina multiple elements return karne deta hai. Isse unnecessary `<div>` nesting avoid hoti hai.

🔹 **English (Interview):**
React Fragments allow us to group multiple elements without adding extra nodes to the DOM. Instead of wrapping everything in a `<div>`, we can use `<React.Fragment>` or shorthand `<> </>`.

---

19. What are Error Boundaries in React?

🔹 **Hinglish:**
Error boundaries special components hote hain jo apne child components ke runtime errors ko catch karte hain aur fallback UI dikhate hain instead of breaking the whole app.

🔹 **English (Interview):**
Error Boundaries are special React components that catch runtime errors in their child component tree and display a fallback UI instead of crashing the entire app. They are implemented using lifecycle methods like `componentDidCatch` and `getDerivedStateFromError`.

---

20. How does React handle performance optimization?

🔹 **Hinglish:**
React performance optimize karne ke liye:

- `React.memo` (component re-render avoid)
- `useMemo` (expensive calculation cache)
- `useCallback` (function re-creation avoid)
- Code Splitting & Lazy Loading
- Virtual DOM diffing

🔹 **English (Interview):**
React optimizes performance using techniques like `React.memo`, `useMemo`, `useCallback`,

Virtual DOM diffing, and features like code splitting and lazy loading to avoid unnecessary re-renders and load only what's required.

---

21. What is React Router? How routing works in React?

🔹 **Hinglish:**
React Router ek library hai jo single-page apps me multiple pages ka illusion deti hai. Ye URL ke basis pe component render karta hai. Nested routes, dynamic params, navigation sab support karta hai.

🔹 **English (Interview):**
React Router is a library for routing in React applications. It enables client-side navigation by mapping URL paths to components. It supports nested routes, route parameters, redirects, and programmatic navigation, making SPAs behave like multi-page apps.

---

22. What is Code Splitting and Lazy Loading in React?

🔹 **Hinglish:**
Code splitting matlab app ke bundle ko chote parts me todna. Lazy loading matlab component sirf tabhi load hoga jab zarurat ho (on demand). React me `React.lazy` + `Suspense` use karte hain.

🔹 **English (Interview):**
Code splitting divides the application bundle into smaller chunks, and lazy loading ensures that a component is loaded only when required. In React, this is achieved using `React.lazy` and `Suspense`.

---

23. What are Higher Order Components (HOCs)?

🔹 **Hinglish:**
HOC ek function hota hai jo ek component leta hai aur ek new component return karta hai with extra functionality. Ye mainly **reusability** ke liye use hota hai.

🔹 **English (Interview):**
A Higher Order Component (HOC) is a function that takes a component and returns a new component with added props or behavior. It is commonly used for reusability, cross-cutting concerns, and code abstraction.

---

24. What are Custom Hooks? Why do we create them?

🔹 **Hinglish:**
Custom hooks ek tarika hai apni logic ko reuse karne ka (like data fetching, form handling). Ye normal JS functions hote hain jo React hooks use karte hain internally.

🔹 **English (Interview):**
Custom Hooks are reusable functions that encapsulate stateful logic using existing React hooks. They help extract common logic such as fetching data or handling forms, making code cleaner and more reusable.

---

25. What are Rules of Hooks? Why are they important?

🔹 **Hinglish:**
Rules of Hooks:

1. Hooks sirf **top-level** pe call karna (loops, conditions ke andar nahi).
2. Hooks sirf **React function components ya custom hooks** ke andar use karna.

Agar ye follow nahi karoge to bugs aur inconsistent state ho sakti hai.

🔹 **English (Interview):**
The Rules of Hooks are:

1. Only call hooks at the top level of your component, never inside loops or conditions.
2. Only call hooks from React functional components or custom hooks.

These rules ensure that React can consistently track hook calls across renders.

---

26. What is JSX? Why do we use it?

🔹 **Hinglish:**
JSX ek syntax extension hai JavaScript ke liye jo HTML jesa dikhta hai. Isse hum UI ko likh sakte hain JavaScript code ke andar easily. React JSX ko compile karke `React.createElement()` calls me convert karta hai.

🔹 **English (Interview):**
JSX is a syntax extension for JavaScript that looks similar to HTML. It allows us to write UI

elements in a declarative way inside JavaScript. Under the hood, JSX is transformed into `React.createElement()` calls that generate React elements.

---

27. Difference between React and other frameworks (like Angular, Vue)?

🔹 **Hinglish:**

- **React** ek library hai (sirf UI banane ke liye) – lightweight aur flexible.
- **Angular** ek full-fledged framework hai (routing, state, HTTP sab built-in).
- **Vue** React aur Angular ke beech ka balance hai, simpler learning curve.

🔹 **English (Interview):**
React is a library focused on building user interfaces, offering flexibility and a component-based approach. Angular is a complete framework with built-in solutions for routing, state management, and HTTP requests. Vue provides a middle ground, with a simpler learning curve and built-in reactivity.

---

28. What is Reconciliation in React?

🔹 **Hinglish:**
Reconciliation wo process hai jisme React decide karta hai ki Virtual DOM me kya change hua aur kaunsa part actual DOM me update karna hai. Ye efficient diffing algorithm use karta hai jisse unnecessary DOM updates avoid ho.

🔹 **English (Interview):**
Reconciliation is the process React uses to compare the new Virtual DOM with the previous one and determine the minimal number of changes required in the real DOM. This process makes React efficient by avoiding unnecessary DOM updates.

---

29. How do you handle forms in React?

🔹 **Hinglish:**
Forms mostly **controlled components** ke through handle karte hain jaha input ka value state se bind hota hai aur `onChange` event se update hota hai. Agar directly DOM se value lena ho to **uncontrolled components** use karte hain (via refs). Badi apps me libraries like **Formik / React Hook Form** use hoti hain.

🔹 **English (Interview):**
In React, forms are typically handled using controlled components where the form input's value is controlled by React state and updated through event handlers. Uncontrolled components rely on refs to access form values directly. For large-scale forms, libraries like Formik or React Hook Form are commonly used.

---

30. Difference between Redux and Context API?

🔹 **Hinglish:**

- **Context API** simple global state share karne ke liye hota hai, lightweight hai aur chhoti apps ke liye sahi hai.
- **Redux** ek advanced state management library hai jo predictable state container, middleware, devtools support deta hai. Large apps me use karna best hai.

🔹 **English (Interview):**
Context API is a simple built-in solution in React for sharing global state without prop drilling, suitable for small to medium apps. Redux, on the other hand, is a dedicated state management library that provides predictable state containers, middleware, and developer tools, making it more suitable for complex and large-scale applications.

✅ Trusted Resources

1. **React Official Documentation**
   - React docs – Main Concepts
   - React docs – Hooks
   - [React docs – Advanced Guides (Performance, Context, Optimizing)]
2. **GeeksforGeeks – React Interview Questions**
   - 70+ questions from basic to advanced. GeeksforGeeks

3. **GitHub Repo – ReactJS Interview Questions & Answers**
    - SudheerJ’s repo: detailed questions + answers. GitHub
4. **YouTube Videos**
    - *Top 20 React JS Interview Questions For 2025* | *Intellipaat* YouTube

🚀 HTML Interview Questions (Intermediate Level)

1. What is DOCTYPE in HTML?

Hinglish: DOCTYPE browser ko batata hai ki HTML kaunsa version use ho raha hai (HTML5 etc).
English: DOCTYPE is a declaration that tells the browser which version of HTML the document is using, ensuring proper rendering (e.g., `<!DOCTYPE html>` for HTML5).

2. Difference between Block, Inline, and Inline-block elements?

Hinglish:

- Block: puri line le leta (div, p).
- Inline: sirf jitna content ho utna space (span, a).
- Inline-block: inline behave karta but height/width set kar sakte hain.
  English:
- Block-level: occupy full width (e.g., `<div>`).
- Inline: only takes as much width as content (e.g., `<span>`).
- Inline-block: behaves like inline but allows width/height to be applied.

3. What are semantic HTML tags?

Hinglish: Ye tags apne content ka meaning batate hain jaise `<header>`, `<footer>`, `<article>`, `<section>`.
English: Semantic HTML tags convey meaning of the content, such as `<header>`, `<article>`, `<nav>`, making code more accessible and SEO-friendly.

4. Difference between HTML vs HTML5?

Hinglish: HTML5 me multimedia tags (audio, video), semantic elements, localStorage/sessionStorage, canvas support aaya.
English: HTML5 introduced multimedia tags (`<audio>`, `<video>`), semantic tags (`<article>`, `<section>`), canvas, and offline storage features like localStorage and sessionStorage.

5. What is the difference between `id` and `class`?

Hinglish: `id` unique hota hai ek element ke liye; `class` multiple elements share kar sakte hain.
English: An `id` uniquely identifies a single element, while `class` can be shared across multiple elements for styling and scripting.

🎨 CSS Interview Questions (Intermediate Level)

6. Difference between relative, absolute, fixed, and sticky positioning?

Hinglish:

- Relative: apni normal position ke respect me.
- Absolute: nearest positioned ancestor ke respect me.
- Fixed: viewport ke respect me, scroll karne pe bhi wahi rehta.
- Sticky: relative + fixed dono ka mix, scroll hone pe stick ho jata.
 English:
- Relative: positioned relative to itself.
- Absolute: relative to nearest positioned ancestor.
- Fixed: relative to viewport, doesn't move on scroll.
- Sticky: toggles between relative and fixed depending on scroll.

7. Difference between Flexbox and Grid?

Hinglish: Flexbox one-dimensional layout (row/column); Grid two-dimensional (row + column dono).
English: Flexbox is a one-dimensional layout system (row or column), while CSS Grid is two-dimensional, handling both rows and columns simultaneously.

8. What is the difference between inline CSS, internal CSS, and external CSS?

Hinglish:

- Inline: `style` attribute.
- Internal: `<style>` tag inside HTML file.
- External: separate `.css` file.
  English:
- Inline applies styles directly on the element.
- Internal uses `<style>` within HTML.
- External links a `.css` file using `<link>`.

9. Difference between relative units (em, rem, %) and absolute units (px)?

Hinglish: px fixed hota hai; em/rem parent/root font-size pe depend karta; % parent ke size pe based hota.
English: px is absolute. em is relative to parent's font-size, rem is relative to root's font-size, and % is relative to the parent container.

10. What is z-index in CSS?

Hinglish: z-index stacking order control karta hai (kaunsa element upar ya niche dikhna hai).
English: z-index controls the stacking order of elements, determining which element appears in front or behind others.

⚡ JavaScript Interview Questions (Intermediate Level)

11. Difference between var, let, and const?

Hinglish: var function scoped, re-declare ho sakta; let block scoped, re-assign ho sakta; const block scoped, re-assign nahi ho sakta.
English: var is function-scoped and can be re-declared; let is block-scoped and can be reassigned; const is block-scoped and cannot be reassigned.

12. What is Hoisting in JavaScript?

Hinglish: Hoisting me variables/functions declarations top pe move ho jate hain, isliye unhe declare karne se pehle bhi access kar sakte ho (but var = undefined, let/const = TDZ).
English: Hoisting means variable and function declarations are moved to the top of their scope. var is hoisted with undefined, while let/const are hoisted but remain in a temporal dead zone.

13. Difference between == and ===?

Hinglish: `==` type coercion karta hai, `===` strict comparison karta hai (value + type dono check karta hai).
English: `==` checks equality with type coercion, while `===` checks both value and type strictly.

14. What are closures in JavaScript?

Hinglish: Closure ek function hai jo apne parent scope ke variables ko access kar sakta hai, even after parent function return ho jaye.
English: A closure is a function that retains access to its outer scope's variables, even after the outer function has returned.

15. What is the difference between synchronous and asynchronous JavaScript?

Hinglish:

- Sync: ek task khatam hoga tabhi dusra chalega.
- Async: tasks parallel chal sakte hain (callbacks, promises, async/await).
  English:
- Synchronous: executes one task at a time, blocking further execution.
- Asynchronous: allows non-blocking operations using callbacks, promises, or async/await.