Avanzados y Funcione...

Curso de Fundamentos de TypeScript

→|BOOLEANS

Numbers 9/24

RECURSOS MARCADORES

El tipo de dato `number` se usa para variables que contendrán números positivos, negativos o decimales.

Operaciones

En JavaScript, una variable de tipo `number` puede fácilmente ser concatenado con otra de tipo `string` :

```
//JavaScript
let myNumber = 30;
myNumber = myNumber + "5"; //El resultado sería '305'
```

Sin embargo, esto podría llevar confusiones y errores durante la ejecución del programa, además de estar cambiando el tipo de dato de la variable. Por ello, en TypeScript solo se pueden hacer operaciones numéricas entre números valga la redundancia:

```
//TypeScript
let myNumber: number = 30;

myNumber = myNumber + 10; //CORRECTO
myNumber = myNumber + "10"; //INCORRECTO
```

Uso de variables sin inicializar

- Serán señalados como errores aquellas variables que queramos usar sin haberles dado un valor inicial:

```
//TypeScript
let productInStock: number;
console.log("Product in stock: " + productInStock);
```

Señalar que si no se va a inicializar aún la variable, definir explícitamente el tipo de dato, pues TypeScript no puede inferirlo si no tiene un valor inicial.

Conversión de números de tipo string a tipo number

Para esto usaremos el método `parseInt` :

```
let discount: number = parseInt("123");

let numeroString: string = "100";
let nuevoNumero: number;
nuevoNumero = parseInt(numeroString);
```

Esto funciona si el string tiene solo y exclusivamente números que no empiecen con 0. De lo contrario, el resultado será de tipo `NaN` (Not a Number):

```
//TypeScript
let numeroPrueba: number = parseInt("palabra");
console.log(numeroPrueba); //NaN
```

Binarios y Hexadecimales

TypeScript nos puede indicar error si intentamos definir números binarios que tengan números que no sean 0 o 1 y si declaramos hexadecimales usando valores fuera del rango:

```
//**********TypeScript**********
//Binarios: se definen colocando "0b" al inicio del valor
let primerBinario = 0b1010; //CORRECTO
let segundobinario = 0b1210; //INCORRECTO. El 2 es inválido

//Hexadecimales: se definen colocando "0x" al inicio del valor
let primerHexa = 0xfff; //CORRECTO
let segundoHexa = 0xffz; //INCORRECTO. El "z" es inválido
```

En consola, si están correctamente asignados, se hará una conversión a decimal de dichos números:

```
let primerHexa = 0xfff;
console.log(primerHexa); // 4095

let primerBinario = 0b1010;
console.log(primerBinario); // 10
```

Consejo

Cuando definas una variable de tipo de dato `number`, es preferible que el nombre de tipo sea en minúscula. Esto como buena práctica, pues se hará referencia al tipo de dato `number` y **no al objeto** `Number` propio del lenguaje:

```
let myNumber: number = 20; // Buena practica.
let otherNumber: Number = 20; // Mala practica.
```

Contribución creada por: Martín Álvarez.

Archivos de la clase

- /
- package.json
- package-lock.json
- .gitignore
- tsconfig.json
- .editorconfig
- src

Lecturas recomendadas