

Typescript

Typescript, o que é?

O que é Typescript?

- <https://www.typescriptlang.org/>
- TypeScript é uma linguagem de programação de código aberto desenvolvida pela Microsoft.
- Ela é uma extensão da linguagem JavaScript que adiciona recursos de tipagem estática ao JavaScript
 - Chamamos o TypeScript de **superset** do Javascript

O que é Typescript?

- O TypeScript detecta quando passamos um valor diferente para uma variável declarada durante o desenvolvimento e avisa na IDE (no seu editor de código).
- Isso reflete num ambiente muito mais seguro enquanto o código está sendo digitado.

Vantagens

- Verificação de tipos em tempo de compilação
- Melhor suporte a IDE
- Facilita a colaboração em equipes grandes
- Código mais robusto

Desvantagens

- Curva de aprendizado é maior
- Compilação adicional
- Tamanho do arquivo
- Problemas de compatibilidade
 - Algumas bibliotecas precisam de esforços adicionais para funcionar

Iniciando

Iniciando

- É preciso ter o Node instalado
- Instale o Typescript de maneira global: *npm install -g typescript*
- Crie um projeto simples que tenha um arquivo **index.html** e um **index.ts**
- Abra o projeto no VSCode
- Crie uma pasta chamada **dist**
- No terminal execute o comando para criar o arquivo de configuração do TypeScript: `tsc --init`

Iniciando

- Abra o arquivo **tsconfig.json**
- Procure a propriedade **target**, descomente e troque para **'es6'**
- Procure a propriedade **outDir**, descomente e troque o valor para **'./dist'**

Iniciando

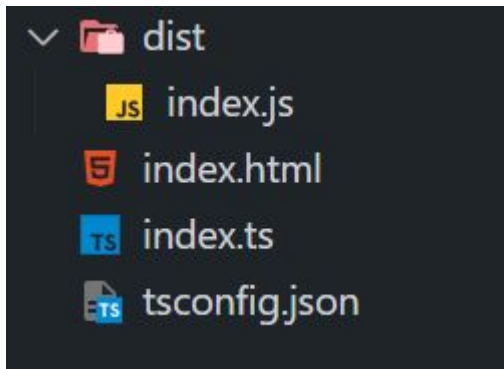
- No arquivo **index.ts** crie uma função simples de *soma* e exiba o retorno no console



```
1  const somar = (a: number, b: number) => {  
2      return a + b;  
3  };  
4  
5  console.log(somar(5, 8));
```

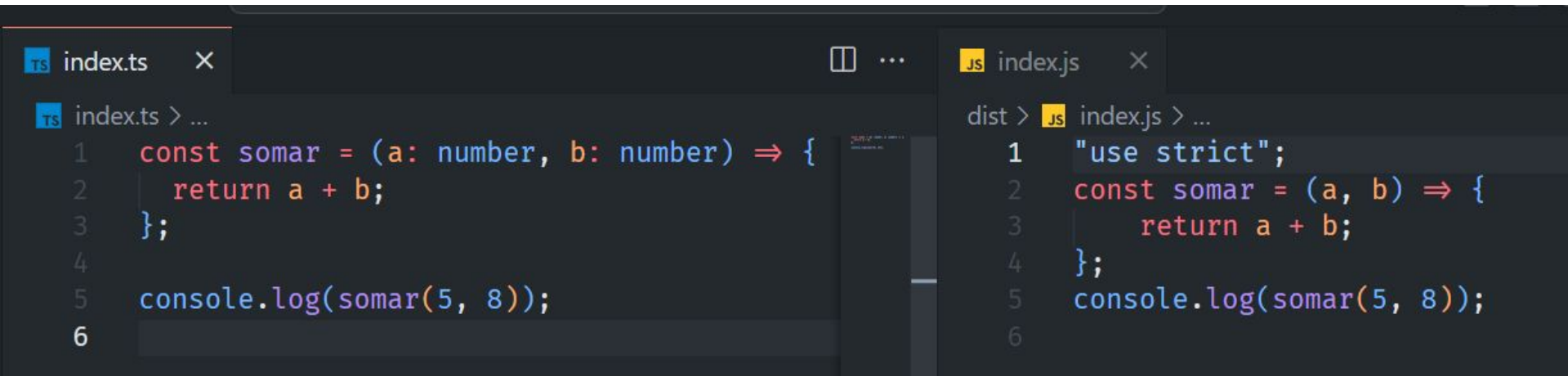
Iniciando

- No terminal rode o comando `tsc`
- Verá que o Typescript criou uma pasta **dist** e dentro dela o arquivo **index.js**



Iniciando

- Comparando verá que o conteúdo é bem semelhante



The image shows a side-by-side comparison of two code files in a dark-themed editor. The left pane shows a TypeScript file named 'index.ts' with a function 'somar' that takes two numbers and returns their sum, followed by a log statement. The right pane shows a JavaScript file named 'index.js' with the same function logic, but it includes a 'use strict' directive at the top. The code is color-coded: keywords in blue, variables in purple, and literals in orange.

```
TS index.ts > ...
1  const somar = (a: number, b: number) => {
2    return a + b;
3  };
4
5  console.log(somar(5, 8));
6

JS index.js > ...
1  "use strict";
2  const somar = (a, b) => {
3    return a + b;
4  };
5  console.log(somar(5, 8));
6
```

Iniciando

- Isso acontece pois o Typescript é usado apenas durante o desenvolvimento
- O arquivo a ser referenciado no html será o **dist/index.js**

Tipos

String



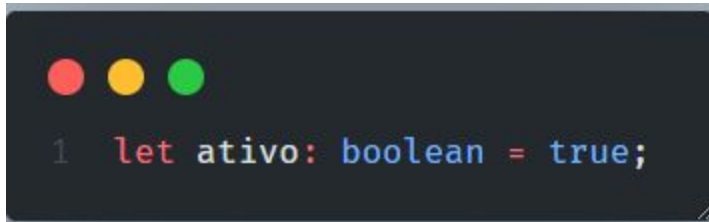
```
1 let nome: string = "Typescript";
```

Number



```
1 let idade: number = 10;
```


Boolean



```
1  let ativo: boolean = true;
```

Array



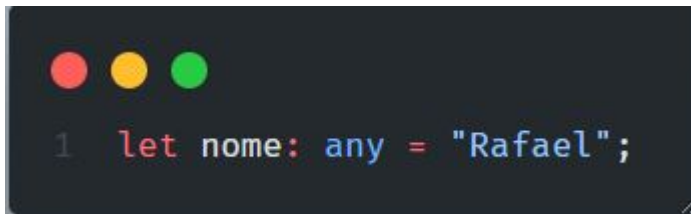
```
1 let nome: string[] = ["Rafael", "Gabriel", "Miguel"];  
2 let idades: number[] = [100, 200, 300];
```



```
1 let campos: (string | number)[] = [37, "fulano"];
```

Any

- Usado quando aceita qualquer tipo de dado ou desconhecemos o seu valor: deve ser usada com *muito cuidado!*



```
1 let nome: any = "Rafael";
```

Void

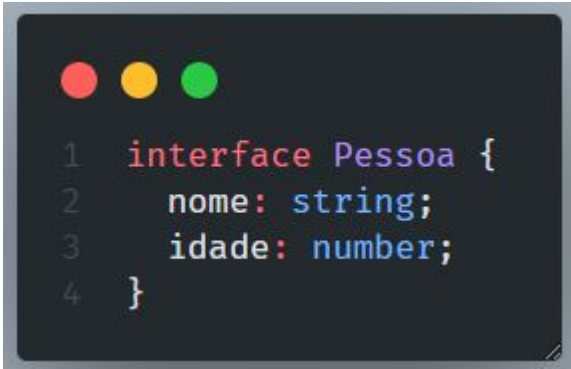
- Usado para funções sem retorno:




```
1  const exibirConsole = (mensagem: string): void => {  
2      console.log(mensagem);  
3  };
```

Interfaces

- São conjuntos de métodos e propriedades que geralmente descrevem um objeto.
- A interface pode ser declarada antes da função principal ou em um arquivo separado.



```
1 interface Pessoa {  
2     nome: string;  
3     idade: number;  
4 }
```



```
1 let dev: Pessoa = {  
2     nome: "Rafael",  
3     idade: 18,  
4 };
```

Interfaces



```
1 // Dados primitivos
2 const [enabled, setEnabled] = useState<boolean>(false);
3
4
5 // Interface
6 interface Status = "idle" | "loading" | "success" | "error";
7
8 const [status, setStatus] = useState<Status>("idle");
```

Propriedades Opcionais

- Nem sempre temos certeza que um dado será exibido ou lido em nossa aplicação, para isso temos a propriedade opcional do TypeScript, que são marcadas com um "?".
- Nesses casos, os objetos da interface podem ou não definir essas propriedades.

Propriedades Opcionais



```
1 interface Pessoa {  
2   nome: string;  
3   idade: number;  
4   funcao?: string;  
5 }
```



```
1 let dev: Pessoa = {  
2   nome: "Rafael",  
3   idade: 18,  
4   };
```



```
1 let dev: Pessoa = {  
2   nome: "Rafael",  
3   idade: 18,  
4   funcao: "Programador Frontend Pleno",  
5   };
```


Estendendo Interfaces

- As interfaces podem estender uma ou mais interfaces, isso é, uma interface pode ter as propriedades declaradas em uma outra interface.
- Isso torna as interfaces e suas propriedades mais flexíveis e reutilizáveis.

Extendendo Interfaces



```
1 interface Pessoa {  
2     nome: string;  
3     idade: number;  
4 }  
5  
6 interface Desenvolvedor extends Pessoa {  
7     funcao: string;  
8     senioridade: string;  
9     tempo: number;  
10 }
```



```
1 let dev: Desenvolvedor = {  
2     nome: "Rafael",  
3     idade: 18,  
4     funcao: "Programador",  
5     senioridade: "Pleno",  
6     tempo: 5,  
7 };
```

Eventos



```
1  const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {  
2    console.log(e.target.value);  
3  };  
4  
5  const handleSubmit = (e: React.SyntheticEvent) => {  
6    e.preventDefault();  
7  };
```

Props



```
1  interface MyButtonProps {  
2    title: string;  
3    disabled: boolean;  
4  }  
5  
6  function MyButton({ title, disabled }: MyButtonProps) {  
7    return (  
8      <button disabled={disabled}>{title}</button>  
9    );  
10 }  
11  
12 export default function MyApp() {  
13   return (  
14     <div>  
15       <h1>Welcome to my app</h1>  
16       <MyButton title="I'm a disabled button" disabled={true}/>  
17     </div>  
18   );  
19 }
```

Dúvidas? 🤔