05

Para saber mais: string v String e number vs Number

Durante nossas aulas, é provável que vocês tenham reparado que o Visual Studio Code, durante o processo de autocomplete, nos permite escolhermos além dos tipos `string` e `number`, os tipos `String` e `Number`. Há diferença?

Os tipos que começam em minúsculo equivalem a declaração literal. Vejamos um exemplo:

```
let nome = 'Flávio';
let idade = 20;
```

O TypeScript infere o tipo, sendo assim, a sintaxe é a mesma coisa que:

```
let nome: string = 'Flávio';
let idade: number = 20;
```

Se fizermos `typeof` nas duas variáveis temos como resultado `string` e `number` respectivamente:

```
let nome: string = 'Flávio';
console.log(typeof(nome));  // string
let idade: number = 20;
console.log(typeof(idade));// number
```

Contudo, JavaScript permite criar strings e números não como literais, mas como objetos:

```
let nome = new String('Flávio');
console.log(typeof(nome)); // Object
let idade = new Number(20);
console.log(typeof(idade)); // Object
```

Qual a diferença?

Os tipos `string` e `number` são literais e guardam um valor primitivo. Contudo, se tentarmos chamar algum método em variáveis declaradas com esses tipos, eles são empacotados automaticamente (auto-boxing) para `String` e `Number` respectivamente.

É por isso que esse código funciona:

```
let nome = 'Flávio';
nome.replace('/vio/', 'vião'); // faz auto-boxing
```

Dessa forma, TypeScript permite distinguir entre o tipo literal e o tipo objeto. Contudo, a boa prática é usarmos os tipos literais `number` e `string`, porque em JavaScript `new String()` e `new Number()` são raramente usados.