FRONTEND INTERVIEW QUESTIONS

HTML, CSS, JavaScript & React - Comprehensive Guide

PART 1: HTML (20 QUESTIONS)

Semantic Tags

1. What are semantic HTML tags?

Semantic HTML tags provide meaning to the content they wrap. Examples include <header>, <footer>, <article>, and <section>, making it easier for search engines and developers to understand the page structure.

2. What is the <article> tag used for?

The <article> tag is used for self-contained content that can stand alone, such as a blog post or news article.

3. What is the difference between <section> and ?

<section> is a semantic tag that groups related content, while is a non-semantic tag used purely for layout purposes.

4. Why should you use semantic tags in HTML?

Semantic tags enhance accessibility, SEO, and make the code more understandable by defining the purpose of the content.

Attributes

5. What is an attribute in HTML?

An attribute is additional information provided inside the tag, modifying its behavior. For example, class or id.

6. What is the purpose of the alt attribute in images?

The alt attribute provides alternative text for an image, improving accessibility and SEO.

7. How do you create a hyperlink in HTML?

Use the href attribute in an <a> tag, e.g., Link.

8. How do you use the target attribute in an anchor tag?

The target attribute specifies where to open the linked document. For example, target="_blank" opens it in a new tab.

HTML Elements

9. What is the difference between block-level and inline elements?

Block-level elements (e.g., , <p>) take up the full width, while inline elements (e.g., , <a>) only take up as much width as their content.

10. What are void elements in HTML?

Void elements are elements that do not have closing tags. Examples include <img>,
, and <input>.

11. What is the purpose of the <meta> tag?

The <meta> tag provides metadata about the HTML document, such as charset, author, and description, which helps in SEO and page rendering.

12. How do you create an ordered list in HTML?

Use the <ol> tag for ordered lists and <li> for each list item, e.g., Item 1
Item 2.

Forms and Inputs

13. How do you create a form in HTML?

Use the <form> tag and include input elements like <input>, <textarea>, <select>, and <button>. For example, <form action="/submit" method="POST">.

14. What is the purpose of the method attribute in forms?

The method attribute specifies how the form data is sent to the server (GET or POST).

15. How do you create a checkbox input in HTML?

Use the <input type="checkbox"> tag for checkboxes, e.g., <input type="checkbox" name="subscribe" value="yes">.

16. How do you associate a label with a form element?

Use the for attribute in the <label> tag to associate it with an id of an input, e.g., <label for="name">Name</label><input id="name" type="text">.

Media

17. How do you embed an image in HTML?

Use the <img> tag with src and alt attributes, e.g., <img src="image.jpg" alt="description">.

18. How do you embed a video in HTML?

Use the <video> tag with the src and controls attributes, e.g., <video src="video.mp4" controls></video>.

19. What is the purpose of the <audio> tag?

The <audio> tag embeds sound content on a web page, with controls like play, pause, and volume.

20. How do you add a YouTube video to your HTML page?

You can embed a YouTube video using the <iframe> tag, e.g., <iframe src="https://www.youtube.com/embed/videoid"></iframe>.

PART 2: CSS (15 QUESTIONS)

Selectors

1. What are CSS selectors?

CSS selectors are used to select the HTML elements you want to style, such as element (p), class (.className), and ID (#id).

2. What is the difference between a class selector and an ID selector?

A class selector applies to multiple elements, while an ID selector applies to only one unique element.

3. How do you select all <p> elements inside a ?

Use the descendant selector: div p {} to select all <p> elements within a .

Box Model

4. What is the CSS box model?

The box model describes the layout of elements, including the content, padding, border, and margin.

5. How do padding and margin differ in the box model?

Padding is the space between the content and the element's border, while margin is the space outside the border between the element and other elements.

6. How do you set the width and height of an element including its padding and border?

Use box-sizing: border-box; to include padding and border in the element's total width and height.

Positioning and Layout

7. What is the position property in CSS?

The position property defines how an element is positioned on the page. Values include static, relative, absolute, and fixed.

8. What is the difference between absolute and relative positioning?

absolute positions an element relative to its nearest positioned ancestor, while relative positions an element relative to its normal position.

9. What is Flexbox in CSS?

Flexbox is a layout model that makes it easier to align and distribute space among items in a container using properties like display: flex, justify-content, and align-items.

Responsive Design

10. What are media queries in CSS?

Media queries allow you to apply CSS rules based on device characteristics like screen width, using @media rules, e.g., @media (max-width: 600px) {}.

11. What is the difference between min-width and max-width in media queries?

min-width applies styles when the viewport width is greater than the specified value, while max-width applies styles when the width is less than the specified value.

12. How do you create a responsive grid layout?

Use CSS Grid or Flexbox along with media queries to adjust the layout for different screen sizes.

Styling

13. How do you change the background color of an element?

Use the background-color property, e.g., body { background-color: lightblue; }.

14. What is the color property in CSS?

The color property sets the color of the text, e.g., p { color: red; }.

15. How do you change the font size of an element?

Use the font-size property, e.g., h1 { font-size: 32px; }.

PART 3: JAVASCRIPT (25 QUESTIONS)

DOM Manipulation

1. What is the DOM in JavaScript?

The DOM (Document Object Model) is a representation of the HTML structure as objects, allowing JavaScript to interact with and manipulate the content and layout.

2. How do you select an element by its ID in JavaScript?

Use document.getElementById('id') to select an element by its ID.

3. How do you add a class to an HTML element using JavaScript?

Use element.classList.add('class-name') to add a class to an element.

4. How do you remove an element from the DOM in JavaScript?

Use element.remove() to remove an element from the DOM.

5. How do you change the text content of an element in JavaScript?

Use element.textContent = 'New text' to change the text of an element.

Control Flow

6. What is a for loop in JavaScript?

A for loop is used to execute a block of code a certain number of times. Example: for (let i = 0; i < 5; i++) { console.log(i); }.

7. What is an if-else statement in JavaScript?

An if-else statement executes a block of code if a condition is true, otherwise it runs the code in the else block.

8. What is the purpose of the switch statement in JavaScript?

A switch statement allows you to execute different blocks of code based on the value of a variable.

9. What is a while loop in JavaScript?

A while loop repeatedly executes a block of code as long as a specified condition is true.

10. How do you exit a loop in JavaScript?

Use the break statement to exit a loop prematurely.

ES6 Features

11. What are arrow functions in JavaScript?

Arrow functions are a shorthand syntax for writing functions in JavaScript. Example: const add = (a, b) => a + b;.

12. What is destructuring in JavaScript?

Destructuring allows you to extract values from arrays or objects into variables. Example: const [a, b] = [1, 2];.

13. What is the difference between let and const in JavaScript?

let allows you to reassign variables, while const creates read-only constants that cannot be reassigned.

14. What is template literals in JavaScript?

Template literals are string literals that allow embedded expressions, using backticks (`) and ${} for placeholders.

15. What are default parameters in JavaScript?

Default parameters allow you to initialize function parameters with default values if no value is passed.

16. What is the spread operator in JavaScript?

The spread operator (...) allows an iterable to expand in places where multiple arguments are expected.

17. What is the rest parameter in JavaScript?

The rest parameter (...args) allows you to pass an indefinite number of arguments to a function.

18. What are promises in JavaScript?

Promises represent asynchronous operations that either resolve or reject. Example: new Promise((resolve, reject) => {}).

APIs

19. What is an API?

An API (Application Programming Interface) allows applications to communicate with each other, providing data and services.

20. What is the Fetch API in JavaScript?

The Fetch API is used to make network requests to retrieve resources from a server. Example: fetch('https://api.example.com/data').then(response => response.json()).

21. How do you handle errors in Fetch API requests?

Use .catch() to handle errors in a Fetch request, e.g., fetch(url).catch(error => console.error('Error:', error)).

22. What is JSON and how is it used with APIs?

JSON (JavaScript Object Notation) is a lightweight format for data interchange, commonly used to send and receive data in API requests.

23. What is the difference between GET and POST requests in APIs?

GET requests retrieve data, while POST requests send data to the server.

24. How do you send data in a POST request using Fetch API?

Use the fetch() function with the method set to POST and include data in the body. Example:

```
fetch(url, {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({key: 'value'})
});
```

25. What is CORS and how does it relate to APIs?

CORS (Cross-Origin Resource Sharing) is a security feature that restricts how resources on a web page can be requested from another domain.

PART 4: REACT INTERVIEW QUESTIONS (2025 EDITION)

Beginner Level (1–20)

1. What is React?

React is a JavaScript library for building reusable UI components. Example: Used for single-page apps (Facebook, Netflix, Instagram).

2. What are components in React?

Components are independent, reusable pieces of UI. Example: <Navbar />, <Button />, <ProductCard />.

3. What is JSX?

JSX stands for JavaScript XML — allows writing HTML inside JavaScript. Example: const element = <h1>Hello, World!</h1>;

4. What are props?

Props are inputs to components (like function parameters). Example: <User name="Mounika" />.

5. What is state in React?

State is mutable data that determines component behavior. Example: const [count, setCount] = useState(0);

6. Difference between props and state?

- **Props:** passed from parent, immutable
- **State:** managed inside component, mutable

7. What is the Virtual DOM?

A lightweight representation of the actual DOM for efficient updates. Example: React compares old vs new virtual DOM → updates only changed parts.

8. What is useState?

A React Hook to manage component state. Example: const [isOpen, setIsOpen] = useState(false);

9. What is useEffect?

Used for side effects (fetching data, event listeners, etc.). Example: useEffect(() => fetchUsers(), []);

10. What are keys in React lists?

Unique identifiers that help React track items efficiently. Example: key={user.id} in a .map().

11. What is conditional rendering?

Rendering UI based on conditions. Example: {isLoggedIn ? <Dashboard /> : <Login />}

12. What is React Fragment?

Lets you return multiple elements without an extra DOM node. Example:

```
<>
 <h1>Title</h1>
 <p>Text</p>
</>
```

13. What is one-way data binding?

Data flows from parent → child only (through props). Example: Parent sends data to child components.

14. How do you handle events in React?

Using camelCase and functions. Example: <button onClick={handleClick}>Click</button>

15. What is controlled vs uncontrolled component?

- **Controlled:** React controls input via state
- **Uncontrolled:** Uses the DOM directly Example: Controlled forms handle validation and UI updates.

16. What is the purpose of defaultProps?

Provides default prop values when not supplied.

17. What are React Hooks?

Functions that let you use state and lifecycle features in functional components.

18. What is the rule of Hooks?

Hooks must be called at the top level and only inside React functions.

19. What are higher-order components (HOC)?

Functions that take a component and return a new component. Example: withAuth(Component) for protected pages.

20. What is the difference between class and functional components?

Functional components use hooks, class components use lifecycle methods.

Intermediate Level (21–50)

21. What is the Context API?

Provides a way to share state globally without prop drilling. Example: Theme or authentication data.

22. What is useContext?

Hook to access context values directly. Example: const theme = useContext(ThemeContext);

23. What is useReducer?

Manages complex state logic like Redux. Example: const [state, dispatch] = useReducer(reducer, initialState);

24. What is React.memo()?

Prevents unnecessary re-renders if props don't change. Example: export default React.memo(MyComponent);

25. What is useCallback?

Memoizes a callback function to avoid re-creation. Example: const handleClick = useCallback(() => doSomething(), []);

26. What is useMemo?

Memoizes a computed value for performance. Example: const total = useMemo(() => sum(items), [items]);

27. What is React Router?

A library for SPA routing. Example: <Route path="/home" element={<Home />} />

28. Difference between <Link> and <a> tags?

<Link> prevents full page reloads (client-side routing).

29. What is lazy loading?

Dynamically loading components or data when needed. Example: const About = React.lazy(() => import('./About'));

30. What is Suspense in React?

Displays fallback UI while components are loading.

31. What are error boundaries?

Components that catch runtime errors in child components.

32. What is code splitting?

Splitting bundles to improve load times using dynamic import.

33. What is Prop Drilling?

Passing props through multiple layers unnecessarily. Solution: Use Context API or Redux.

34. What is React Portal?

Renders children into a DOM node outside the parent hierarchy. Example: Modals or tooltips.

35. What is hydration in SSR?

Attaching event listeners to server-rendered HTML on client load.

36. What are synthetic events?

Cross-browser wrapper around native events in React.

37. What is the difference between React and Angular?

React is a library (view layer); Angular is a full-fledged framework.

38. What is reconciliation in React?

React's algorithm that compares old and new virtual DOM trees.

39. What is controlled input in React forms?

Input values are synced with React state.

40. What is lifting state up?

Moving shared state to the nearest common ancestor.

41. What is StrictMode?

Highlights potential problems in app development (e.g., deprecated APIs).

42. What is useRef()?

Accesses DOM elements or stores mutable values. Example: Auto-focusing an input.

43. How to optimize React app performance?

- Use React.memo

- Code splitting
- Lazy loading
- Debouncing
- Avoid inline functions

44. What is React Fiber?

Internal engine that improves scheduling and rendering performance.

45. What is concurrent mode?

Enables React to interrupt rendering to keep UI responsive.

46. What is reconciliation algorithm (Diffing)?

Determines minimal DOM updates after state changes.

47. What are render props?

Passing a function as a child to share logic. Example: <Mouse>{pos => {pos.x} }</Mouse>

48. What is React Profiler?

Tool to measure performance and render frequency.

49. What are fragments useful for?

Grouping children without extra DOM elements.

50. What are React keys and why important?

Help React identify which items changed, added, or removed.

Advanced Level (51–75)

51. What is Redux?

Predictable state management library. Flow: Action → Reducer → Store → UI Update.

52. What is Redux Toolkit?

Simplified Redux setup with less boilerplate.

53. What is difference between Redux and Context API?

Redux handles complex state logic; Context suits small-scale global states.

54. What is React Query (TanStack Query)?

Library for server-state management (data fetching + caching).

55. What are Server Components?

New React 18+ feature for rendering parts of UI on server (improves speed).

56. What is useTransition() hook?

Marks state updates as non-urgent to keep UI smooth.

57. What is Suspense for data fetching?

Waits for async operations to complete before rendering.

58. What is useDeferredValue()?

Defers rendering of non-urgent updates (like filtering large lists).

59. What is useImperativeHandle?

Customizes ref handling between parent and child.

60. What is forwardRef()?

Passes refs from parent to child components.

61. What is a custom hook?

Function starting with "use" that encapsulates reusable logic. Example: useFetch(url) for data fetching.

62. What are portals used for?

To render modals or dropdowns outside the parent DOM hierarchy.

63. What is hydration error?

Mismatch between server-rendered HTML and client React tree.

64. What is Tree Shaking?

Removing unused code during bundling (Webpack, Vite).

65. What is Error Handling in async React apps?

Try/catch in async functions or error boundaries.

66. What is useSyncExternalStore()?

Hook for subscribing to external data sources (e.g., Redux).

67. What is useLayoutEffect() vs useEffect()?

- **useEffect:** runs after paint

- **useLayoutEffect:** runs before paint (use carefully)

68. What is hydration in Next.js?

The process of React attaching JS functionality to pre-rendered HTML.

69. What is StrictMode's purpose in React 18?

Simulates unmount/remount cycles to detect side effect bugs.

70. What are transitions and suspense boundaries?

Allow smoother loading experiences and background updates.

71. What are React Server Actions (React 19)?

Let you run async server functions directly from client components.

72. What is optimistic UI update?

Temporarily updating UI before server confirmation. Example: Adding a comment before API success response.

73. What is lazy hydration?

Hydrating only critical UI parts first for faster interaction readiness.

74. What is streaming SSR?

React sends chunks of HTML progressively for better performance.

75. What are micro-frontends in React?

Architecting apps as independently deployable frontend modules.

Real-World Scenario Questions

76. How would you handle API errors gracefully?

Use try/catch + show fallback UI + retry with exponential backoff.

77. How to improve initial page load time?

Use lazy loading, bundle splitting, CDN caching, and SSR (Next.js).

78. How do you manage large forms efficiently?

Use react-hook-form or Formik for controlled inputs and validation.

79. How to manage multiple theme modes (dark/light)?

Use Context or CSS variables + localStorage to persist user preference.

80. What would you use for global state management?

- Context for small apps
- Redux/RTK or Zustand for large apps
- React Query for server state

Document End