About types

Table of Contents

Types of Types

typesOfTypes.ts

```
let any_a: any = 666            // type: any
let any_b: any = ['danger']     // type: any
let any_c: any = any_a + any_b  // type: any
```

Table 1. Type of types

Type	Description	Bullet Points	Code

`any`	• **Godfather of types**
	• **Use it as Last resort only**
	• sdsd
		• dsd
		• fsf
			Correct any type
			`let any_a: any = 666 // type: any`
			`let any_b: any = ['danger'] // type: any`
			`let any_c: any = any_a + any_b // type: any`
			
			Compile error any type
			`let no_any_a = 666`
			`let no_any_b = ['danger']`
			`let no_any_c = no_any_a + no_any_b // Operator '+' cannot be applied to types 'number' and 'string[]'.ts(2365)`

Type	Description	Bullet Points	Code

`unknown`	• Type-safe counterpart of `any` type	You can only assign a `unknown` variable (let's say aVar) iff:
		1. If the variable `bVar: any` [i.e `aVar = bVar`]
		2. If `aVar` has type-check before asignment [i.e `if(typeOf(aVar) == boolean){aVar = cVar}` where `cVar: boolean`]
			Using unknown type
			`let myVar: unknown;`
			`let myVar1: unknown = myVar; // No error`
			`let myVar2: any = myVar; // No error`
			`let myVar3: boolean = myVar; // Type 'unknown' is not assignable to type 'boolean'`
			
			`if( typeof myVar1 == "boolean") {`
			`    let myVar4 = myVar`
			`}`

`boolean`	Gotchas: 1.		*Using boolean type*
			`let boolean_a = true // boolean`
			`var boolean_b = false // boolean`
			`const boolean_c = false // true`
			`let boolean_d : boolean = true // boolean`
			`let boolean_e : true = true // true: Type literal`
			`let boolean_f : true = false // Type 'false' is not assignable to type 'true'.ts(2322)`

Type	Description	Bullet Points	Code

`number`	Types:
	1. integers
	2. flats
	3. positives
	4. negatives
	5. Infinity
	6. NaN
		Another Type in typescript is `BigInt`. `int` supports 2^53; `BigInt` supports bigger
		
		`let bigint_a: bigint = 100n`
		``let bigint_b: bigint = 100 // Error TS2322: Type 100 not assignable to type `bigint` ``
		
		`bigint` is not supported in JS Engine rn. So better to not use it for now
			Using number type
			
			`let number_a = 1234`
			`let number_b = Infinity * 0.10`
			`const number_c  = 1234`
			`let number_d : number = 1234`
			`let number_e : 1234.12 = 1234.12`
			`let number_f : 1234.12 = 12 // Type '12' is not assignable to type '1234.12'.ts(2322)`

`string`			*Using string type*
			
			`let string_a = 'ayman'`
			`let string_b = 'billy'`
			`const string_c  = 'chinmay'`
			`let string_d : string = 'zoom'`
			`let string_e = string_a + ' ' + string_b + string_c`
			`let string_f : 'john' = 'zoe' // Error TS2322: Type 'zoe' is not assignable to type 'john'.ts`

Type	Description	Bullet Points	Code
`symbol`	`symbol` was added in ES2015 TODO: Learn more		*Using symbol type* `let symbol_a = Symbol('a') // symbol` `let symbol_b: symbol = Symbol('b') // symbol` `var symbol_c = symbol_a === symbol_b // boolean` `let symbol_d = symbol_a + 'x' // Error TS2469: The '+' operator cannot be applied to type 'symbol'.` `const symbol_e = Symbol('e') // typeof e` `const symbol_f: unique symbol = Symbol('f') // typeof f` `let symbol_g: unique symbol = Symbol('f')// Error TS1332: A variable whose type is a 'unique symbol' type must be 'const'.` `let symbol_h = symbol_e === symbol_e // boolean` `let symbol_i = symbol_e === symbol_f // Error TS2367: This condition will always return 'false' since the types 'unique symbol' and 'unique symbol' have no overlap.`

Type Literal

> A type that represents a single vawlue and nothing else

```
let a: true = true // Telling TS that a is not only just a boolean, but it is also of
the value true
```

Learn type widening in chapter06-Advanced types