

Tipagem no Typescript

O Typescript possui esse nome devido à tipagem que foi adicionada à linguagem, além de outras estruturas que linguagens fortemente tipadas possuem. Por isso, vamos ver como funciona a tipagem na linguagem.


A tipagem no Typescript funciona de duas formas: a tipagem **implícita** ou a tipagem **explícita**. Na tipagem implícita, você não informa diretamente o tipo da variável, mas o Typescript interpreta e dá um tipo à variável automaticamente. Caso você tente colocar outro valor diferente do tipo da variável, o Typescript informará erro



A variável **msg**, do exemplo acima, é do tipo **string** e, a última linha, tentando atribuir um valor numérico à variável. Isso, no Typescript, dará erro de tipo. No Javascript, isso seria possível, mas o Typescript informa erro nessas atribuições.


Obs: O Typescript, mesmo informando erro na questão de atribuição de valores às variáveis, mesmo assim, ele ainda transforma o arquivo Typescript em Javascript, pois o JS aceita essas atribuições. No entanto, caso deseje impedir esse comportamento do Typescript, você precisa passar a flag `--noEmitOnError` na hora de transformar o TS em JS. Assim, caso o compilador veja algum erro desse tipo, ele impedirá que o arquivo JS seja gerado.

Na tipagem explícita, você informa diretamente no código o tipo da variável, ficando claro tanto para o compilador quanto para o programador o tipo das variáveis.



```
let idade: number = 25  
console.log(`Eu tenho ${idade} anos!`)
```

Aqui declaramos uma variável que aceita variáveis numéricas. Diferentemente de outras linguagens tipadas que têm tipos diferentes para números inteiros e números decimais, o Typescript e o Javascript possuem um tipo único para números: o **number**. Então, a variável *idade* declarada acima aceita tanto um valor inteiro quanto decimal.



```
let myStr: string = 'Valor em String'  
let myNum: number = 10  
let myBool: boolean = true
```

Essas são as tipagens básicas de qualquer linguagem. Agora vamos ver alguns outros tipos do Typescript.

Arrays

As variáveis também podem receber um tipo Array. Você pode criar uma variável que recebe um array de strings, como no exemplo abaixo



```
let strArray: Array<string> = ['Lorem', 'Ipsum', 'Dolor', 'Sit', 'A Met']  
let strArray2: string[] = ['Lorem', 'Ipsum', 'Dolor', 'Sit', 'A Met']
```

As duas maneiras tipam uma variável como um array de strings. Você pode criar um array de quaisquer outros tipos, como arrays de números, booleanos, de objetos ou de qualquer outro tipo.

Objetos

Você já viu Objetos como esse no Javascript:



```
let funcionario = {  
  nome: 'Paulo',  
  email: 'paulo@mail.com',  
  idade: 35,  
}
```

Quando você cria um objeto como esse no Typescript, ele infere à variável que ela deve receber um objeto com os campos **nome (string)**, **email (string)** e **idade (number)**. Caso você informe à variável um outro objeto com alguma propriedade diferente dessas ou a propriedade não receba um valor, o Typescript informará que o código está errado

Você também pode fazer uma tipagem explícita da variável. Utilizando as mesmas propriedades e tipos das propriedades do objeto acima, o código ficaria

```
let funcionario: { nome: string, email: string, idade: number } = {  
  nome: 'Paulo',  
  email: 'paulo@mail.com',  
  idade: 35,  
}
```

Nesse caso, você está explicitando as propriedades e seus tipos que são esperados nesse objeto. No entanto, dependendo da quantidade de propriedades que seu objeto tenha, fazer esse tipo de estrutura para explicitar a tipagem de uma variável não é a melhor apropriada. Por isso, para facilitar a tipagem explícita desses tipos, podemos criar um **Alias**.

Alias é um tipo especial criado pelo desenvolvedor, informando quais campos uma variável do nosso tipo especial deve ter. Veja no exemplo abaixo:

```
type Funcionario = {  
  nome: string,  
  email: string,  
  idade: number  
}  
  
let funcionario: Funcionario = {  
  nome: 'Paulo',  
  email: 'paulo@mail.com',  
  idade: 35,  
}
```

A palavra chave `type` acompanhada do nome do tipo especial que você criou e do nome e tipo de suas propriedades é Alias. Com o Alias criado, basta informar que a variável é do seu tipo especial, assim o Typescript já sabe quais campos e quais tipos de dados a variável deve ter.


Obs: Quando uma variável é tipada com um objeto, todos os campos, por padrão, são obrigatórios dentro do objeto. No entanto, você pode deixar alguns campos como opcionais. Para isso, basta que, após informar o nome da propriedade, você adicione um ponto de interrogação (?) após o nome. Isso fará com que o Typescript entenda que uma determinada propriedade não precise ser passada.



```
type Funcionario = {  
  nome: string,  
  email: string,  
  idade: number,  
  telefone?: string  
}  
  
let funcionario: Funcionario = {  
  nome: 'Paulo',  
  email: 'paulo@mail.com',  
  idade: 35,  
  telefone: '11223344'  
} // funciona  
  
let funcionario2: Funcionario = {  
  nome: 'Paulo',  
  email: 'paulo@mail.com',  
  idade: 35  
} // funciona
```


Union Types

Imagine o cenário: no contexto de uma aplicação que você está desenvolvendo, ela aceita que números telefônicos possam ser salvos tanto como strings quanto como valores numéricos. Utilizando o Typescript, você tem duas maneiras de resolver isso: você pode criar duas variáveis, uma do tipo string e outra do tipo number, como no exemplo abaixo:



```
let telefoneStr: string = '11223344'  
let telefoneNum: number = 11223344
```

Ou você pode criar uma variável que aceita ao mesmo tempo tanto valores em string quanto valores numéricos utilizando o **union type**. O Union Type permite que uma variável assumam mais de um tipo ao mesmo tempo. O exemplo acima, usando os Union Types, ficaria




```
let telefone: string | number = '11223344'  
telefone = 11223344
```

Utilizando o pipe (|), o Typescript permite que variáveis assumam mais de um tipo de valor.

Tipo “any”

O Javascript é conhecido por sua tipagem dinâmica, ou seja, qualquer variável pode assumir qualquer valor. O Typescript, para continuar com essa característica tão importante da linguagem, implementou o tipo **any**. Esse tipo permite que uma variável receba qualquer tipo de valor. Desde um booleano a um objeto. Isso permite a flexibilidade que o Javascript sempre trouxe nativamente dentro do Typescript



```
let minhaVar: any = 'Hello World'

console.log(minhaVar) // -> Hello World

minhaVar = 5

console.log(minhaVar) // -> 5

minhaVar = true

console.log(minhaVar) // -> true

minhaVar = ['5', 5, true, false, 0, 1]

console.log(minhaVar) // -> ['5', 5, true, false, 0, 1]
```

Contudo, mesmo o Typescript aceitando esse tipo de nomenclatura, evite usá-la em qualquer contexto da sua aplicação. Você pode usá-la, por exemplo, para testar e descobrir qual o tipo de dado retornado em uma função. Mas evite usar em qualquer momento da sua aplicação com TS. Isso pode levar a problemas futuros no seu desenvolvimento.