17 DE SETEMBRO DE 2024 / #TYPESCRIPT

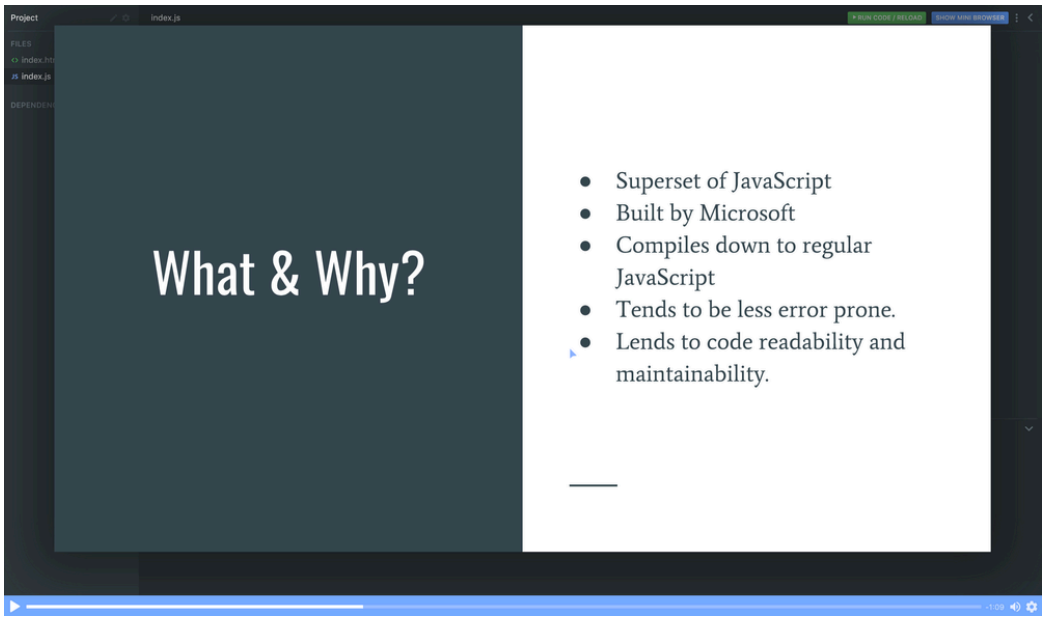
Aprenda TypeScript em cinco minutos – um tutorial para iniciantes

Tradutor: Daniel Rosa

Autor: freeCodeCamp.org (em inglês)

Artigo original: Learn TypeScript in 5 minutes - A tutorial for beginners

Escrito por: Per Harald Borgen



Clique aqui para conferir o curso gratuito de TypeScript no Scrimba

TypeScript é um superconjunto tipado de JavaScript, que visa tornar a linguagem mais escalável e confiável.

Ele é de código aberto e tem sido mantido pela Microsoft desde que eles o criaram, em 2012. No entanto, o TypeScript teve seu avanço inicial como a linguagem de programação principal no Angular 2. Ele vem crescendo continuamente desde então, também nas comunidades do React e do Vue.

Neste tutorial, você aprenderá o básico do TypeScript com a ajuda de exemplos práticos.

Também estamos lançando um curso gratuito de TypeScript em 22 partes no Scrimba. Deixe seu e-mail aqui se quiser acesso antecipado!

Vamos começar.

Instalando o TypeScript

Antes de começarmos a programar, precisamos instalar o TypeScript no nosso computador. Usaremos o `npm` para isso, então abra o terminal e digite o seguinte comando:

```
npm install -g typescript
```

Uma vez instalado, podemos verificar executando o comando `tsc -v` que exibirá a versão do TypeScript instalada.

Escrevendo algum código

Vamos criar nosso primeiro arquivo TypeScript e escrever algum código nele. Abra seu IDE ou editor de texto favorito e crie um arquivo com o nome de `first.ts` — Para arquivos TypeScript, usamos a extensão `.ts`.

Por enquanto, vamos apenas escrever algumas linhas de JavaScript puro, pois todo código JavaScript também é válido no TypeScript:

```
let a = 5;
let b = 5;
let c = a + b;

console.log(c);
```

O próximo passo é compilar nosso TypeScript em JavaScript puro, já que os navegadores querem arquivos `.js` para ler.

Compilando o TypeScript

Para compilar, executaremos o comando `tsc filename.ts`, que cria um arquivo JavaScript com o mesmo nome, mas com uma extensão diferente, e que eventualmente podemos passar para nossos navegadores.

Então, abra o terminal na localização do arquivo e execute o seguinte comando:

```
tsc first.ts
```

Dica: se você quiser compilar todos os arquivos TypeScript dentro de qualquer pasta, use o comando: `tsc *.ts`

Tipos de dados

TypeScript—como o nome sugere—é a versão tipada do JavaScript. Isso significa que podemos especificar tipos para diferentes variáveis no momento da declaração. Elas sempre manterão o mesmo tipo de dados dentro daquele escopo.

A tipagem é um recurso muito útil para garantir confiabilidade e escalabilidade. A verificação de tipos ajuda a garantir que nosso código funcione conforme o esperado. Além disso, ajuda a encontrar *bugs* e erros e documentar adequadamente nosso código.

A sintaxe para atribuir um tipo a qualquer variável é escrever o nome da variável seguido por um sinal `:`, o nome do tipo seguido por um sinal `=` e o valor da variável.

Existem três tipos diferentes em TypeScript: o tipo `any`, os tipos `Built-in` (incorporados), e os tipos `User-defined` (definidos pelo usuário). Vamos dar uma olhada em cada um deles.

Tipo any

O tipo de dado `any` é o superconjunto de todos os tipos de dados em TypeScript. Dar a qualquer variável o tipo `any` é equivalente a optar por não verificar o tipo dessa variável.

```
let myVariable: any = 'This is a string'
```

Tipos incorporados

Estes são os tipos que são nativos do TypeScript. Eles incluem `number`, `string`, `boolean`, `void`, `null` e `undefined`.

```
let num: number = 5;
let name: string = 'Alex';
let isPresent: boolean = true;
```

Tipos definidos pelo usuário

Os tipos definidos pelo usuário incluem `enum`, `class`, `interface`, `array` e `tuple`. Discutiremos alguns desses mais adiante neste artigo.

Programação orientada a objetos

O TypeScript suporta todos os recursos da programação orientada a objetos, como classes e interfaces. Essa capacidade é um grande impulso para o JavaScript — que sempre teve dificuldades com sua funcionalidade de OOP, especialmente desde que os desenvolvedores começaram a usá-lo para aplicações de grande escala.

Classe

Na programação orientada a objetos, uma classe é o modelo de objetos. Uma classe define como um objeto seria em termos de características e funcionalidades desse objeto. Uma classe também encapsula dados para o objeto.

TypeScript tem suporte embutido para classes, que não eram suportadas pelo ES5 e versões anteriores. Isso significa que podemos usar a palavra-chave `class` para declarar uma classe facilmente.

```
class Car {

// campos
  model: String;
  doors: Number;
  isElectric: Boolean;

constructor(model: String, doors: Number, isElectric: Boolean) {
    this.model = model;
    this.doors = doors;
    this.isElectric = isElectric;
  }

displayMake(): void {
    console.log(`This car is ${this.model}`);
  }

}
```

No exemplo acima, declaramos uma classe `Car`, junto com algumas de suas propriedades, que estamos inicializando no `construtor`. Também temos um método que exibiria alguma mensagem usando sua propriedade.

Vamos ver como podemos criar uma instância dessa classe:

Para criar um objeto de uma classe, usamos a palavra-chave `new`, chamamos o construtor da classe e passamos suas propriedades. Agora, esse objeto `Prius` possui suas próprias propriedades de `model`, `doors` e `isElectric`. O objeto também pode chamar o método `displayMake`, que terá acesso às propriedades do `Prius`.

Interface

O conceito de interfaces é outro recurso poderoso do TypeScript, que permite definir a estrutura das variáveis. Uma interface é como um contrato sintático ao qual um objeto deve obedecer.

Interfaces são melhor descritas através de um exemplo real. Suponha que tenhamos um objeto `Car`:

```
const Car = {
  model: 'Prius',
  make: 'Toyota',
  display() => { console.log('oi'); }
}
```

Se olharmos para o objeto acima e se tentarmos extrair sua assinatura, ela seria:

```
{
  model: String,
  make: String,
  display(): void
}
```

Se quisermos reutilizar essa assinatura, podemos declará-la na forma de uma interface. Para criar uma interface, usamos a palavra-chave `interface`.

```
interface ICar {
  model: String,
  make: String,
  display(): void
}

const Car: ICar = {
  model: 'Prius',
  make: 'Toyota',
  display() => { console.log('oi'); }
}
```

Aqui, declaramos uma interface chamada `ICar` e criamos um objeto `Car`. `Car` está agora vinculado à interface `ICar`, garantindo que o objeto `Car` defina todas as propriedades que estão na interface.

Conclusão

Espero que isso tenha dado a você uma visão rápida sobre como o TypeScript pode tornar seu JavaScript mais estável e menos propenso a erros.

O TypeScript está ganhando muito impulso no mundo do desenvolvimento para a web. Há também um número crescente de desenvolvedores do React que o estão adotando. O TypeScript é definitivamente algo que qualquer desenvolvedor de *front-end* deve conhecer.

Boa programação para você! 🙂

Agradecemos pela leitura! O autor deste artigo é Per Borgen, co-fundador do Scrimba – a maneira mais fácil de aprender a programar. Confira o bootcamp de design responsivo para a web do Scrimba se quiser aprender a construir sites modernos em um nível profissional.

Clique aqui para acessar o bootcamp avançado
