TS TypeScript TUTORIAL

TypeScript Number

Summary: in this tutorial, you'll learn about the TypeScript number data types.

All numbers in TypeScript are either floating-point values or big integers. The floating-point numbers have the type `number` while the big integers get the type `bigint`.

The number type

The following shows how to declare a variable that holds a floating-point value:

```
let price: number;
```

Or you can initialize the `price` variable to a number:

```
let price = 9.95;
```

As in JavaScript, TypeScript supports the number literals for decimal, hexadecimal, binary, and octal literals:

Decimal numbers

The following shows some decimal numbers:

```
let counter: number = 0;
let x: number = 100,
    y: number = 200;
```

Binary Numbers

The binary number uses a leading zero followed by a lowercase or uppercase letter “B” e.g., `0b` or `0B` :

```
let bin = 0b100;
let anotherBin: number = 0B010;
```

Note that the digit after `0b` or `0B` must be `0` or `1` .

Octal Numbers

An octal number uses a leading zero followed by the letter `o` (since ES2015) `0o` . The digits after `0o` are numbers in the range `0` through `7` :

```
let octal: number = 0o10;
```

Hexadecimal numbers

Hexadecimal numbers use a leading zero followed by a lowercase or uppercase letter `X` (`0x` or `0X`). The digits after the `0x` must be in the range (`0123456789ABCDEF`). For example:

```
let hexadecimal: number = 0XA;
```

JavaScript has the `Number` type (with the letter `N` in uppercase) that refers to the non-primitive boxed object. You should not use this `Number` type as much as possible in TypeScript.

Big Integers

The big integers represent the whole numbers larger than 2^{53} – 1. A Big integer literal has the `n` character at the end of an integer literal like this:

```
let big: bigint = 9007199254740991n;
```

Summary

- All numbers in TypeScript are either floating-point values that get the number type or big integers that get the `bigint` type.
- Avoid using the `Number` type as much as possible.