Can you explain the key differences between Java and JavaScript — especially in terms of typing, runtime behavior, and use cases?”

"Java is a statically typed, compiled, object-oriented language primarily used for large-scale backend systems and Android apps.  
JavaScript, on the other hand, is a dynamically typed, interpreted scripting language mainly used for interactive web development."

* “In Java, variable types are fixed once declared, and errors are caught before execution. In JavaScript, variables can change type at any time — which offers flexibility but can lead to runtime bugs.”

**Runtime and Compilation**

* **Java**:
  + Compiled into bytecode by java C
  + Runs on the JVM (Java Virtual Machine)
  + Strong emphasis on performance and memory management
* **JavaScript**:
  + Interpreted or JIT compiled (e.g., by V8 engine in Chrome/Node.js)
  + Runs in browser or Node.js
  + Prioritizes responsiveness and rapid development

**Real-World Use Cases**

* **Java**:
  + Banking, enterprise, Android apps, backend microservices
* **JavaScript**:
  + Frontend UIs, interactive websites, real-time apps with Node.js

**How does dynamic typing in JS affect debugging?"**

**Answer:** “Dynamic typing means a variable’s type can change during execution, which can lead to bugs that are **not caught until runtime**. For example, you might expect a number but accidentally get a string — and only notice when a function misbehaves or breaks.”

A black screen with white text

AI-generated content may be incorrect.

**2. "What’s a runtime error you’ve encountered in JS due to dynamic typing?"**

**Answer:**

“I once passed a user object instead of a string to a function expecting a name. JS didn’t throw an error — it just converted [object Object] to a string silently, which broke the UI. The issue wasn’t caught until much later.”

**3. "Can you show how type coercion works in JS?"**

**Answer:**

“Type coercion is when JavaScript **automatically converts** one data type to another. This often happens in operations like +, ==, or with Boolean checks.”

**4. "Why doesn’t JavaScript have interfaces like Java?"**

**Answer:**

“JavaScript is dynamic and loosely typed, so it doesn’t enforce type contracts at compile time. Instead of formal interfaces, JS relies on **duck typing** — if it walks and quacks like a duck, it’s a duck.”

But developers can mimic interfaces using:

* **JSDoc comments**
* **TypeScript (which adds interfaces)**
* **Manual runtime checks**

**5. "If JavaScript is dynamic, how do TS or linters help solve the problem?"**

**Answer:**“TypeScript adds static typing to JavaScript — so types are checked at compile time. Linters like ESLint enforce coding rules and catch **bad patterns**, unused vars, or potential bugs.”