

Crack the Front-End Developer Interview

Top 30 Questions
You Must Know



HTML Basics

1. Difference between `<div>` and `<section>` tags in HTML?

- `<div>`: Generic container with no semantic meaning.
- `<section>`: Denotes a thematic grouping of content, often with a heading.

2. How to create a table in HTML, and when should you use it?

- Use `<table>` with `<tr>`, `<th>`, and `<td>` tags. Use for tabular data only.

```
<table>
  <tr>
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr>
```

```
    <td>John</td>
    <td>30</td>
  </tr>
</table>
```



3. What are semantic HTML tags, and why are they important?

- Tags like `<header>`, `<footer>`, `<article>` provide meaning to content and improve SEO and accessibility.

4. Purpose of the `alt` attribute in images?

- Describes the image for accessibility and is displayed if the image fails to load.

5. What are meta tags, and how do they affect a web page?

- Provide metadata about the page. E.g., `<meta charset="UTF-8">` sets the character encoding.

CSS Fundamentals

6. Difference between `id` and `class` selectors in CSS?

- `id`: Unique, applied to one element (`#id`).
- `class`: Reusable, applied to multiple elements (`.class`).

7. How does the CSS Box Model work?

- Includes **content**, **padding**, **border**, and **margin**. Controls spacing and layout.

8. How does the CSS Box Model work?

- **relative**: Position relative to itself.
- **absolute**: Positioned relative to nearest positioned ancestor.
- **fixed**: Position relative to the viewport.

9. Difference between **inline**, **block**, and **inline-block** elements?

- **inline**: No new line; width/height not respected (e.g., ``).
- **block**: Occupies full width (e.g., `<div>`).
- **inline-block**: Combines properties of both.

10. How do media queries work?

Used to apply styles based on device properties like width.

```
@media (max-width: 600px) {  
  body { font-size: 14px; }  
}
```

JavaScript Essentials

11. Difference between var, let, and const?

- **var**: Function-scoped.
- **let**: Block-scoped, reassignable.
- **const**: Block-scoped, not reassignable

12. Explain closures in JavaScript.

- A closure allows a function to access variables from its outer scope even after the outer function has returned.


```
function outer() {  
  let count = 0;  
  return function inner() {  
    count++;  
    return count;  
  };  
}  
  
const counter = outer();  
console.log(counter()); // 1  
console.log(counter()); // 2
```

13. What is the DOM?

- **DOM** (Document Object Model) represents a web page as a tree structure. It allows JavaScript to manipulate **HTML** and **CSS**.

14. What are arrow functions?

- Concise syntax for functions, no binding of **this**.

```
const add = (a, b) => a + b;
```

15. Explain event delegation

- Event delegation attaches a single event listener to a parent element to handle events for its children.

Advanced JavaScript Topics

16. Difference between synchronous and asynchronous programming?

- Synchronous: Tasks are executed sequentially.
- Asynchronous: Tasks can run in the background (e.g., `setTimeout`).

17. What is the Promise API?

- Used to handle asynchronous operations.


```
fetch('url').then(response => console.log(response));
```

18. Purpose of **async** and **await**?

- **async** declares an asynchronous function. **await** pauses execution until a **Promise** resolves

19. What are higher-order functions?

- Functions that take other functions as arguments or return them. E.g., **map**, **filter**.

20. What is hoisting in JavaScript?

- Variables and function declarations are moved to the top of their scope before code execution.



Front-End Frameworks and Libraries

21. What is the Virtual DOM?

- A lightweight copy of the DOM used in libraries like React for efficient updates.

22. Difference between two-way and one-way data binding?

- Two-way: Changes in UI and data affect each other (e.g., Angular).
- One-way: Data flows only from parent to child (e.g., React).

23. What are React hooks?

- Functions like `useState` and `useEffect` that let you manage state and side effects in functional components.

24. Purpose of useState and useEffect?

- **useState**: Manage state.
- **useEffect**: Handle side effects like API calls.

25. What is a component lifecycle?

- The phases (mounting, updating, unmounting) a component goes through in React or Vue.js.

Performance Optimization

26. Strategies for improving web app performance?

- Minify assets, lazy load images, use a CDN, optimize JavaScript.

27. What is lazy loading?

- Loading resources (e.g., images, scripts) only when needed.

28. What is critical CSS?

- CSS required for above-the-fold content, improving load speed.

29. How to handle large datasets efficiently?

- Use pagination, virtualization (e.g., react-window).

30. Purpose of a CDN?

- A network of servers that delivers content quickly to users based on their location.



Ready to Tame Recursion?

Upskill with  **Learnbay**

India's most trusted

Program For **Working Professional**

Data Structure Algorithms & System Design

With **Gen-AI** For Software Developers

Program electives:



GenAI



Product management



DevOps



Data engineering



Get Certification from :



IIT
Guwahati

WOLFF UNIVERSITY/



Microsoft