Tipos primitivos en TypeScript

5 minutos

Comencemos por revisar los tipos más básicos y comunes que se pueden encontrar al escribir código de JavaScript o TypeScript. Estos tipos básicos formarán más adelante los bloques de compilación principales de tipos más complejos.

Tipo booleano

El tipo de datos más básico es el valor `true` o `false`, conocido como booleano.

Por ejemplo:

TypeScript

```
let flag: boolean;
let yes = true;
let no = false;
```

Tipos numéricos y BigInteger

Al igual que en JavaScript, todos los números en TypeScript son valores de número de punto flotante o BigIntegers. Estos números de punto flotante obtienen el tipo `number`, mientras que los valores BigIntegers obtiene el tipo `bigint`. Además de los literales hexadecimales y decimales, TypeScript también admite los literales binarios y octales introducidos en ECMAScript 2015.

Por ejemplo:

TypeScript

```
let x: number;
let y = 0;
let z: number = 123.456;
let big: bigint = 100n;
```

Tipo de cadena

La palabra clave `string` representa secuencias de caracteres almacenados como unidades de código Unicode UTF-16. Al igual que JavaScript, TypeScript también usa comillas dobles (`"`) o comillas simples (`'`) para rodear los datos de cadena.

He aquí algunos ejemplos:

TypeScript

```
let s: string;
let empty = "";
let abc = 'abc';
```

En TypeScript, también puede usar cadenas de plantilla, que pueden abarcar varias líneas y tener expresiones insertadas. Estas cadenas están rodeadas por el carácter de comilla simple/tilde grave (') y las expresiones insertadas tienen el formato `${ expr }`.

Por ejemplo:

TypeScript

```
let firstName: string = "Mateo";
let sentence: string = `My name is ${firstName}.
    I am new to TypeScript.`;
console.log(sentence);
```

Este ejemplo produce el resultado siguiente:

Consola

```
My name is Mateo.
    I am new to TypeScript.
```

Los tipos void, null y undefined

JavaScript y TypeScript tienen dos valores primitivos que se usan para indicar un valor ausente o con inicialización anulada: `null` y `undefined`. Estos tipos son más útiles en el contexto de las funciones, por lo que se tratarán con más detalle en un módulo posterior.

Siguiente unidad: Ejercicio: Enumeraciones

Continuar >

¿Qué tal lo estamos haciendo? ☆ ☆ ☆ ☆ ☆