UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

# Fundamentos de TypeScript

## QXD0253 - Desenvolvimento de Software para Web

Prof. Bruno Góis Mateus (brunomateus@ufc.br)

# Agenda

- Introdução
- Configurando Ambiente
- Sistemas de Tipo e Inferência de Tipo
- Orientação a objetos
- Criando projetos com TypeScript

# Introdução

# Introdução

## TypeScript

- É uma linguagem fortemente tipada construída em cima do JavaScript
  - Criada e mantida pela **Microsoft**
- Superconjunto do **JavaScript *(typed)***
  - Qualquer programa em JS é um programa válido em TS
- Projetada para ser usada no Front-end e no Back-end
  - Compilada / Traduzida para JS

# Introdução

## Características

- TypeScript é um superset do ECMAScript 2015, logo ela herda as seguintes características:
  - Classes
  - Módulos
  - Funções Arrow
  - Parâmetros opcionais e com valores padrão

# Introdução

## Características

- Tem como principal característica o sistema de tipos
  - É possível determinar os tipos de variáveis e parâmetros
  - Por meio de checagem estática erros de tipo são capturados mais cedos
  - Em várias ocasiões, a declaração do tipo é desnecessária devido ao mecanismo de inferência de tipos

# Introdução

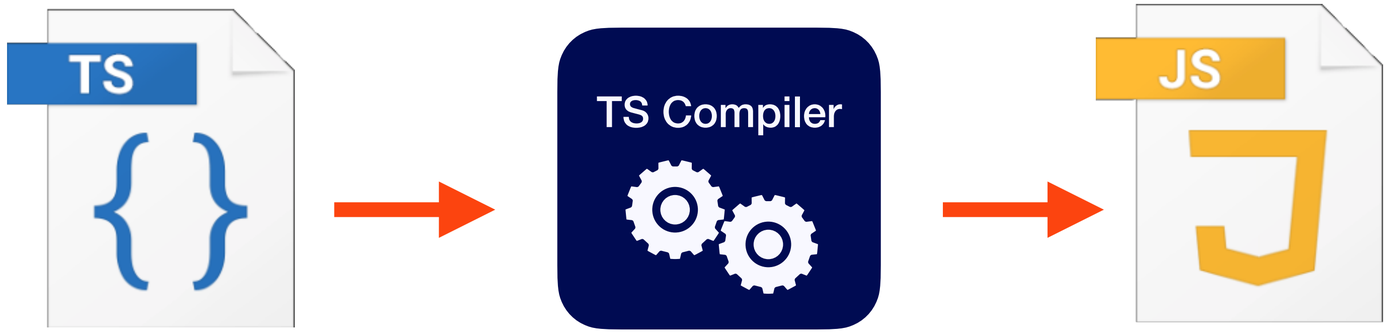
## Características

- Outra features de código introduzidas pelo TS são:
  - Interfaces
  - Classes Abstratas
  - Modificadores de visibilidade
  - Generics
  - Optionals
  - Function Overloading
  - Decorators e mais

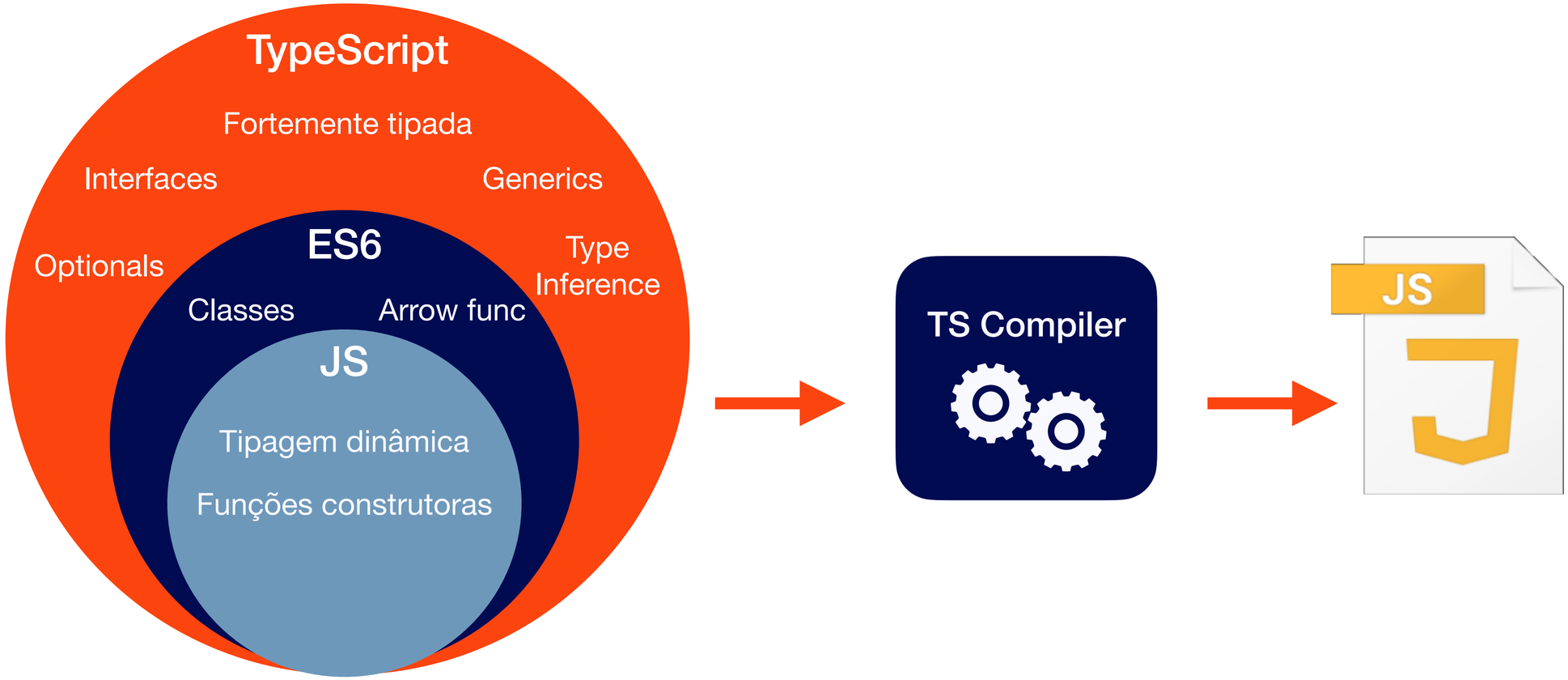
# Navegadores entendem TypeScript ?

# Introdução

- É necessário converter TS para JS
- Este processo é feito por um compilador ou uns transpilador
- TS pode ser executado em qualquer ambiente que suporte JS



# Introdução


TypeScript
Fortemente tipada
Interfaces
Generics
Optionals
Type Inference
ES6
Classes
Arrow func
JS
Tipagem dinâmica
Funções construtoras
TS Compiler
JS

# Introdução

## Vantagens

- Checagem de tipos estática (opcional)
  - O compilador alerta sobre erros relacionados a tipagem
- Detecção de bugs mais cedo
  - O compilador alerta sobre erros que seriam apenas descobertos durante a execução
- Maior legibilidade
- Otimizações de código
  - Type annotation, Generics, API Documentation, Intellisense

# Introdução

## Diferenças

### JavaScript

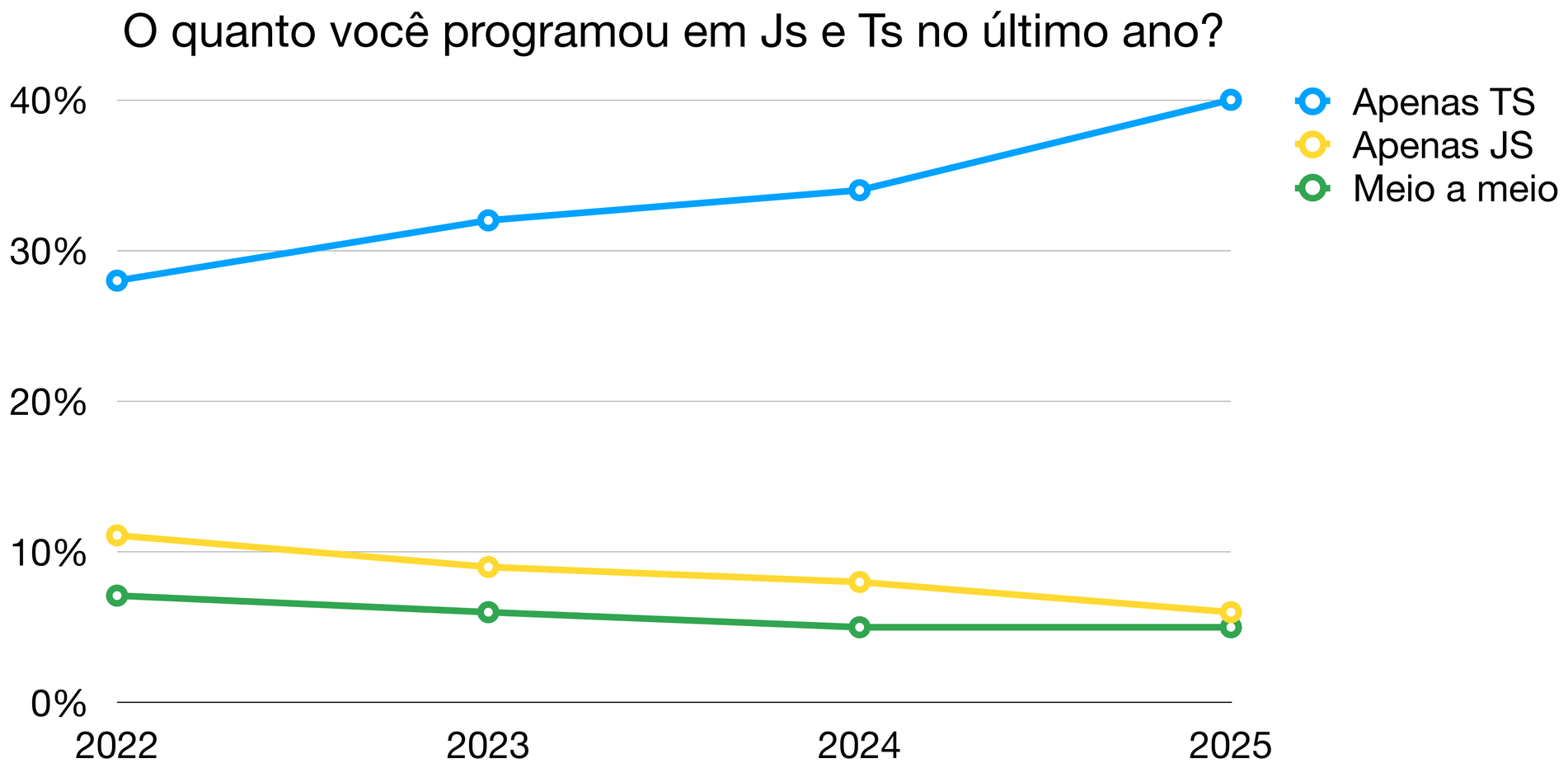
- Criada por Brendan Eich (Netscape) e lançada em 1995
- Linguagem de script leve com foco em ajudar o desenvolvimento de páginas web dinâmicas
- Fracamente tipada e c/ suporte somente a tipagem dinâmica
- Mais adequada para pequenos projetos
- Executada diretamente nos navegadores
- Bibliotecas JS funcionam por padrão

### TypeScript

- Criada pela Microsoft e lançada em 2012
- Superset de JS criada para lidar com a complexidade de código em grandes projetos OO
- Fortemente tipada c/ suporte a tipagem estática e dinâmica
- Mais adequada para aplicações maiores
- Convertida em JS capaz de ser executado por navegadores
- Bibliotecas JS funcionam por padrão

# Introdução

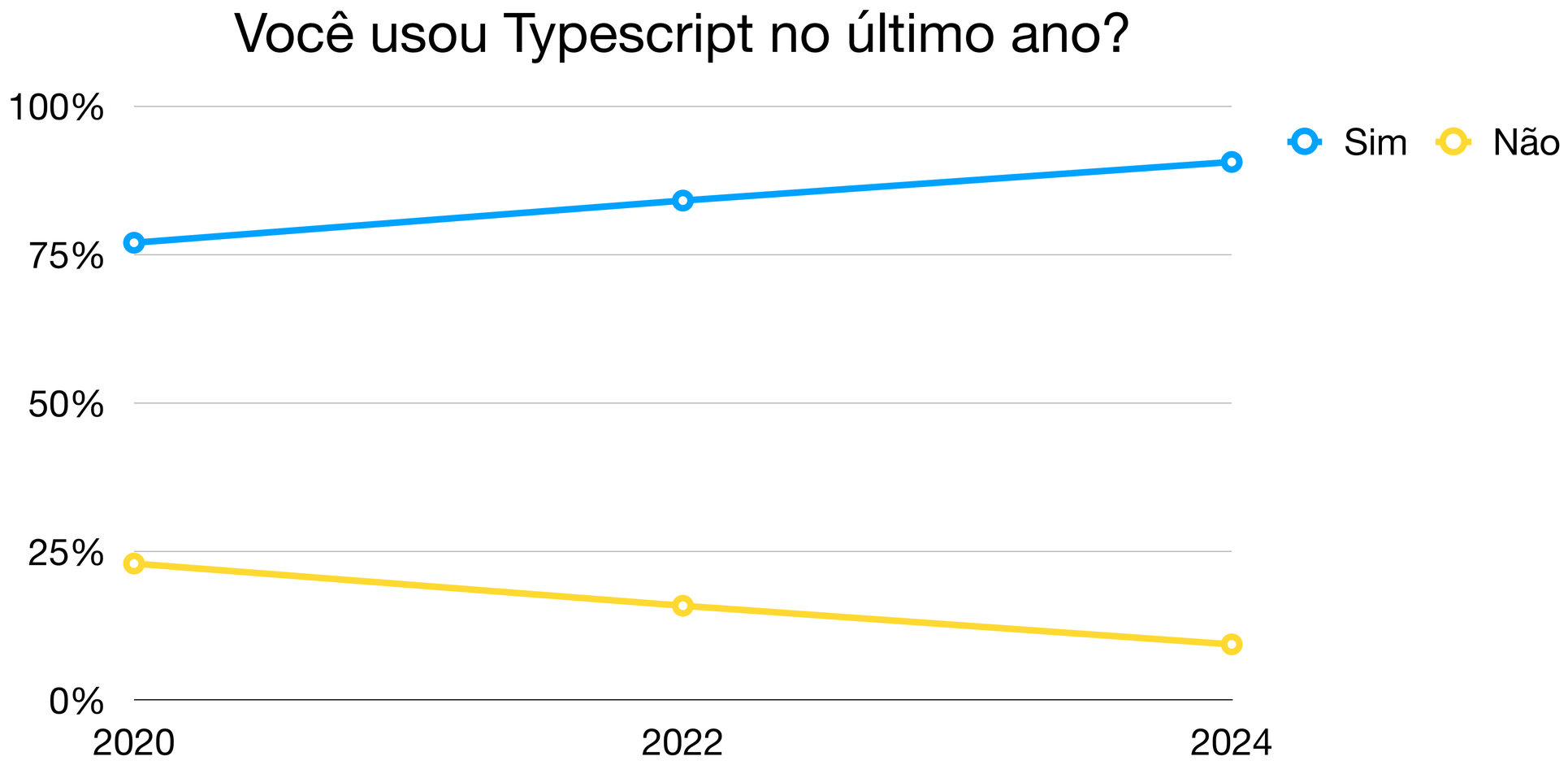
## Comunidade TS




Fonte: stateofjs.com

# Introdução

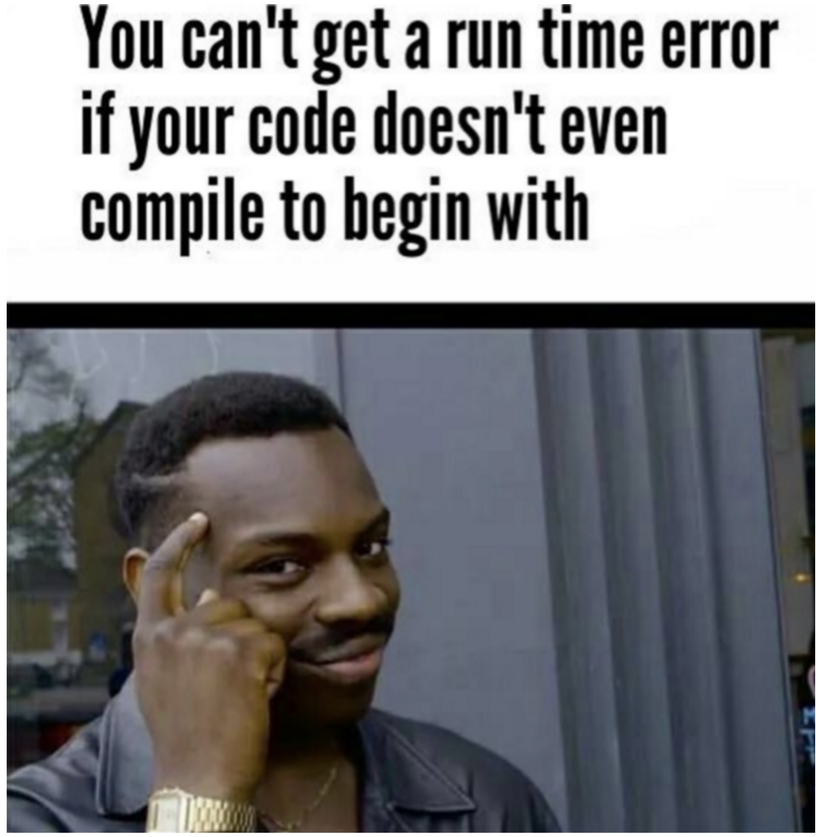
## Comunidade TS

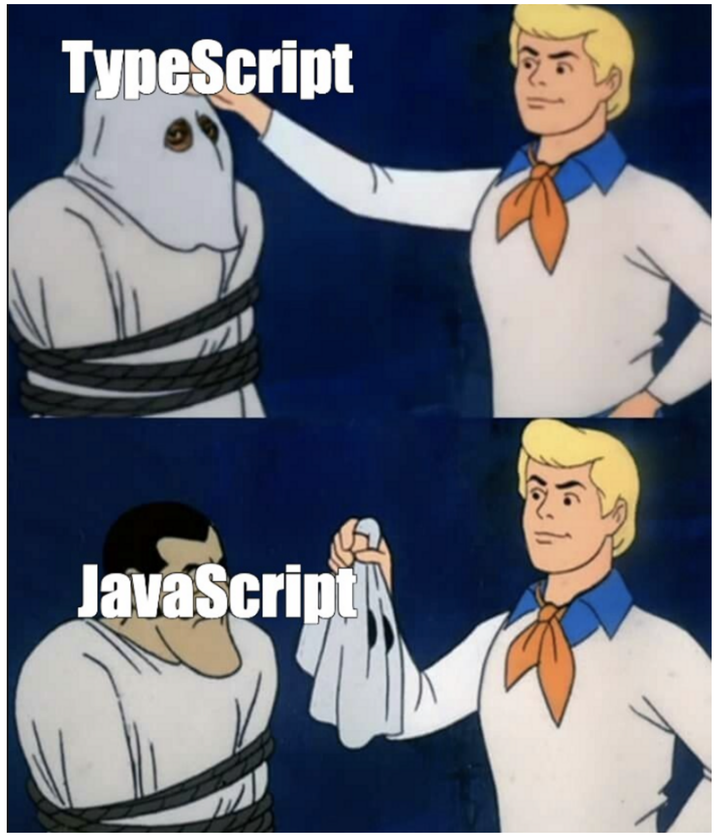



Fonte: State of Frontend

# Introdução

YOU COME TO ME AT RUNTIME
TO TELL ME THE CODE YOU ARE EXECUTING DOES NOT COMPILE


You can't get a run time error if your code doesn't even compile to begin with


TypeScript
JavaScript

# Configurando o ambiente

# Configurando o ambiente

## Passa a passo

1. Instalar o Node
   - Windows: https://nodejs.org/en/download/
   - Linux: `$ sudo apt-get install node`
   - Mac: `$ brew install node`
   - Node Version Manager
2. Instalar o gerenciador de pacotes do Node
   - Linux: `$ sudo apt-get install npm`

# Configurando o ambiente

## IDE recomendada

Visual Studio Code

# Configurando o ambiente

## Compilando e executando

1. Compilando um arquivo .ts
   - `$ tsc meu_arquivo.ts`
2. Executar .js gerado com o node
   - `$ node meu_arquivo.js`

# Configurando o ambiente

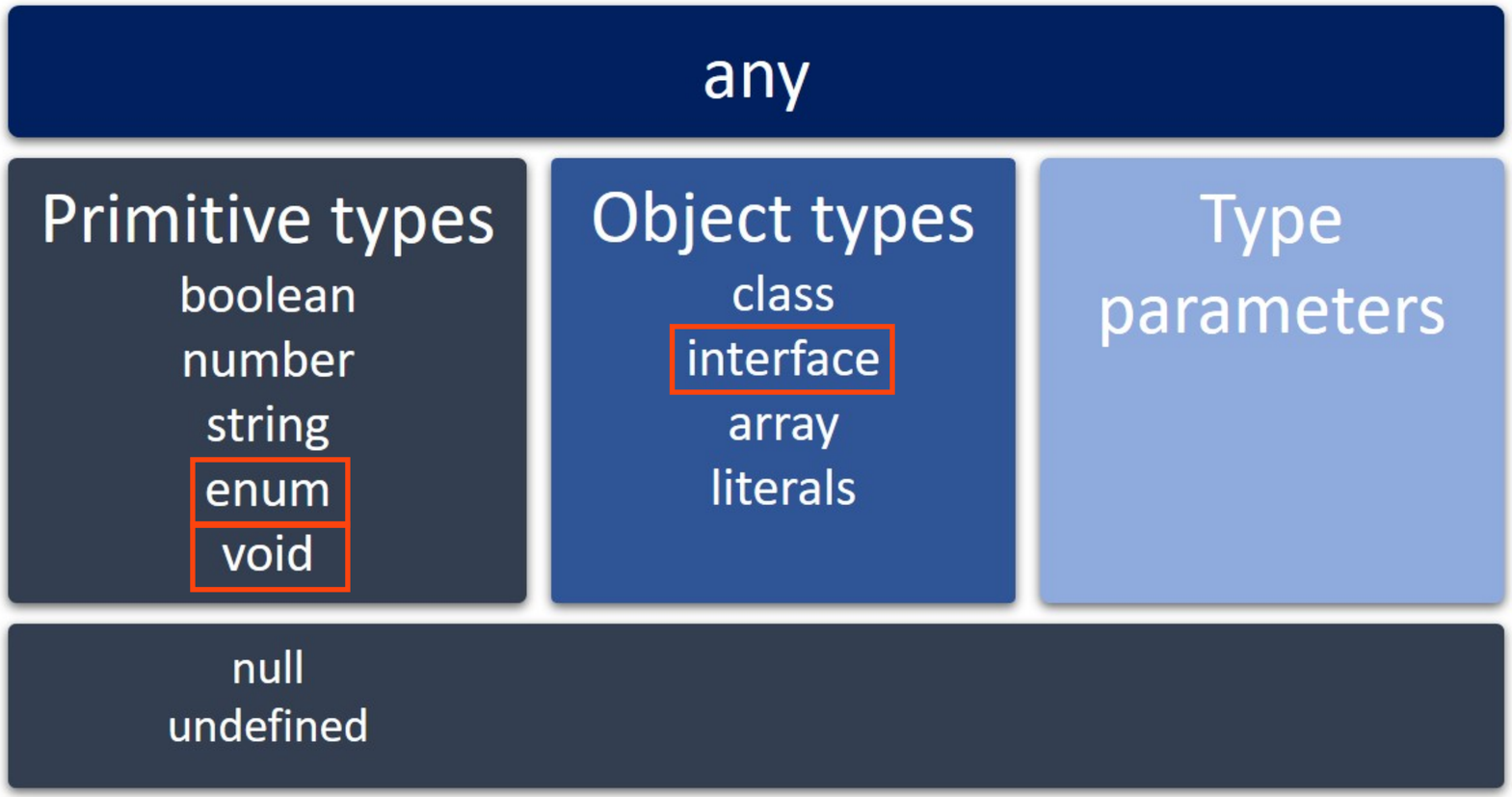
## Configurando o TypeScript - tsconfig.json

- Define a raiz do projeto e as opções de compilação requeridas pelo projeto
  - AlwaysStrict
  - TypeChecking
  - JavaScriptSupport
  - StrictNullChecks
  - Lib

Referência completa:
https://www.typescriptlang.org/tsconfig

# Sistema de Tipo e Inferência de Tipo

# Sistema de Tipo e Inferência de Tipo



# Sistema de Tipo e Inferência de Tipo

## Declaração de variáveis

- É possível definir o tipo de cada variável
- Ao tentar atribuir um valor do tipo errado, o compilador acusa o erro

```
let flag: boolean;
let x: number;
let big: bigint = 100n;
let firstName: string = "Mateo";

x = "10"; //Erro!
```

# Sistema de Tipo e Inferência de Tipo

## Union

- Descrevem valores que podem assumir mais de um tipo
- Restringe a atribuição aos tipos especificados
- Intellisense

```
let multiType: number | boolean;
multiType = 20;          //* Valid
multiType = true;        //* Valid
multiType = "twenty";    //* Invalid
```

# Sistema de Tipo e Inferência de Tipo

## Union

- Tipos literais

```
type testResult = "pass" | "fail" | "incomplete";
let myResult: testResult;
myResult = "incomplete";     //* Valid
myResult = "pass";           //* Valid
myResult = "failure";

type dice = 1 | 2 | 3 | 4 | 5 | 6;
let diceRoll: dice;
diceRoll = 1;     //* Valid
diceRoll = 2;     //* Valid
diceRoll = 7;     //* Invalid
```

# Sistema de Tipo e Inferência de Tipo

## Type Guards

| Type | Predicate |
|---|---|
| string | typeof s === "string" |
| number | typeof n === "number" |
| boolean | typeof b === "boolean" |
| undefined | typeof undefined === "undefined" |
| function | typeof f === "function" |
| array | Array.isArray(a) |

# Sistema de Tipo e Inferência de Tipo

## Enum

- Fornecem uma maneira fácil de trabalhar com um conjunto restrito de valores
- Um enum, é um nome simbólico para um conjunto de valores
  - São tratados como um tipo de dado
- Quando uma função ou variável aceita um número limitado de valores, considere utilizar enum

```
enum Direcoes {
  Cima,
  Baixo,
  Esquerda,
  Direita
}
```

# Sistema de Tipo e Inferência de Tipo

## Enum

- Vantagens:
  - Ajuda na redução de erros relacionado
  - Facilita mudanças futuras
  - Aumenta a clareza e a legibilidade do código quando utilizadas com sabedoria

# Sistema de Tipo e Inferência de Tipo

## Tuplas

- São tipos especiais de Array
- Permitem a criação de Arrays com tipos mistos
- Dimensão e tipos restritos

```
let person1: [string, number] = ['Marcia', 35];
let person2: [string, number] = ['Marcia', 35, true]; // Erro
let person3: [string, number] = [35, 'Marcia']; // Erro
```

# Sistema de Tipo e Inferência de Tipo

## Funções

```
let addNumbers = function (x: number, y: number): number {
    return x + y;
}

let addNumbers1 = (x: number, y: number): number => x + y;


function addNumbers2 (x: number, y?: number): number {
    return x + y;
}

function addNumbers3 (x: number, y = 25): number {
    return x + y;
}
```

# Sistema de Tipo e Inferência de Tipo

## Funções

```
type Message = {
   text: string;
   sender: string;
}

function displayMessage({text, sender}: Message) {
    console.log(`Message from ${sender}: ${text}`);
}
```

# Orientação a Objetos

# Orientação a Objetos

## Interfaces

- São construções de *design-time*
- São removidas durante a transpilação para JS
- É possível usá-las sem a necessidade de envolvermos classes
  - Definem uma estrutura de dados

# Orientação a Objetos

## Interfaces

```
interface IceScream {
  flavour: string
}

interface Sundae extends IceCream {
    sauce: 'chocolate' | 'caramel' | 'strawberry';
    nuts?: boolean;
    whippedCream?: boolean;
    instructions?: boolean;
}
```

# Orientação a Objetos

## Classes

- Classes em TS possuem modificadores de acesso
  - public (default), protected, private
- É possível definir métodos e atributos de classe
- Podemos definir classes abstratas

# Orientação a Objetos

## Classes

```
interface Enviavel {
    enviar(): boolean;
}

abstract class Notificacao implements Enviavel {
    private readonly _id: string;// Apenas visível nesta classe e imutável
    protected destinatario: string;
    public mensagem: string;
    private static totalNotificacoesEnviadas: number = 0;

    constructor(destinatario: string, mensagem: string) {
        this._id = Math.random().toString(36).substr(2, 9);
        this.destinatario = destinatario;
        this.mensagem = mensagem;
    }
}
```

# Orientação a Objetos

## Classes

```
abstract class Notificacao implements Enviavel {
    get id(): string {
        return this._id;
    }

    public static obterTotalEnviado(): number { }

    protected incrementarContador(): void { }

    abstract validarDestinatario(): boolean;

    public enviar(): boolean { }
}
```

# Orientação a Objetos

## Classes

```
class NotificacaoEmail extends Notificacao {
    private assunto: string;

    constructor(destinatario: string, assunto: string, mensagem: string) {
        super(destinatario, mensagem);
        this.assunto = assunto;
    }

    // Implementação obrigatória do método abstrato
    public validarDestinatario(): boolean {}

    public override enviar(): boolean {}
        return false;
    }
}
```

# Criando projetos com TypeScript

# Criando projetos com TypeScript

## Frontend Vanilla - Sem frameworks

```
// Usando o template do vite para aplicações Vanilla com Ts
npm create vite@latest nome_da_sua_app -- --template vanilla-ts

//ou
npm create vite@latest nome_da_sua_app
// após definir o pacote
x Vanilla
x TypeScript


// Para detalhes acesse: https://vite.dev/guide/#trying-vite-online
```

Nome da sua aplicação

# Criando projetos com TypeScript

## Frontend Vanilla - React / Vue

```
// Usando o template do vite para aplicações com suporte a framework
npm create vite@latest nome da sua app
// após definir o pacote
x React / Vue
x TypeScript


// Para detalhes acesse: https://vite.dev/guide/#trying-vite-online
```

Nome da sua aplicação

# Criando projetos com TypeScript

## Criando um projeto

```
npm init --yes
npm install express cors
npm install -D typescript
npm install -D @types/node @types/express @types/cors

# TSX garante a execução dos arquivos com hot reload
npm install --save-dev tsx
npx tsc --init
```

# Criando projetos com TypeScript

## Sugestão de tsconfig

```
{
  "compilerOptions": {
    "target": "ES2020",
    "module": "ESNext",
    "moduleResolution": "NodeNext",
    "rootDir": "src",
    "outDir": "dist",
    "esModuleInterop": true,
    "strict": true,
    "skipLibCheck": true
  },
  "include": ["src"]
}
```

# Criando projetos com TypeScript

## Configurando os scripts

```
{
  "name": "meu-projeto",
  "version": "1.0.0",
  "type": "module",
  "scripts": {
    "dev": "tsx watch src/index.ts",
    "start": "tsx src/index.ts",
    "build": "tsc",
    "typecheck": "tsc --noEmit"
  },
  "devDependencies": {
    "tsx": "^4.0.0",
    "typescript": "^5.6.0",
    "@types/node": "^20.0.0"
  }
}
```

# Referências

- Documentação oficial do TypeScript
- TypeScript vs. JavaScript - Avelon Pang
- Typescript vs Javascript - Diffrence you should know
- State of Js
- Mini Curso gratuito de TypeScript - Willian Justen
- Build JavaScript applications using TypeScript

# Por hoje é só