JS Must-Know Topics

For advanced JavaScript interviews, interviewers often focus on concepts that demonstrate a deep understanding of the language, its runtime, and its advanced features. Here's a curated list of key JavaScript topics that might be covered in interviews for experienced developers:

---

1. Event Loop and Concurrency

- **Key Topics:**
  - How the **Event Loop** works.
  - Understanding the **Call Stack**, **Web APIs**, **Task Queue**, and **Microtask Queue**.
  - Differences between **microtasks** (`Promise.then`, `MutationObserver`) and **macrotasks** (`setTimeout`, `setImmediate`).
  - Real-world implications of asynchronous execution order.
- **Example Questions:**
  - Explain the Event Loop with an example.
  - What is the difference between microtasks and macrotasks, and when do they execute?

---

2. Closures

- **Key Topics:**
  - Definition and creation of closures.
  - Common use cases like function factories, memoization, and private variables.
  - How closures work with loops (`var` vs `let`).

- **Example Questions:**
  - How do closures retain access to variables even after the outer function has executed?
  - Write a function that generates a counter using closures.

---

3. Scope and Hoisting

- **Key Topics:**
  - **Lexical Scope** and **Dynamic Scope**.
  - The **Scope Chain**.
  - **Hoisting** behavior of `var`, `let`, and `const`.
  - Temporal Dead Zone (TDZ).
- **Example Questions:**
  - How is the scope chain resolved during execution?
  - What is hoisting, and how does it differ between `var`, `let`, and `const`?

---

4. Prototypes and Inheritance

- **Key Topics:**
  - Understanding **Prototypal Inheritance** and the `prototype` chain.
  - `__proto__` vs `prototype`.
  - ES6+ Class Syntax and how it uses prototypes under the hood.
  - Object creation using `Object.create()`.
- **Example Questions:**
  - How does prototypal inheritance work in JavaScript?
  - What happens if a property is not found in an object? How is it resolved?

---

5. Asynchronous Programming

- **Key Topics:**

- Promises: chaining, error handling, combinators (`Promise.all`, `Promise.race`, etc.).
- `async/await` and error handling.
- Using `setTimeout`, `setInterval`, and `requestAnimationFrame`.
- Handling race conditions and deadlocks.

- **Example Questions:**
  - What is the difference between `async/await` and Promises?
  - How do you cancel a `setTimeout` operation?

---

6. Functional Programming Concepts

- **Key Topics:**
  - Higher-order functions (`map`, `filter`, `reduce`).
  - **Immutability** and side effects.
  - Function currying and composition.
  - Pure functions and lazy evaluation.
- **Example Questions:**
  - Write a custom implementation of `Array.prototype.map`.
  - Explain currying with an example.

---

7. Modules

- **Key Topics:**
  - ES6 Modules (`import` and `export`) vs CommonJS (`require`, `module.exports`).
  - Named vs Default Exports.
  - Module scope and singleton behavior.
- **Example Questions:**
  - How does tree-shaking work in ES6 modules?
  - What are the differences between `require` and `import`?

8. Error Handling

- **Key Topics:**
  - `try...catch` and `finally` blocks.
  - Custom error objects using `class` and `extends`.
  - Error propagation in asynchronous code.
- **Example Questions:**
  - How do you handle errors in Promises vs `async/await`?
  - Write a custom error class.

---

9. Memory Management

- **Key Topics:**
  - Garbage Collection and how it works.
  - Concepts like **Memory Leaks** and how to prevent them.
  - WeakMap and WeakSet for managing object references.
- **Example Questions:**
  - What are common sources of memory leaks in JavaScript?
  - How does garbage collection work in V8?

---

10. Object-Oriented Programming (OOP)

- **Key Topics:**
  - Creating objects: Constructor functions, `class`, and `Object.create`.
  - Encapsulation, Polymorphism, and Inheritance.
  - ES6+ Class syntax and static methods.
- **Example Questions:**
  - What's the difference between `Object.create()` and a constructor function?
  - How do `getters` and `setters` work in JavaScript?

11. Advanced Array and Object Methods

- **Key Topics:**
  - Array methods: `reduce`, `some`, `every`, `find`, `flatMap`.
  - Object methods: `Object.assign`, `Object.entries`, `Object.keys`, `Object.values`.
  - Destructuring and spreading/rest operators.
- **Example Questions:**
  - Write a polyfill for `Array.prototype.reduce`.
  - Explain the use cases of the rest/spread operator in objects.

---

12. Event Handling

- **Key Topics:**
  - Event delegation and bubbling/capturing.
  - Removing event listeners.
  - Synthetic events in libraries like React.
- **Example Questions:**
  - How does event delegation improve performance?
  - Write an example of event delegation using vanilla JavaScript.

---

13. JavaScript Runtime and Environment

- **Key Topics:**
  - How JavaScript runs in browsers vs Node.js.
  - Global objects like `window` and `global`.
  - Modules and IIFE (Immediately Invoked Function Expression).
- **Example Questions:**
  - What is the difference between `window` and `global`?
  - Why is `use strict` useful?

14. Performance Optimization

- **Key Topics:**
  - Debouncing and throttling.
  - Avoiding blocking operations.
  - Using Web Workers for parallel tasks.
- **Example Questions:**
  - Implement a debounce function.
  - How do you handle computationally heavy tasks in JavaScript?

---

15. Type Coercion

- **Key Topics:**
  - Implicit vs explicit type conversion.
  - Truthy and falsy values.
  - Comparing `==` vs `===`.
- **Example Questions:**
  - Why does `[] == ![]` evaluate to true?
  - What's the difference between `null` and `undefined`?

---

16. Regular Expressions

- **Key Topics:**
  - Basic syntax: patterns, flags, and groups.
  - Common use cases: validation, searching, and replacing.
- **Example Questions:**
  - Write a regex to validate email addresses.
  - How do you replace multiple spaces in a string with a single space using regex?

---

17. Advanced Topics

- **Key Topics:**
  - `Proxy` and `Reflect`.
  - Generators and iterators.
  - Dynamic imports and lazy loading.
  - Symbol and its use cases.
- **Example Questions:**
  - What are `Proxy` and `Reflect`, and how do you use them?
  - Explain the use of `Symbol.iterator` in custom objects.

---

18. Testing and Debugging

- **Key Topics:**
  - Unit testing with frameworks like Jest or Mocha.
  - Debugging tools in modern browsers.
  - Writing mock functions and handling async tests.
- **Example Questions:**
  - How do you test an asynchronous function?
  - What debugging tools do you use for JavaScript?

---

19. Design Patterns in JavaScript

- **Key Topics:**
  - Common patterns like Singleton, Factory, Observer, and Module.
  - Implementing patterns in both ES5 and ES6+ syntax.
- **Example Questions:**
  - Write a simple implementation of the Observer pattern.
  - How does the Module pattern work in JavaScript?

---

20. Web APIs

- **Key Topics:**
  - Fetch API and handling network requests.
  - DOM manipulation and traversal.
  - Browser storage options: `localStorage`, `sessionStorage`, `IndexedDB`.
- **Example Questions:**
  - How do you fetch and handle JSON data from an API?
  - What's the difference between `localStorage` and `sessionStorage`?