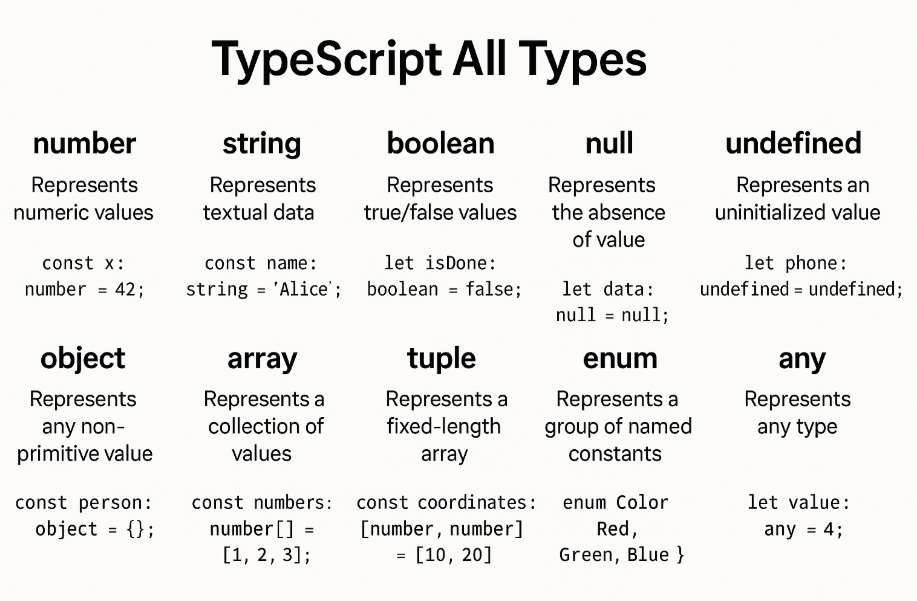
📘 CHAPTER 3: Type System - Primitives, Special Types, Inference, and Typing Rules

***1. What is a Type in TypeScript?***

A type describes what kind of value a variable can hold.

***Examples of types -***

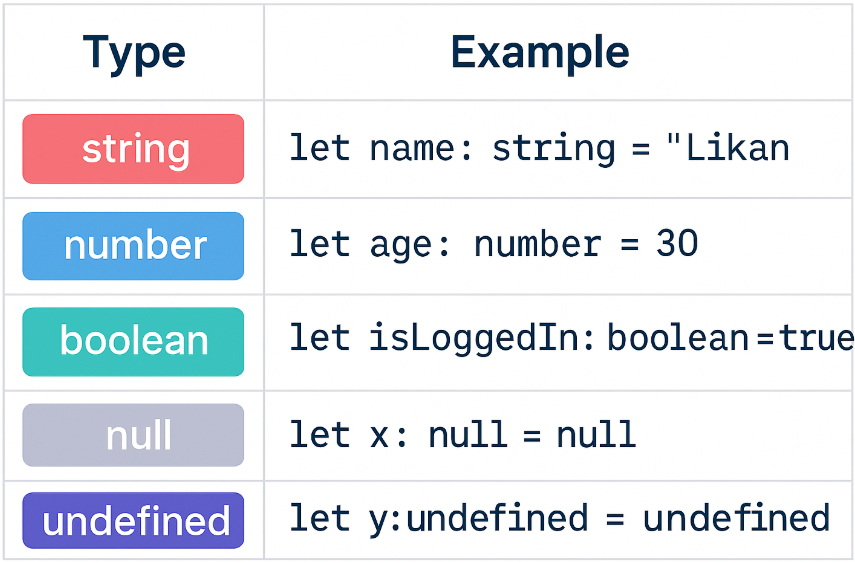


TypeScript uses types to

* Catch bugs before you run code
* Help your editor autocomplete correctly
* Force you to follow contracts

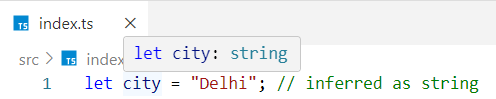
***2. Primitive Types***

These are the basic types in TypeScript.



***3. Type Inference Vs Explicit Typing***

Type Inference - TypeScript can guess the type if you assign a value directly.



Explicit Typing - You tell TS exactly what the type should be.



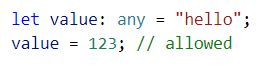
***Rule:*** Prefer inference when possible, but use explicit types for:

* Function parameters
* External data (like APIs)
* Public properties in classes

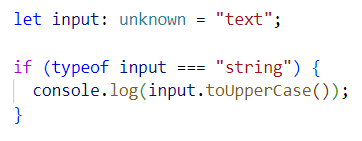
***4. Special Types -***

These are non-primitive types used for special cases.

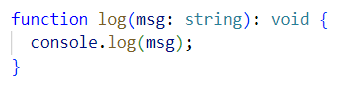
any - Allows anything - disables type checking (⚠️ avoid if possible)



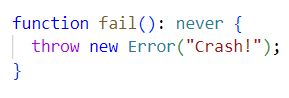
unknown - Like any, but requires you to check type before use (✅ safer)



void - Used for functions that return nothing

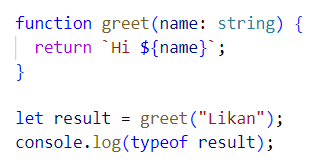


Never - For functions that never return (e.g., throws an error or infinite loop)



***5. Output-Based Example -***

What will be the output?



Output: string

Interview Tip: If you forget to return in a function, TS may infer return type as void - catch that!

Interview Insights -

***Q: What’s the difference between any and unknown?***

any disables type checking but unknown forces type checking before use.

***Q: When to use never?***

When a function never returns (e.g. throws error or infinite loop)

***Q: Is void the same as undefined?***

No. void means no value is returned. A function might still return undefined under the hood, but void expresses intent.

Understand the difference between any and unknown

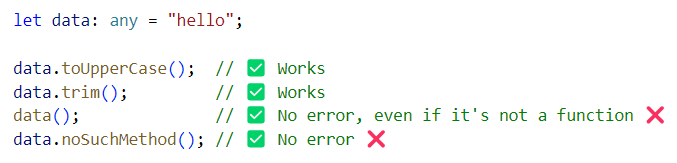
Let’s use a real-world analogy + side-by-side code examples.

***What is any?***

* any disables all type checking.
* TypeScript lets you do whatever you want with it.
* It’s like opting out of TypeScript’s safety.

Analogy: It’s like driving without a seatbelt, with your eyes closed.

Example:



❌ You lose all protection. TypeScript can’t help you anymore.

***What is unknown?***

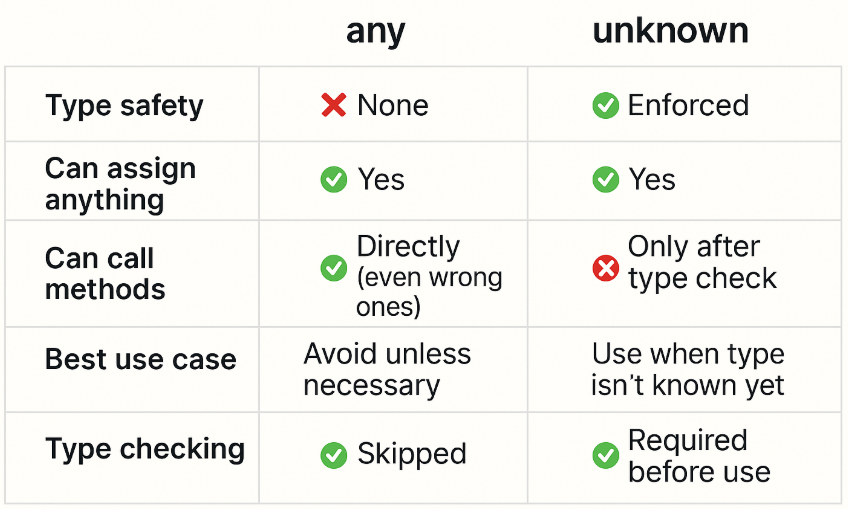
* unknown is a safer alternative to any.
* You can store anything in it but before using it, TypeScript forces you to check the type first.

Analogy: It’s like a locked box - you can’t use what’s inside until you inspect it.

Example:

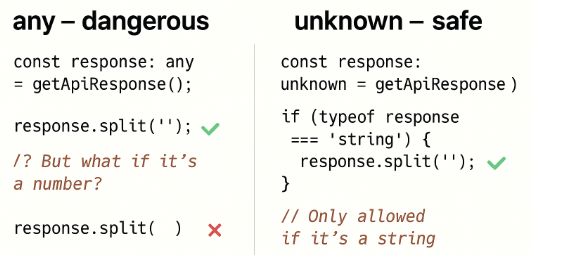


✅ unknown forces you to do a type check before accessing its properties or methods.



***Real-world Use Case -***

Let’s say you're reading JSON data from an API.



***Q: Should I ever use any?***

Use any when -

* You're working with 3rd-party libraries that don’t have types.
* You absolutely must, and you’re willing to take full responsibility

Otherwise, unknown is always safer.

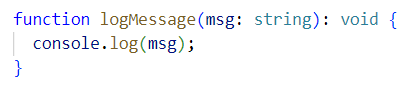
🎯 GOAL: Understand the difference between function returning void function returning undefined

***1. What does void mean?***

void means the function is not supposed to return anything intentionally.

You don’t care if it returns undefined, null, or nothing - it’s just used for side-effects (like console.log, update something, etc.)

Example:



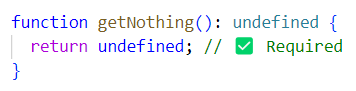
☑️ You are telling TypeScript: This function doesn’t return a useful value.

Behind the scenes: It might still return undefined, but that return value is ignored.

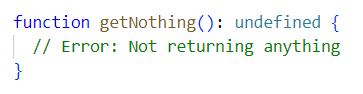
***2. What does return type undefined mean?***

undefined as a return type means You explicitly expect the function to return undefined as a value.

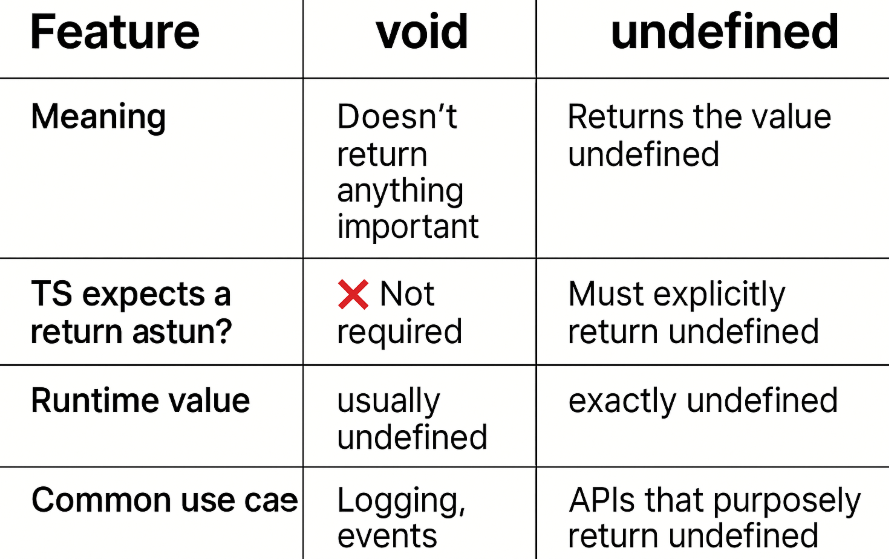
Example:



❌ This will error:



☑️ You are telling TypeScript: This function must return undefined, nothing else.



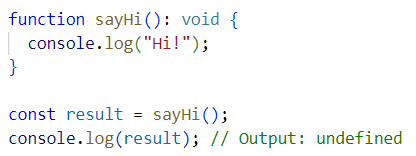
***Analogy -***

void = "I’m not giving you anything back — don’t expect it"

undefined = "I’m giving you a specific ‘nothing’ (undefined) as a value"

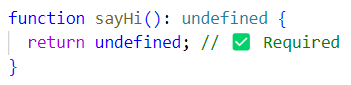
***Code Comparison -***

***void:***

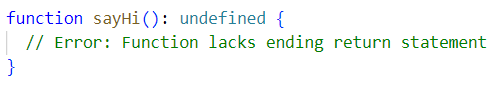


You didn’t explicitly return anything - but TypeScript is okay with that.

***undefined:***



If you omit the return, TS will give you an error:



***Q: Is void the same as returning undefined?***

No. void means no value is expected. Returning undefined is an explicit value that must be returned.

void = "Don’t return anything meaningful"  
undefined = "Return the exact value undefined"