Angular TS – Type Safety in TypeScript

[**Task 1 — Type Safety in TypeScript** 1](#_Toc205972007)

[**1. Purpose** 1](#_Toc205972008)

[**2. Theory** 1](#_Toc205972009)

[**3. Prerequisites** 2](#_Toc205972010)

[**4. Code Example (Step-by-Step)** 2](#_Toc205972011)

[**5. Project Structure Snapshot** 3](#_Toc205972012)

[**6. Running the Code** 3](#_Toc205972013)

[**7. Summary** 3](#_Toc205972014)

**Task 1 — Type Safety in TypeScript**

**Topic:** Types (Primitive, Union, Any, Unknown)  
**Focus:** Type Safety

**1. Purpose**

The purpose of this task is to understand how **TypeScript enforces type safety** through explicit type annotations.  
We will explore:

* Primitive types (string, number, boolean, null, undefined)
* Union types (multiple allowed types)
* any type (turns off type checking — unsafe)
* unknown type (safe alternative to any)

**2. Theory**

**Type safety** means your variables, parameters, and return values have a defined type, preventing accidental misuse at compile time.

**Primitive Types**

TypeScript supports basic types like:

let name: string = "Kiran";

let age: number = 30;

let isAdmin: boolean = true;

**Union Types**

Allow a variable to hold **one of several types**:

let score: number | string;

score = 100;

score = "Hundred";

**any**

Disables type safety for a variable:

let data: any = 5;

data = "hello"; // Allowed, but unsafe

**unknown**

Similar flexibility to any, but requires **type checks before use**:

let value: unknown = "Hello";

if (typeof value === "string") {

console.log(value.toUpperCase()); // Safe

}

**3. Prerequisites**

* Node.js installed
* TypeScript installed globally (npm install -g typescript ts-node)
* A tsconfig.json setup

**4. Code Example (Step-by-Step)**

**File:** task1-types.ts

// Primitive types

let username: string = "Kiran";

let userAge: number = 35;

let isLoggedIn: boolean = true;

// Union type

let productId: number | string;

productId = 101;

productId = "SKU-101";

// Any type (not safe)

let randomValue: any = 10;

randomValue = "Now I'm a string";

randomValue = true; // No errors

// Unknown type (safe)

let safeValue: unknown = "TypeScript";

if (typeof safeValue === "string") {

console.log("Uppercase:", safeValue.toUpperCase());

}

// Function with type safety

function addNumbers(a: number, b: number): number {

return a + b;

}

console.log("Username:", username);

console.log("User Age:", userAge);

console.log("Is Logged In:", isLoggedIn);

console.log("Product ID:", productId);

console.log("Random Value (any):", randomValue);

console.log("Addition Result:", addNumbers(5, 7));

**5. Project Structure Snapshot**

typescript-fundamentals/

├── task1-types.ts

├── tsconfig.json

└── package.json

**6. Running the Code**

ts-node task1-types.ts

**Expected Output:**

Username: Kiran

User Age: 35

Is Logged In: true

Product ID: SKU-101

Random Value (any): true

Uppercase: TYPESCRIPT

Addition Result: 12

**7. Summary**

* **Primitive types** → Best for simple, fixed type values.
* **Union types** → Allow flexibility without losing type safety.
* **any** → Avoid using unless absolutely necessary (turns off type checking).
* **unknown** → Safer alternative to any, forces type checks.
* **Type safety** → Catches errors early during compilation, reducing runtime bugs.