Good understanding of JavaScript/TypeScript (ES6+, async/await, modules)

A good understanding of modern JavaScript (ES6+) and TypeScript involves proficiency in key features for writing cleaner, more efficient, and robust code, particularly in managing **asynchronous operations** and organizing code into **modules**.

Core Concepts

ES6+ (ECMAScript 2015 and later): This is the modern standard for JavaScript, introducing significant enhancements that improve code readability and structure. Key features include:

`let` and `const`: Block-scoped variable declarations that prevent common bugs associated with the older `var` keyword.

Arrow Functions: A more concise syntax for writing functions, which also provides a more intuitive handling of the `this` keyword.

Classes: Syntactical sugar over JavaScript's existing prototype-based inheritance, making object-oriented programming more familiar to developers from other languages.

Destructuring: A convenient way to extract values from arrays or properties from objects into distinct variables.

Promises: Objects representing the eventual completion (or failure) of an asynchronous operation and its resulting value, which help manage "callback hell".

`async` / `await`: Introduced in ES2017, `async` / `await` is built on top of Promises and provides a syntax that makes asynchronous code look and behave more like synchronous code, making it much easier to read and debug.

The `async` keyword declares a function that always returns a Promise.

The `await` keyword is used *inside* an `async` function to pause execution until a Promise settles (resolves or rejects), returning the resolved value.

Proper error handling is achieved using standard `try...catch` blocks within `async` functions.

Modules (`import` / `export`): ES6 introduced a native, standardized module system for organizing code into separate, reusable files.

`export` statements are used to share functions, objects, or variables from a module file.

`import` statements are used to access those exports in other files.

This modular approach helps in building scalable applications and managing dependencies efficiently.

TypeScript: As a statically typed superset of JavaScript, TypeScript builds on ES6+ syntax and adds powerful features like type safety, type inference, and type annotations. This allows errors related to types to be caught at compile time rather than runtime, leading to more robust and maintainable code, especially in large-scale applications

---

Revision #2
Created 29 October 2025 02:43:34 by AI API
Updated 19 November 2025 05:33:41 by AI Channel