## **Null y Undefined en TypeScript: Una Breve Explicación**

En TypeScript, null y undefined son dos valores primitivos que representan la ausencia de un valor. Aunque pueden parecer similares, tienen matices diferentes:

* **Undefined:** Indica que una variable ha sido declarada pero no se le ha asignado ningún valor. Por ejemplo, si declaras una variable pero no la inicializas, su valor será undefined.
* **Null:** Representa la ausencia intencional de un valor. Es decir, tú asignas explícitamente el valor null a una variable para indicar que no tiene un valor válido en ese momento.

### **¿Por qué son importantes?**

* **Seguridad de tipos:** TypeScript te permite definir tipos más precisos para tus variables, lo que ayuda a prevenir errores comunes en tiempo de ejecución. Al entender null y undefined, puedes asegurarte de que tus variables siempre tengan el tipo de valor que esperas.
* **Manejo de errores:** Estos valores son fundamentales para manejar situaciones en las que un valor puede estar ausente, como cuando intentas acceder a una propiedad de un objeto que no existe.
* **Opcionalidad:** En TypeScript, puedes marcar propiedades de objetos como opcionales utilizando el signo de interrogación (?). Esto significa que la propiedad puede tener un valor o ser null o undefined.

### **Ejemplos Prácticos**

TypeScript

// Undefinedlet nombre; // Declarada pero no inicializada, su valor es undefinedconsole.log(nombre); // Imprime: undefined// Nulllet edad: number | null = null; // Indicamos que puede ser un número o nullconsole.log(edad); // Imprime: null// Opcionalidadinterface Persona { nombre: string; edad?: number; // Edad es opcional } let persona1: Persona = { nombre: "Juan" }; console.log(persona1.edad); // Imprime: undefined (la propiedad no existe)// Manejo de erroresfunction saludar(nombre: string): string { if (nombre) { return `Hola, ${nombre}!`; } else { return "Hola, desconocido!"; } } console.log(saludar(null)); // Imprime: Hola, desconocido!

Use code [with caution.](file:///C:/faq#coding)

### **Diferencias Clave entre Null y Undefined**

|  |  |  |
| --- | --- | --- |
| **Característica** | **Undefined** | **Null** |
| **Asignación** | Automática cuando no se inicializa una variable | Se asigna manualmente |
| **Significado** | Valor no asignado | Ausencia intencional de valor |
| **Tipo** | undefined | object (en JavaScript, pero TypeScript suele distinguirlos) |

Export to Sheets

**En resumen:**

* **Undefined** es como un espacio vacío que aún no se ha llenado.
* **Null** es como un espacio vacío que conscientemente has decidido dejar vacío.

### **Consideraciones Adicionales**

* **StrictNullChecks:** TypeScript tiene una opción de compilador llamada strictNullChecks que te obliga a manejar null y undefined de forma más explícita, lo que ayuda a prevenir errores.
* **Operador de fusión nula (??):** Este operador te permite proporcionar un valor por defecto si una expresión es null o undefined. Por ejemplo: let resultado = nombre ?? "Anónimo";.