TypeScript – null vs. undefined

1. Problem Statement

Case Study: The Real-Time User Profile Dashboard

You are building a real-time user profile dashboard for a social platform.

- Some users may not have filled out all their profile details (like age or bio).
- Sometimes, you need to show that a value is "intentionally empty" (e.g., user hasn't set their age), while other times a value is simply "not yet set" (e.g., waiting for data from the server).
- Your code must handle both situations safely, avoiding runtime errors and making it clear to other developers what each value means.

The challenge:
How do you use `null` and `undefined` in TypeScript to clearly represent the difference between "no value" and "not yet set," and how do you handle these values in user data and logic?

2. Learning Objectives

By the end of this lesson, you will be able to:

- Understand the difference between `null` and `undefined` in TypeScript.
- Know when to use `null` vs. `undefined` for variables, function returns, and object properties.
- Safely check, assign, and handle both values in your code.
- Model real-world data where some values may be missing or intentionally empty.

3. Concept Introduction with Analogy

Analogy: Empty Seats vs. Unassigned Seats

Imagine a theater:

- **An empty seat (null):** The seat exists, but nobody is sitting there-it's intentionally left empty.
- **An unassigned seat (undefined):** The seat hasn't been assigned yet-maybe the ticket system hasn't finished processing.

In your dashboard, `null` means "this field is intentionally empty," while `undefined` means "this field hasn't been set yet."

4. Technical Deep Dive

What is `null`?

- `null` is a primitive value representing "no value" or "empty."
- Explicitly assigned to a variable to show it is intentionally empty.
- TypeScript treats `null` as its own type.

Example:

```
let a: null = null;
console.log(a); // Output: null
console.log(typeof a); // Output: object
```

Reinitializing with null:

```
let age: number | null = null;
console.log(age); // null
age = 28;
console.log(age); // 28
```

What is `undefined`?

- `undefined` means a variable has been declared but not assigned a value.
- TypeScript automatically assigns `undefined` to uninitialized variables.
- Can also be explicitly assigned.

Example:

```
let b: undefined = undefined;
console.log(b); // Output: undefined
console.log(typeof b); // Output: undefined
```

Uninitialized variable:

```
let c:  number; console.log(c);  // Output: undefined
```

Function with no return:

```
function greet(name: string): void {
  console.log(`Hello ${name}`);
}
let result = greet("Alice");
console.log(result); // Output: undefined
```

Key Differences Table

Feature	null	undefined
Meaning	Explicitly no value	Not initialized
Typical Use	Intentionally empty/absent value	Variable declared but not assigned
Type Annotation	Has its own type `null`	Has its own type `undefined`
Default Behavior	Does not trigger default params	Triggers default params

Feature	null	undefined
Function Parameters	Explicitly no value	Missing or optional parameters
Object Properties	Deliberately set to no value	May not be initialized
Operational Handling	Must be handled in logic	Often handled with defaults

5. Step-by-Step Data Modeling & Code Walkthrough

1. **User type with nullable and optional properties:**

```
type User = {
  name: string;
  age: number | null;
  email?: string;
};
```

2. **User with null and undefined properties:**

```
let user1: User = {
 name: "John Doe",
 age: null, // Explicitly no age
 email: "john@example.com"
};

let user2: User = {
 name: "Jane Doe",
 age: 25
 // email is optional and thus undefined
};
```

3. **Checking and handling values:**

```
function printUser(user: User): void {
  let ageInfo = user.age === null ? "Age not provided" : `Age: ${user.
  let emailInfo = user.email ? `Email: ${user.email}` : "Email not set
  console.log(`${user.name} - ${ageInfo}, ${emailInfo}`);
}
```

```
printUser(user1); // John Doe - Age not provided, Email: john@example.
printUser(user2); // Jane Doe - Age: 25, Email not set
```

6. Interactive Challenge

Your Turn!

- Define a type `Profile` with `username` (string), `bio` (string or null), and optional `avatarUrl` (string).
- Create two profiles: one with a null bio and no avatar, and one with both fields set.
- Write a function `showProfile` that prints the username, a default message if bio is null, and a default avatar if `avatarUrl` is undefined.

7. Common Pitfalls & Best Practices

- **Don't confuse `null` and `undefined`:** Use `null` for intentional emptiness, `undefined` for "not set yet."
- **Always check for both `null` and `undefined` when handling optional or missing values.**
- **Use union types (`type | null`) for fields that can be empty.**
- **Use optional properties (`prop?: type`) for fields that may be missing.**

8. Quick Recap & Key Takeaways

- `null` means "intentionally empty"; `undefined` means "not set yet."
- Use `null` for fields that are deliberately empty, and `undefined` for optional or not-yet-initialized values.
- Always check and handle both cases in your logic for safe, predictable code.

9. Optional: Programmer's Workflow Checklist

- Use `null` for fields that are intentionally empty.
- Use `undefined` for optional or not-yet-initialized fields.
- Always check for both `null` and `undefined` before using a value.
- Use union types and optional properties for flexibility.
- Provide sensible defaults when displaying or using possibly missing values.