Frontend Developer Interview Guide

1. HTML Interview Questions

Basic Questions:

1. What is HTML, and why is it used?
2. Explain the difference between block-level and inline elements.
3. What is semantic HTML? Why is it important?
4. What is the purpose of the `alt` attribute in `<img>` tags?
5. How do you create a hyperlink in HTML?
6. What is the difference between `id` and `class` attributes?
7. What are the new features introduced in HTML5?
8. Explain the difference between `<section>`, `<div>`, and `<article>` tags.
9. How do you ensure SEO optimization using semantic HTML tags?

Advanced Questions:

1. How can you make a webpage more accessible using HTML?
2. What is the purpose of `<meta>` tags?
3. How do you use data attributes in HTML, and why are they useful?
4. Explain the concept of `iframes` and their use cases.
5. How does the `picture` element work in responsive design?
6. What is the difference between `<header>`, `<footer>`, and `<nav>` tags, and how do they contribute to semantic structure?

2. CSS Interview Questions

Basic Questions:

1. What is the difference between inline, internal, and external CSS?
2. What is the box model in CSS?
3. How do you add a background image to a webpage?
4. What is the difference between `relative`, `absolute`, and `fixed` positioning?
5. What are pseudo-classes and pseudo-elements?
6. What is the difference between `em` and `rem` units?
7. How do you apply styles specifically for print media?
8. What are media queries, and how do you use them for responsive design?

Advanced Questions:

1. Explain how CSS Grid and Flexbox differ.
2. How does the `z-index` property work, and what are stacking contexts?

3. What is the difference between `@import` and `<link>` in CSS?
4. How do you implement CSS animations?
5. What are keyframes in CSS, and how are they used?
6. Explain the difference between responsive design and adaptive design.
7. What are CSS custom properties (variables), and how do you use them?
8. How do you handle responsiveness for high-resolution displays (e.g., Retina screens)?

3. JavaScript Interview Questions

Basic Questions:

1. What is JavaScript, and how does it differ from Java?
2. What are the different data types in JavaScript?
3. Explain the difference between `let`, `var`, and `const`.
4. What is the difference between `==` and `===`?
5. How does scoping work in JavaScript?
6. What are closures, and how do they work?
7. What is the event loop in JavaScript?
8. What are promises, and how do you use them?
9. Explain the difference between synchronous and asynchronous programming.
10. What is the DOM, and how do you manipulate it with JavaScript?

Advanced Questions:

1. What is a prototype in JavaScript?
2. Explain how `this` works in different contexts.
3. What are arrow functions, and how are they different from regular functions?
4. What is hoisting in JavaScript, and how does it work?
5. Explain different types of loops in JavaScript (e.g., `for`, `while`, `do-while`, `forEach`).
6. What are higher-order functions? Provide examples.
7. What are callback functions, and how do you use them?
8. What is callback hell, and how can it be avoided?
9. What is promise chaining, and how does it compare to callback functions?
10. What is promise hell, and how do `async` and `await` help mitigate it?
11. How do `try...catch` blocks work in JavaScript?
12. Explain the difference between `.map()`, `.forEach()`, and `.reduce()` methods.
13. How do `setTimeout` and `setInterval` work in JavaScript?
14. What is event delegation, and why is it useful?
15. Explain the difference between deep copy and shallow copy in JavaScript.
16. What are JavaScript modules, and how do they work?
17. What is pass-by-value and pass-by-reference in JavaScript?
18. How does destructuring assignment work in JavaScript?
19. What are generator functions and how do they work?

20. Explain how the spread and rest operators function in JavaScript.

JavaScript Code Challenges:

Write a function to flatten a nested array.

```
 function flattenArray(arr) {
   return arr.reduce((flat, current) => flat.concat(Array.isArray(current) ? flattenArray(current) :
current), []);
}
console.log(flattenArray([1, [2, [3, 4], 5]])); // Output: [1, 2, 3, 4, 5]
```

1.

Implement a debounce function in JavaScript.

```
 function debounce(func, delay) {
   let timer;
   return function (...args) {
      clearTimeout(timer);
      timer = setTimeout(() => func.apply(this, args), delay);
   };
}
```

2.

Create a function to fetch data using `async/await` and handle errors with `try...catch`.

```
 async function fetchData(url) {
   try {
      const response = await fetch(url);
      if (!response.ok) throw new Error('Network error');
      const data = await response.json();
      return data;
   } catch (error) {
      console.error('Error:', error);
   }
}
```

3.

4. TypeScript Interview Questions

Basic Questions:

1. What is TypeScript, and how is it different from JavaScript?
2. What are interfaces in TypeScript, and how do you use them?
3. What are enums in TypeScript?
4. What are generics in TypeScript, and why are they used?
5. How does TypeScript handle optional properties?
6. What are type assertions in TypeScript?
7. What is the purpose of the `readonly` modifier?
8. How do you handle null and undefined in TypeScript?

Advanced Questions:

1. What is the `unknown` type in TypeScript, and how does it differ from `any`?
2. Explain TypeScript decorators.
3. What are utility types in TypeScript?
4. How does TypeScript support mixins?
5. How do you handle modules and namespaces in TypeScript?
6. What is the difference between `interface` and `type`?
7. How do you implement type guards in TypeScript?
8. How does TypeScript support function overloading?

Write a function to handle both string and number input using generics.

```
 function combine<T extends string | number>(a: T, b: T): T {
   return (typeof a === "string" && typeof b === "string" ? a + b : (a as number) + (b as number))
as T;
}
```

9.
10. How does TypeScript handle mapped types?
11. How do you define and use union types in TypeScript?
12. What are TypeScript literal types, and when are they useful?

TypeScript Code Challenges:

1. Implement a TypeScript function to calculate the factorial of a number using recursion.
2. Create a TypeScript interface for a `Product` object with properties `name`, `price`, `description`, and `category`.
3. Write a TypeScript class for a basic calculator with methods for addition, subtraction, multiplication, and division.

5. React JS Interview Questions

Basic Questions:

1. What is React, and why is it used?
2. What are components in React?
3. What is the difference between functional and class components?
4. What are props in React?
5. What is state in React, and how is it managed?
6. What is the virtual DOM?
7. How does React handle events?

React Hooks Questions:

1. What are hooks in React, and why were they introduced?
2. Explain `useState` and provide an example.
3. How does `useEffect` work, and what are its use cases?
4. What is `useContext`, and how does it simplify prop drilling?
5. Explain `useReducer` and how it differs from `useState`.
6. What is `useMemo`, and when would you use it?
7. What is `useCallback`, and how does it optimize rendering?
8. How does `useRef` work, and what are its practical use cases?
9. What is `useLayoutEffect`, and how does it differ from `useEffect`?
10. What is `useImperativeHandle`, and how is it used?
11. Explain `useId` and its purpose in React.
12. What are custom hooks, and how do you create one?
13. What is the difference between `useSyncExternalStore` and `useContext`?
14. How do you manage side effects in React effectively?

Advanced Questions:

1. How do you optimize React applications?
2. What is context API, and how do you use it?
3. What are higher-order components (HOCs)?
4. How do you handle forms in React?
5. What are the benefits of using React's Error Boundaries?
6. What is React Suspense, and how is it used with concurrent features?

6. Next.js Interview Questions (App Router Focus)

Basic Questions:

1. What is Next.js, and why is it popular?
2. What are the main features of the Next.js App Router introduced in version 14?
3. How does routing work in the App Router?
4. What are the benefits of server components in Next.js App Router?
5. Explain how layouts are managed in the App Router.

Advanced Questions:

1. How does data fetching work in the Next.js App Router? (e.g., `fetch`, `use`, and `react-query`)
2. How do you implement caching strategies in the Next.js App Router?
3. What are loading states in the App Router, and how are they managed?
4. How do you handle dynamic routing in the Next.js App Router?
5. How do you implement middleware in Next.js for request handling?
6. What are the differences between `pages` and `app` directories in Next.js?
7. How do you optimize images and assets in the App Router?
8. Explain the role of `head.js` in the App Router.
9. How do you configure nested layouts in the App Router?
10. How do you handle streaming data in the Next.js App Router?
11. What is the role of React Server Components in Next.js, and how do they improve performance?
12. How do you manage state in a server component-based architecture?