



Introdução



Ambiente



Tipos Simples



Tipos Especiais



Tipos Compostos



Outros Tipos



TypeScript - Parte 1

Prof. Enzo
Seraphim

Profa. Bárbara
Pimenta Caetano



TypeScript



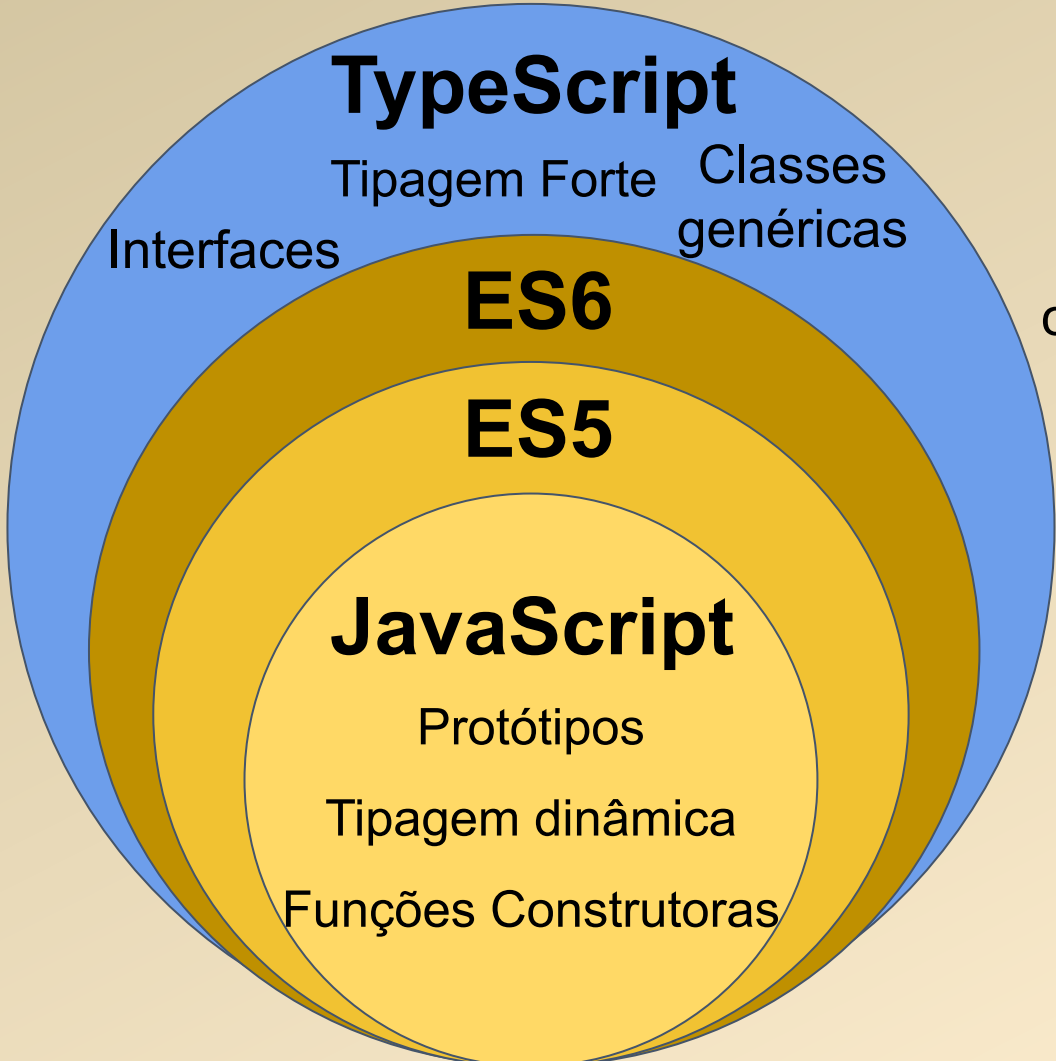
Introdução

- Criado em 1/outubro/2012 versão 0.8
- Inventada por Anders Hejlsberg/Microsoft → arquiteto da linguagem C# e criador das linguagens Delphi e Turbo Pascal
- Superconjunto sintático estrito de JavaScript e adiciona tipagem estática opcional
- Tipos fornecem uma maneira de descrever a forma de um objeto e permitindo validar se seu código está funcionando corretamente

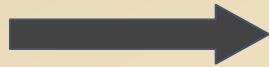




Introdução



compila para



O Browser
não compila
TypeScript



TypeScript

Introdução

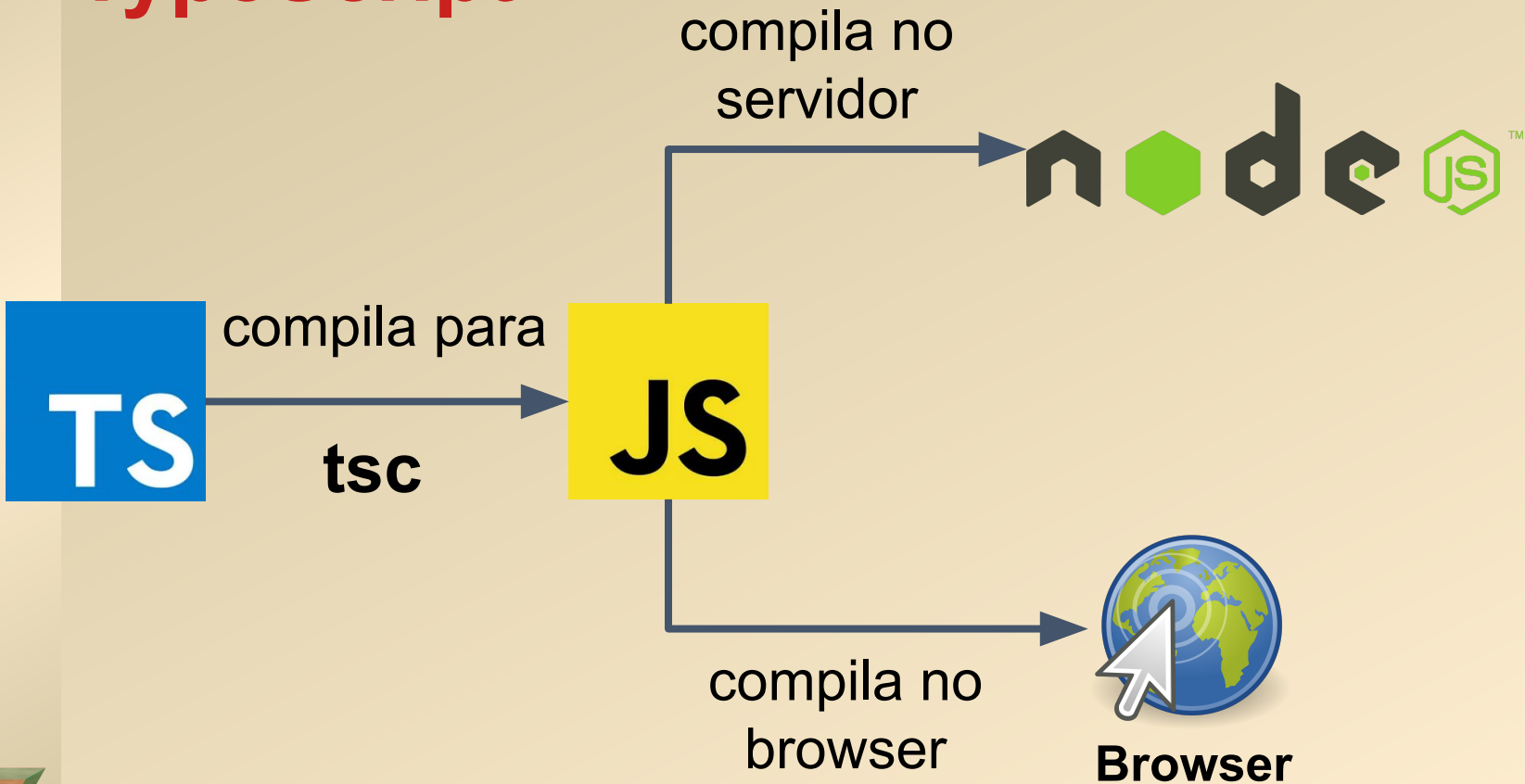
- TypeScript é uma linguagem de programação compilada que traduz o fonte para JavaScript
- É seguro para tipos (Type Safe) por garantir que erros relacionados a tipos de dados incompatíveis ou incorretos sejam detectados em tempo de compilação e não durante a execução
- O fonte JavaScript pode ser interpretado (traduzido para código de máquina) em navegadores, ou node.js ou WinJS





TypeScript

Introdução





Versões – TypeScript

Introdução

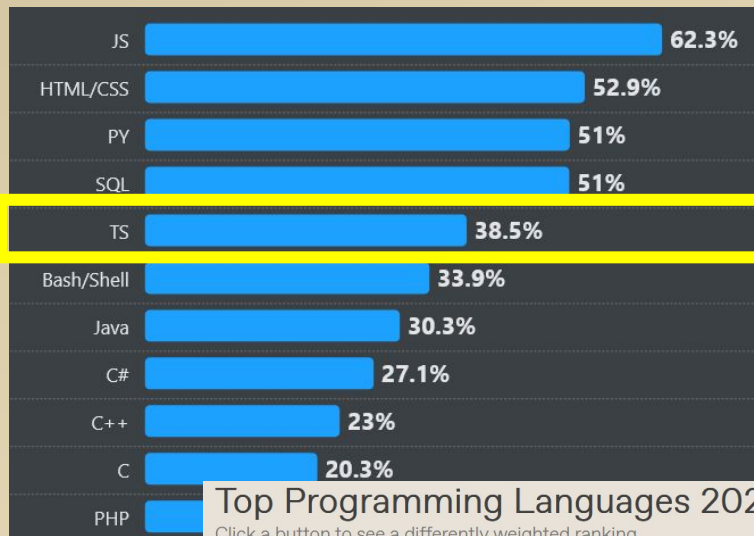
- TypeScript 1.0 em 2/4/2014
- TypeScript 2.0 em 2/9/2016 → recurso opcional impede que variáveis recebam valores nulos (erro de bilhões de dólares)
- TypeScript 3.0 em 30/7/2018 → operador de espalhamento, parâmetros rest com tipos de tupla, parâmetros rest genéricos e assim por diante.
- TypeScript 4.0 em 2/4/2021
- TypeScript 5.0 em 15/3/2023 adicionado suporte a decoradores





Introdução

Stack Overflow/2024



5^a

Top Programming Languages 2024

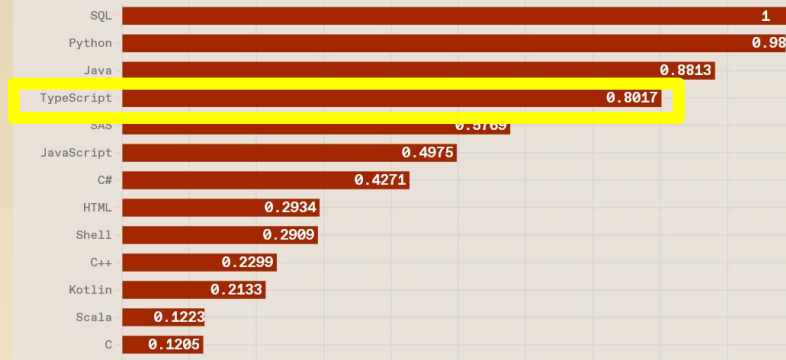
Click a button to see a differently weighted ranking

IEEE Spectrum

Spectrum Trending Jobs

2024

4^a



RedMonk/2024

1 JavaScript

2 Python

3 Java

4 PHP

5 C#

6^a TypeScript

7 CSS

7 C++

9 Ruby

10 C

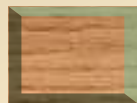
11 Swift



Introdução



Ambiente



Tipos Simples



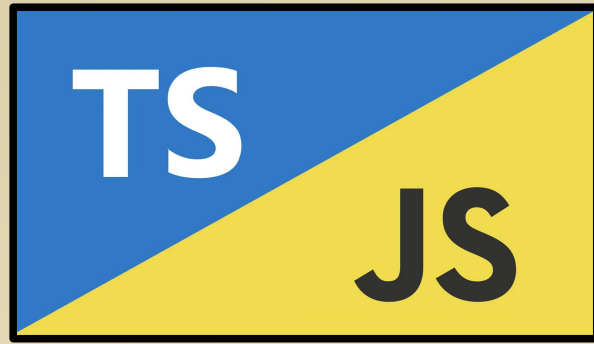
Tipos Especiais



Tipos Compostos



Outros Tipos



TypeScript - Parte 1

Prof. Enzo
Seraphim

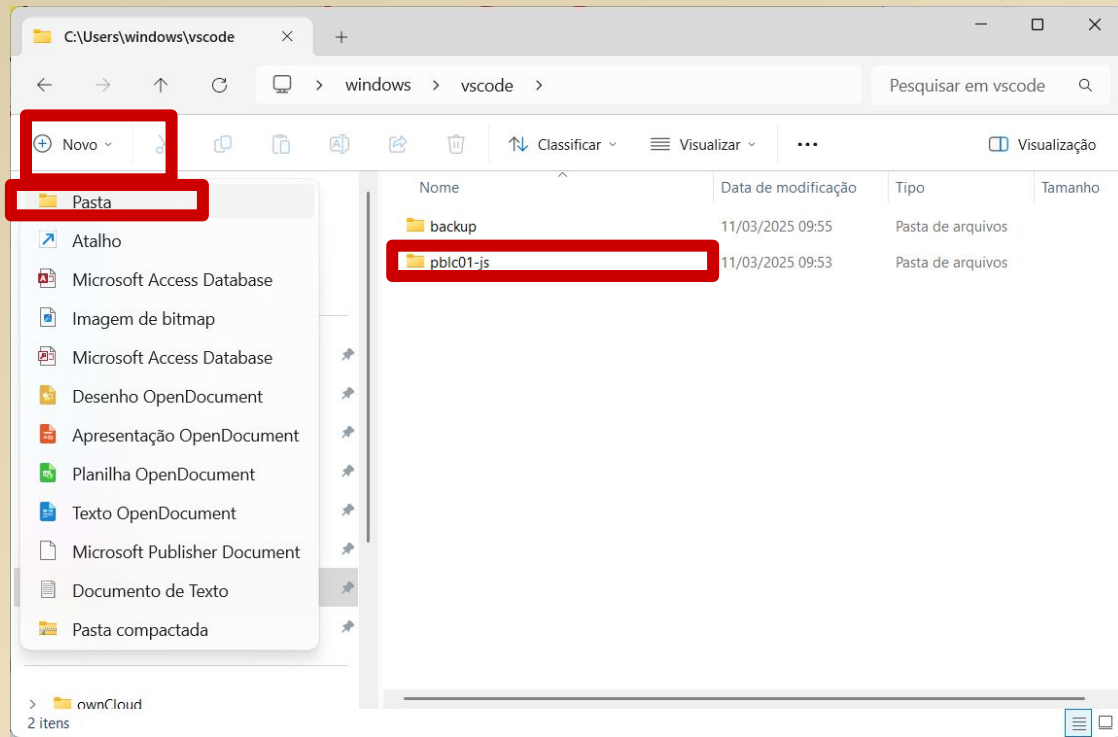
Profa. Bárbara
Pimenta Caetano



Pasta Ambiente TS

→ C:\Users\aluno\vscode\pb1c01ts

Ambiente

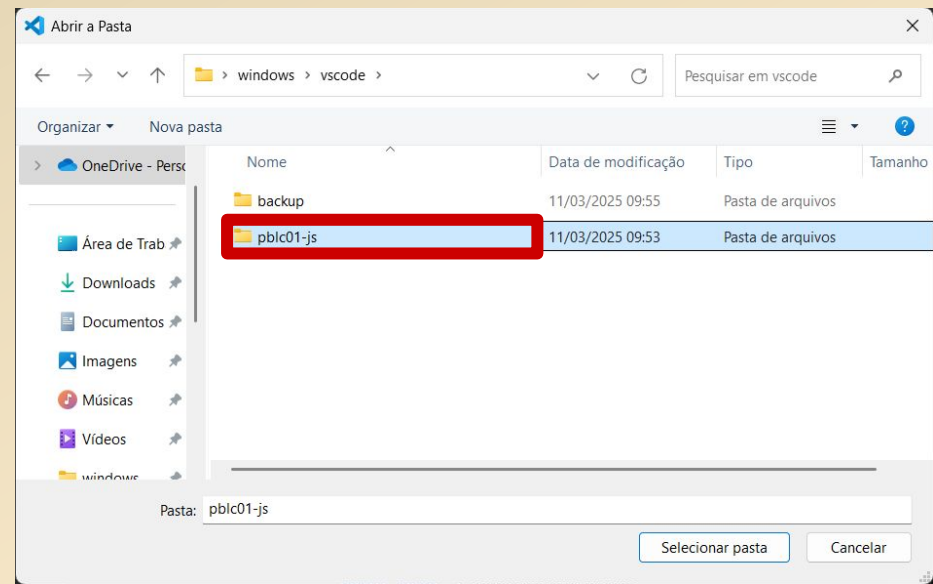
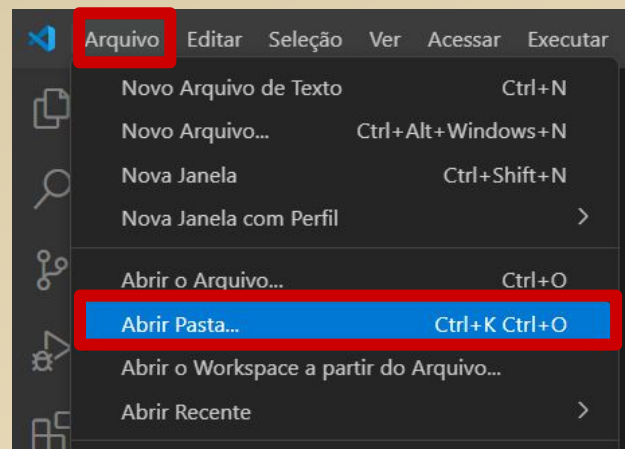




Abrir Pasta VSCode

→ C:\Users\aluno\vscode\pblic01ts

Ambiente





NPM-Node Package Manager

- Ferramenta de gerenciamento de pacotes Node.js:
- Cria ambiente isolado para aplicação
- Instala/Desinstala pacote
- Arquivo package.json com relação pacotes do ambiente.
- Para criar o package.json abra o terminal e digite:

npm init -y

- Para abrir o terminal clique no menu:

Ver | Terminal ou [Ctrl]+[']





TypeScript

→ Para instalar o TypesCRIPT abra o terminal e digite:
`npm install -g typescript`

Introdução



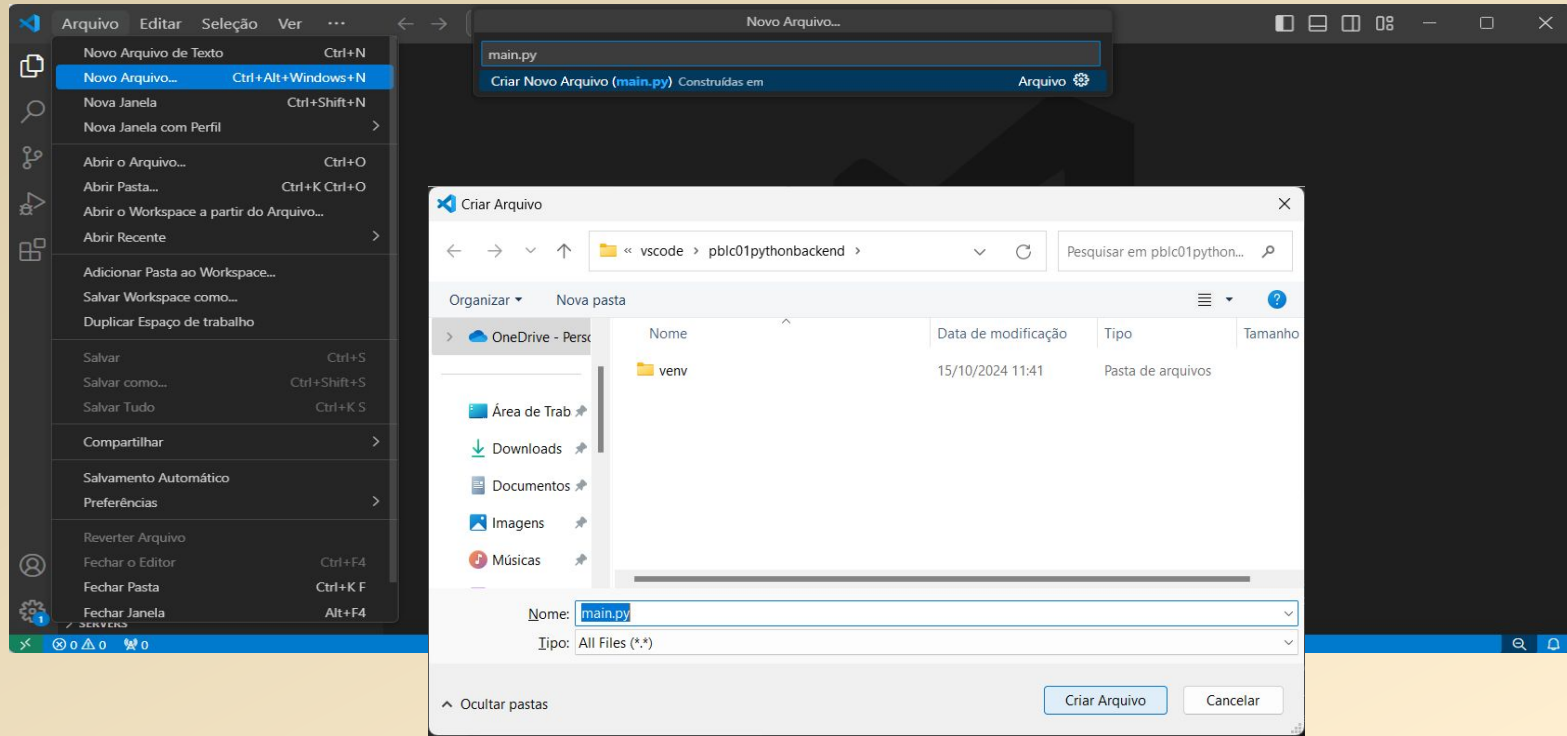


main.ts

→ Arquivo | Novo Arquivo

→ main.ts

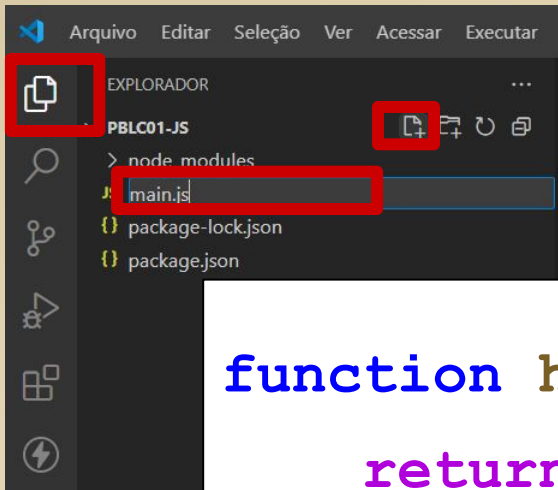
Ambiente





Criando arquivo main.ts

Ambiente



```
function hello(person: string) {  
    return "Hello, " + person;  
}  
  
let userName='Ada Lovelace';  
console.log(hello(userName));
```





Executar

→ Compilar para javaScript

tsc .\main.ts

→ Executar o javaScript

node .\main.js

Ambiente





Instalando Pacote prompt-sync

- Para instalar pacotes de dependências o pacote prompt-sync abra o terminal e digite:
npm install prompt-sync
- Os pacotes de dependências são guardados no diretório node_modules
- Porém o TypeScript não sabe automaticamente quais tipos ela usa (como tipos de função, parâmetros, etc.), porque JavaScript puro não tem tipagem estática.

error





Instalando Pacote de tipo do prompt-sync

```
npm install --save-dev @types/prompt-sync
```

Ambiente

- Esse pacote faz parte do repositório DefinitelyTyped, que contém definições de tipos para bibliotecas JavaScript populares, permitindo que o TypeScript:
- ◆ Entenda a estrutura da biblioteca;
 - ◆ Valide os tipos durante o desenvolvimento;
 - ◆ Te ajude com autocomplete, IntelliSense e verificação de erros.





Instalando Pacote de tipo do node

- Instala os tipos do Node.js (`@types/node`) como dependência de desenvolvimento no seu projeto TypeScript:

```
npm i --save-dev @types/node
```





Instalando Pacote prompt-sync

Ambiente

The screenshot shows the Visual Studio Code interface. In the Explorer sidebar on the left, the file tree for 'AULA 5 - PBL001-05B' is visible. The 'hello' folder is expanded, showing 'node_modules', 'main.js', 'main.ts', 'package-lock.json', 'package.json', and 'tsconfig.json'. The 'node_modules' and 'package-lock.json' files are highlighted with red rectangles. The main editor area displays the 'package.json' file for the 'hello' project. The file content is as follows:

```
1 {  
2   "name": "hello",  
3   "version": "1.0.0",  
4   "main": "index.js",  
5   "scripts": {  
6     "test": "echo \"Error: no test specified\" && exit 1"  
7   },  
8   "keywords": [],  
9   "author": "",  
10  "license": "ISC",  
11  "description": "",  
12  "dependencies": {  
13    "prompt-sync": "^4.2.0"  
14  },  
15  "devDependencies": {  
16    "@types/node": "^22.13.13",  
17    "@types/prompt-sync": "^4.2.3"  
18  }  
19 }  
20
```

The 'dependencies' section, including the 'prompt-sync' dependency, is highlighted with a red rectangle.



tsconfig.json

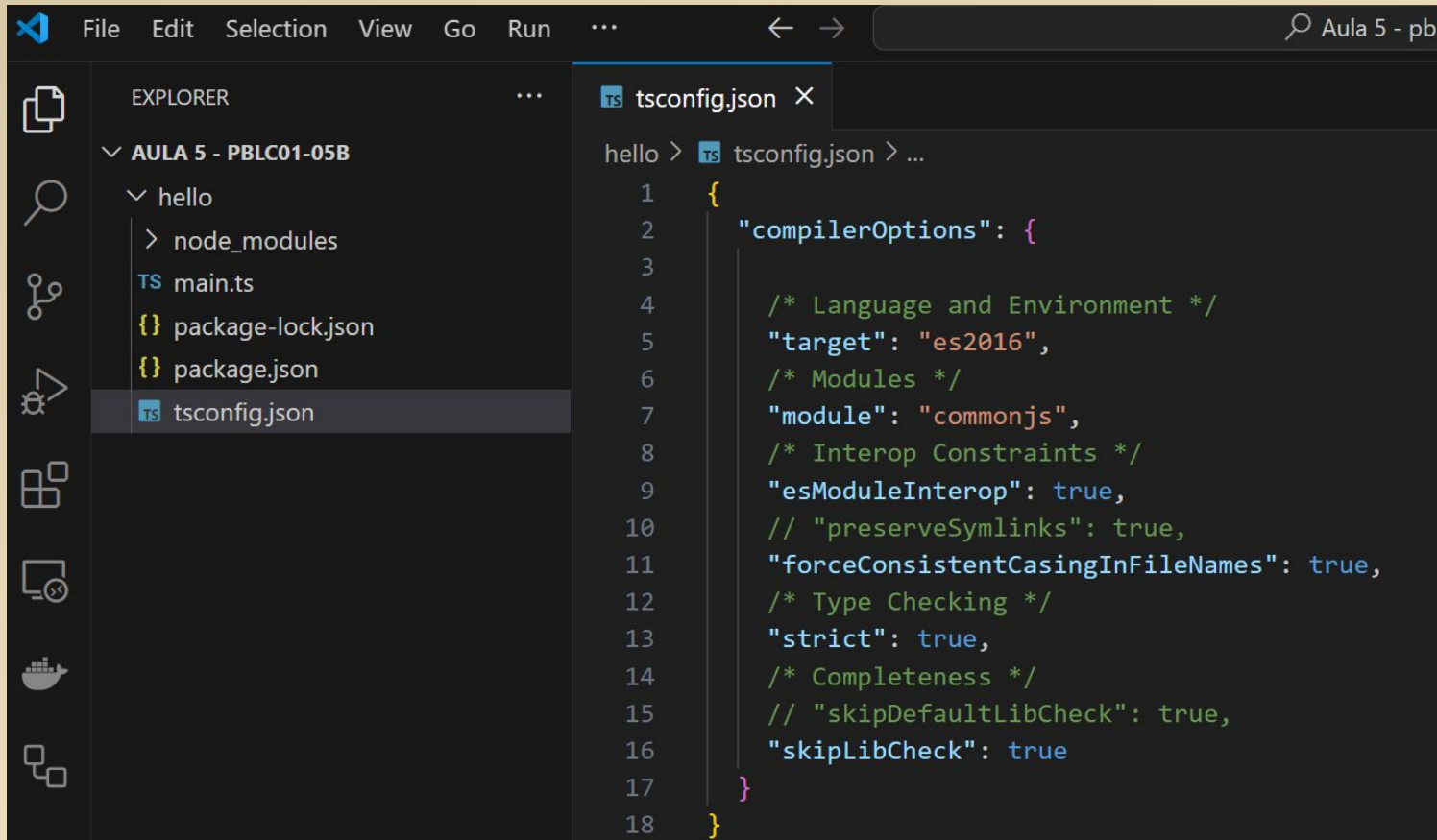
- Esse arquivo é essencial para configurar como o compilador TypeScript (tsc) vai transformar o seu código .ts em .js.
- criar automaticamente um arquivo tsconfig.json na raiz do seu projeto:
tsc --init
- Sem esModuleInterop: true,
o TypeScript pode não permitir
importar o promptSync





Ambiente

tsconfig.json



The image shows a screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project structure with a folder named 'AULA 5 - PBLCO1-05B' containing a subfolder 'hello'. Inside 'hello', there are files 'node_modules', 'main.ts', 'package-lock.json', 'package.json', and 'tsconfig.json'. The 'tsconfig.json' file is selected and its content is displayed in the main editor area. The file content is a JSON object with 'compilerOptions' set to an object containing various TypeScript compiler settings. The settings include 'target' set to 'es2016', 'module' set to 'commonjs', 'esModuleInterop' set to true, 'forceConsistentCasingInFileNames' set to true, 'strict' set to true, and 'skipLibCheck' set to true. The editor has a dark theme and a search bar at the top right.

```
hello > TS tsconfig.json > ...
1  {
2    "compilerOptions": {
3
4      /* Language and Environment */
5      "target": "es2016",
6      /* Modules */
7      "module": "commonjs",
8      /* Interop Constraints */
9      "esModuleInterop": true,
10     // "preserveSymlinks": true,
11     "forceConsistentCasingInFileNames": true,
12     /* Type Checking */
13     "strict": true,
14     /* Completeness */
15     // "skipDefaultLibCheck": true,
16     "skipLibCheck": true
17   }
18 }
```



Hello com promptSync

```
const promptSync = require("prompt-sync");
const userPrompt = promptSync();
function hello(person: string): string {
    return "Hello, " + person;
}
const userName = userPrompt("Enter your
name: ");
console.log(hello(userName));
```





Introdução



Ambiente



Tipos Simples



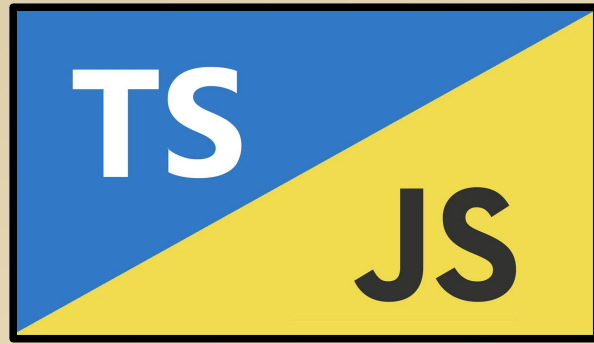
Tipos Especiais



Tipos Compostos



Outros Tipos



TypeScript - Parte 1

Prof. Enzo
Seraphim

Profa. Bárbara
Pimenta Caetano



Tipos simples

Tipos Simples

→ boolean

→ number

→ string

→ bigint





Tipos Simples

Tipos Simples

→ Tipo explícito:

- ◆ Evita erros. Deixa claro qual tipo o valor deve ser

```
let linguagem: string = "TypeScript";
```

→ Tipo implícito:

- ◆ Mesmo sem escrever o tipo, o TS sabe qual é

```
let curso = "Programação";
```

→ Erro na atribuição:

```
let idade: number = 25;
```

```
idade = "vinte e cinco";
```



Tipos Simples

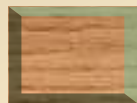
```
const promptSync = require("prompt-sync");
const userPrompt = promptSync();
let qtdText = userPrompt("Quantidade de notas:");
let qtd: number = Number(qtdText);
let soma: number = 0;
for (let i = 1; i <= qtd; i++) {
    let nTexto = userPrompt(`N ${i}: `);
    let n: number = Number(nTexto);
    soma += n;
}
let media: number = soma / qtd;
console.log("A média do aluno é:", media);
```



Introdução



Ambiente



Tipos Simples



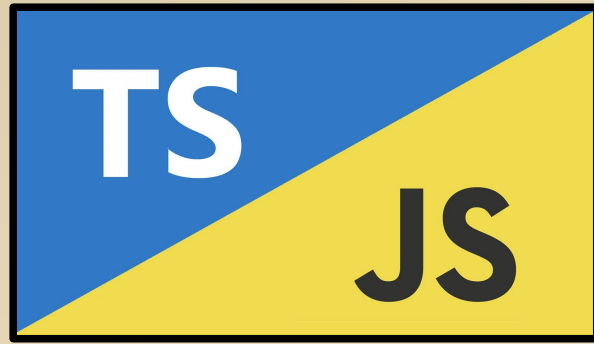
Tipos Especiais



Tipos Compostos



Outros Tipos



TypeScript - Parte 1

Prof. Enzo
Seraphim

Profa. Bárbara
Pimenta Caetano



Tipos Especiais

Tipos Especiais

→ any (explícito)

- ◆ desativa a verificação de tipos

→ unknown | as (explícito)

- ◆ exige verificação ou conversão de tipo antes de usar.

→ never

- ◆ é usado quando uma função nunca retorna — seja por erro ou loop infinito.

→ undefined

- ◆ ausência de valor - não atribuído

→ null

- ◆ ausência de valor - intencionalmente vazio





Tipos Especiais

Tipos Especiais

→ any

```
let valor: any = 10;  
valor = "texto";  
valor = true;  
  
console.log("Valor (any):", valor);
```





Tipos Especiais

Tipos Especiais

→ unknown | as (casting)

```
let dado: unknown = "123";  
// console.log(dado.toUpperCase()); // Erro  
//as garante que é uma string  
//fazendo conversão de tipo  
let texto: string = dado as string;  
console.log("Texto (unknown → string):",  
texto.toUpperCase());
```





Tipos Especiais

Tipos Especiais

→ never

```
function erroFatal(mensagem: string): never
{
    throw new Error(mensagem);
}

erroFatal("Algo deu muito errado!");
```





Tipos Especiais

Tipos Especiais

→ undefined e null

```
let indefinido: undefined = undefined;  
let nulo: null = null;  
  
console.log("Indefinido:", indefinido);  
console.log("Nulo:", nulo);
```





Introdução



Ambiente



Tipos Simples



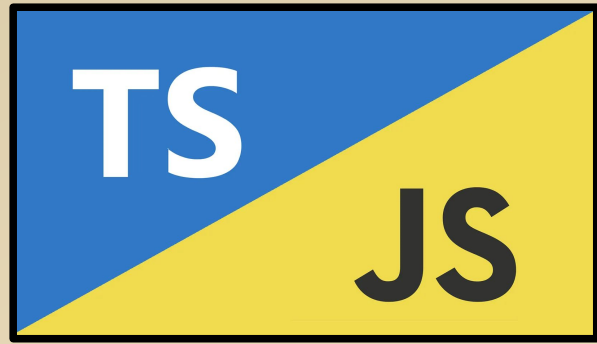
Tipos Especiais



Tipos Compostos



Outros Tipos



TypeScript - Parte 1

Prof. Enzo
Seraphim

Profa. Bárbara
Pimenta Caetano



Tipos Compostos

Tipos Compostos

→ array

```
const names: string[] = [];  
names.push("Dylan");  
console.log(names);
```

→ tuple

```
let ourTuple: [number, boolean, string];  
ourTuple = [5, false, 'jabuticaba'];  
console.log(ourTuple);
```





Tipos Compostos

Tipos Compostos

→ objeto

```
const car:{type: string, model: string, year: number}={
  type: "Toyota",
  model: "Corolla",
  year: 2009
};
console.log(car);
```

→ enum

```
enum CorFavorita {Vermelho = "vermelho",Azul = "azul",}
function mostrar(cor: CorFavorita) {
  console.log("Sua cor favorita é:", cor);
}
let minhaCor: CorFavorita = CorFavorita.Azul;
mostrar(minhaCor);
```





Introdução



Ambiente



Tipos Simples



Tipos Especiais



Tipos Compostos



Outros Tipos



TypeScript - Parte 1

Prof. Enzo
Seraphim

Profa. Bárbara
Pimenta Caetano



Outros tipos

→ Tipo Alias

- ◆ Define apelidos para tipos

→ Tipo União

- ◆ Permitem que uma variável aceite múltiplos tipos.

→ Funções

- ◆ Declare os tipos de entrada e saída
- ◆ Quando não retorna nada = void





Tipos Especiais

Outros Tipos

→ Tipo Alias

```
type Nome = string;
type Idade = number;
type Pessoa = {
  nome: Nome;
  idade: Idade;
};
const aluno: Pessoa = {
  nome: "Ana",
  idade: 22
};
console.log("Aluno:", aluno);
```





Outros Tipos

→ Tipo União

```
let valor: string | number;  
valor = "Texto";  
console.log("Valor:", valor);  
valor = 123;  
console.log("Valor:", valor);
```





Tipos Especiais

Outros Tipos

→ Funções

```
function soma(a: number, b: number): number {  
    return a + b;  
}  
  
let resultado: number = soma(5, 3);  
console.log(resultado);  
  
//void  
function printHello(): void {  
    console.log('Hello!');  
}  
  
printHello();
```



**Prof. Enzo
Seraphim**

**Profa. Bárbara
Pimenta Caetano**

Os logotipos, marcas comerciais e nomes de produtos citados nesta publicação tem apenas o propósito de identificação e podem ser marcas registradas de suas respectivas companhias.



TypeScript - Parte 1