📘 Chapter 10: Discriminated Unions

***What is a Discriminated Union?***

A **Discriminated Union** (also called a **Tagged Union**) is a TypeScript pattern where:

* ✅ You define a **union of object types** → Type1 | Type2 | ...
* ✅ All object types share a **common property** (known as the **discriminant** or **tag**)
* ✅ Each type assigns a **unique literal value** to this property

TypeScript uses the discriminant to **automatically narrow the type** within control structures like if or switch.

### *Why is it called a "Discriminated" Union?*

It’s called **Discriminated** because:

* All types in the union have **one common property** (like status)
* That property has a **different fixed value** in each type
* TypeScript uses this value to **know which type you're working with**

TypeScript uses the discriminant’s value to identify one type apart from another within a union.

This is also known as a Tagged Union in other languages like Rust and Swift.

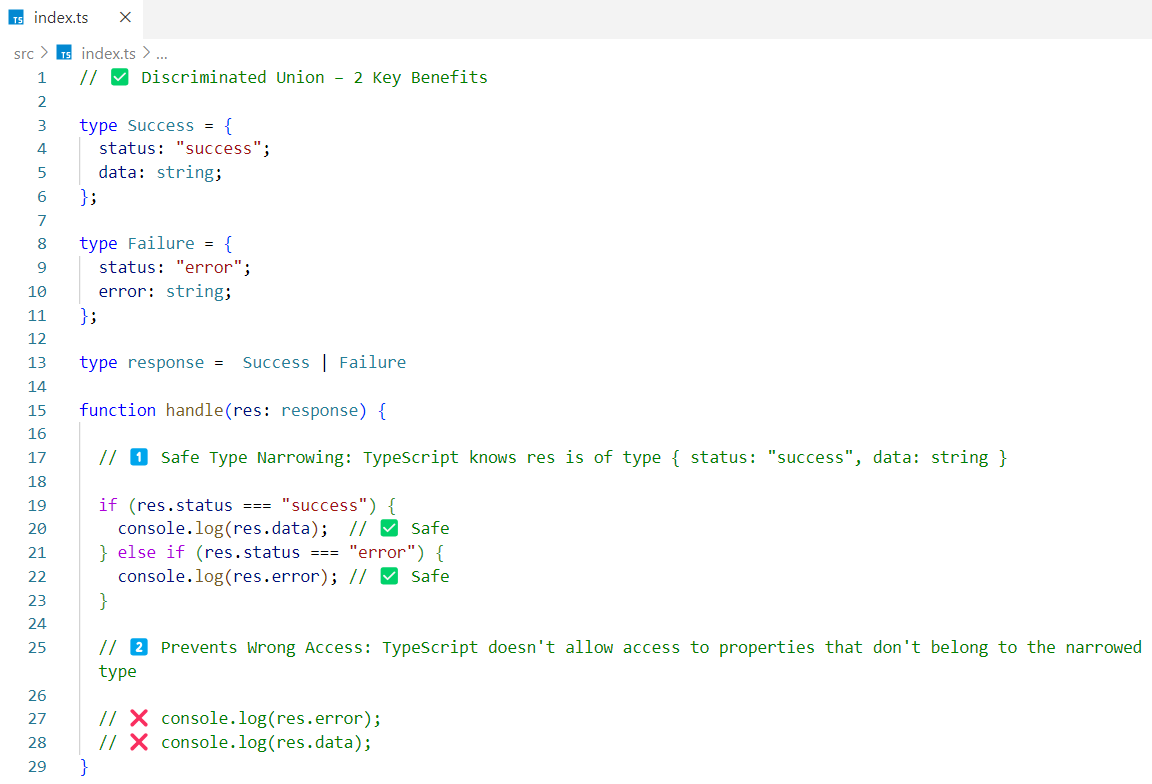
***What problems does it solve?***

#### **Safe Type Narrowing -** TypeScript automatically detects the correct type based on a shared property (like status).No need to manually check properties or use type casting.

#### **Prevents Wrong Property Access -** You can't accidentally access a property that doesn't belong to the current type. TypeScript blocks invalid access at compile time.

#### **Cleaner, More Predictable Code Flow -** You don’t need to use "in" checks or type casting - just simple if or switch.

#### Makes your logic easier to read and maintain.

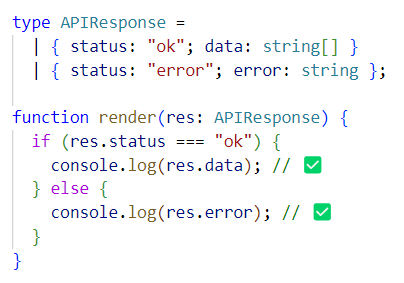


**What’s the Discriminant Here?**

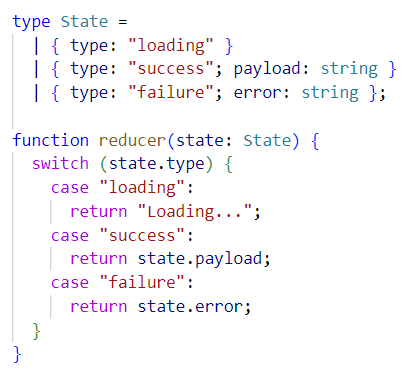
It's the status key. Each type has status but with different literal values: "success" or "error".

***Real-World Use Cases***

1. API Response Handling



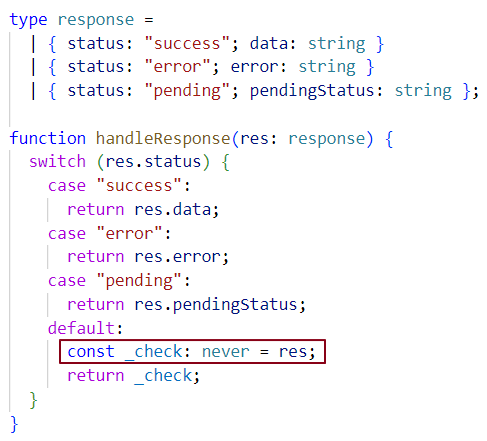
2. Redux State Management



***Exhaustiveness Check with never -***

**What is never type?**

The never type represents a value that should **never exist** or apart of code that should **never be reached**. If the code **is reachable**, TypeScript throws a **compile-time error** - because you missed handling a case.



**If we omit even one case,** then **the never block becomes reachable**, and we get a **compile-time error**. That’s the whole purpose of using never — to make sure the developer **handles all possible cases** in the union.

Interview insights

***Q1: What is a Discriminated Union in TypeScript?***

It’s a union of object types that share a common property (called a discriminant or tag) with unique literal values. TypeScript uses this to narrow types automatically.

***Q2: Why are discriminated unions useful?***

✅ They enable safe type narrowing  
✅ Allow exhaustive checks  
✅ Make code clean and self-documenting

***Q3: Can you give a real-world use case?***

API responses. For example, a status key with "success" or "error" tells you what fields (data or error) are available.

***Q4: What’s the difference between a union and a discriminated union?***

A **basic union** holds multiple unrelated types and needs **manual checks,**  
while a **discriminated union** uses a shared property (tag) to **auto-narrow** types safely.

***Q5: What is the role of the never type in discriminated unions?***

It helps ensure all cases in a switch are handled. If one is missing, TypeScript throws an error during compilation.

***Q6: Is this pattern specific to TypeScript?***

No - other languages like Rust, Swift, and OCaml use this too. But TypeScript’s implementation is especially useful for JS devs.