

An aerial photograph of a university campus. The image shows a large green lawn in the center, surrounded by various buildings, some with red roofs and others with white roofs. There are several ponds and a winding path. The campus is bordered by a dense forest on the right side. The text "WEB ACADEMY" is overlaid in a large, bold, dark blue font.

# WEB ACADEMY

## Frameworks Front-end

Daniel Augusto Nunes da Silva

# **Apresentação**

# Ementa

- **Frameworks Front-end. TypeScript.** Introdução ao **Angular**. Interface de linha de comando (**CLI**) do Angular. Espaços de trabalho e estrutura de projetos no Angular. Execução e *deploy*. **Componentes**, serviços e rotas. **Comunicação com aplicações *back-end*** por meio do protocolo HTTP.



# Objetivos

- **Geral:** Conhecer os principais **procedimentos e técnicas** de desenvolvimento de aplicações para a WEB utilizando **frameworks front-end**, com **ênfase no Angular** e seus principais recursos.
- **Específicos:**
  - Discutir as principais funcionalidades de um framework front-end;
  - Apresentar o TypeScript e sua relação com o JavaScript;
  - Conhecer os fundamentos e principais recursos do framework Angular.
  - Capacitar o aluno para utilização do framework Angular na construção de uma aplicação front-end que se comunica com um serviço back-end.

# Conteúdo programático

## Introdução

- Frameworks front-end;
- Recursos básicos de frameworks front-end;
- Roteamento no lado cliente;
- Introdução ao TypeScript;
- Definição de tipos em TypeScript.

## Fundamentos

- O que é o Angular?;
- Angular CLI;
- Espaços de trabalho;
- Estrutura do projeto, principais arquivos e diretórios.
- Execução e deploy da aplicação.

## Componentes

- Visão geral dos componentes no Angular;
- Templates;
- Diretivas;
- Rotas.

## Comunicação com o back-end

- Serviços e injeção de dependência;
- O serviço HTTP Client;
- Observables (RxJS);
- Interceptadores.

# Bibliografia



## JavaScript e JQuery: desenvolvimento de interfaces web interativas.

Jon Duckett  
1ª Edição – 2016  
Editora Alta Books  
ISBN 9781118871652



## The TypeScript Handbook

Microsoft  
<https://www.typescriptlang.org/docs/handbook/intro.html>



## Engenharia de Software Moderna

Marco Tulio Valente  
<https://engsoftmoderna.info/>



# Sites de referência

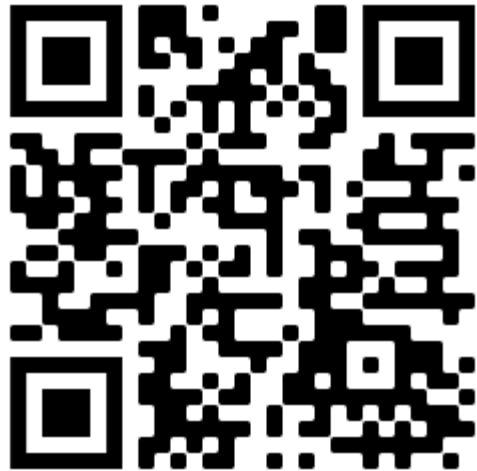
- Angular Docs.
  - <https://v17.angular.io/docs>
- TypeScript Documentation.
  - <https://www.typescriptlang.org/docs/>
- MDN Web Docs: Aprendendo desenvolvimento web.
  - <https://developer.mozilla.org/pt-BR/docs/Learn>
- Angular Tutorial (VS Code).
  - <https://code.visualstudio.com/docs/nodejs/angular-tutorial>

# Ferramentas: Extensões do VS Code

- **Visual Studio Code:** <https://code.visualstudio.com/Download>
- **Angular Language Service (Extensão do VS Code):**  
<https://marketplace.visualstudio.com/items?itemName=Angular.ng-template>
- **Node.js (e npm):** <https://nodejs.org/dist/v20.14.0/node-v20.14.0-x64.msi>
- **Angular CLI:** <https://v17.angular.io/guide/setup-local>
  1. Instalar Node.js v20
  2. `npm install -g @angular/cli@17.3.10`



# Contato



<https://linkme.bio/danielnsilva/>

# Introdução

# Frameworks front-end (lado cliente)

- **Frameworks** fornecem ferramentas que **simplificam as operações comuns de desenvolvimento**.
- Em geral, **frameworks front-end são sinônimo de frameworks JavaScript**, utilizados para construção aplicações com interface de usuário.
- Construir páginas com JavaScript consiste sobretudo em manipular o DOM (utilizar métodos como *createElement*, *appendChild*, etc.).
- Dificuldade: **toda vez que alteramos algum dado da aplicação, é necessário atualizar a interface do usuário**, o que implica em muitas linhas de código para manipular o DOM.

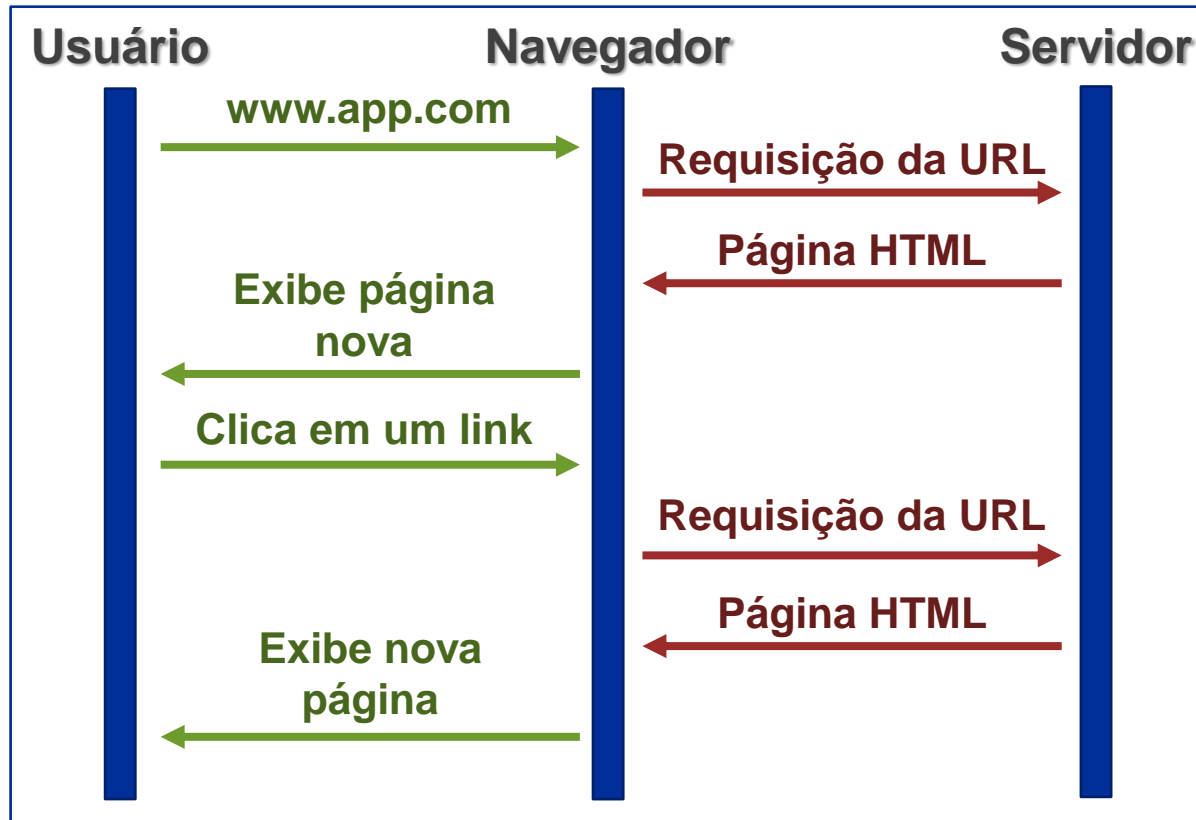
# O que um framework front-end fornece?

- Um **modo declarativo de construção de interfaces de usuário**, permitindo que um grande volume de código para manipular o DOM seja resumido em algumas linhas de código que descrevem como a interface deve funcionar.
- **Conjunto de ferramentas** que simplificam várias tarefas do processo de desenvolvimento.
- **Organização da aplicação em componentes** e/ou módulos reutilizáveis.
- **Roteamento no lado cliente**, onde a atualização do conteúdo da página, por meio de requisições assíncronas e manipulação do DOM, geram novas “pseudo-páginas” que se comportam como URLs diferentes.

# Roteamento no lado cliente e no servidor

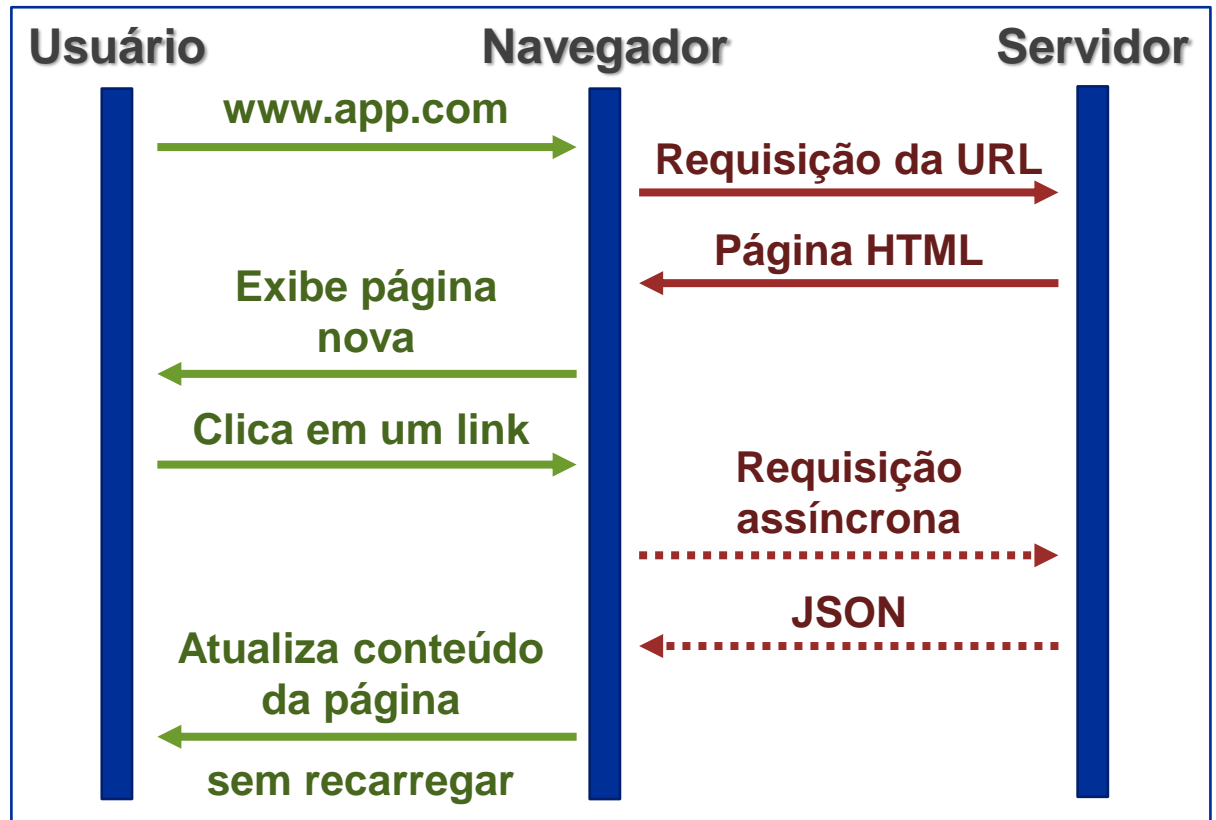
## Roteamento no servidor

Páginas HTML



## Roteamento no cliente

Single Page Application (SPA)

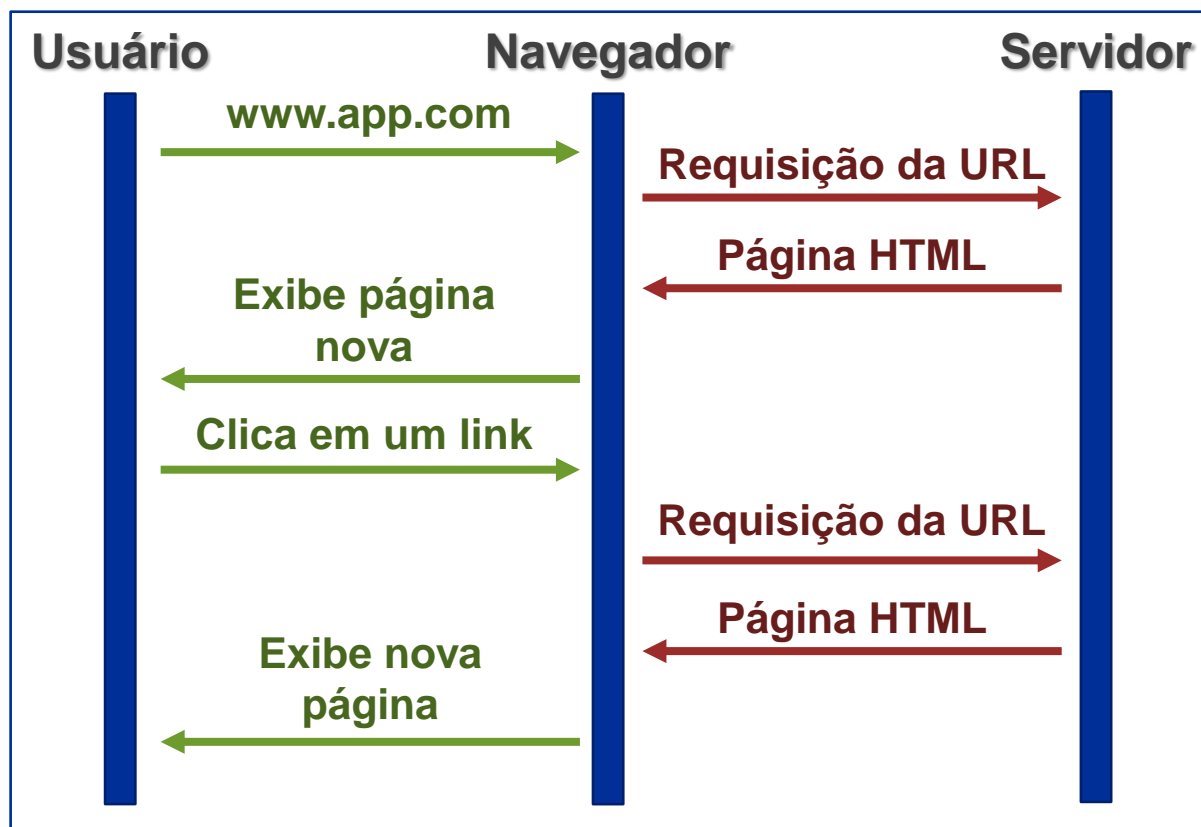




# Roteamento no lado cliente e no servidor

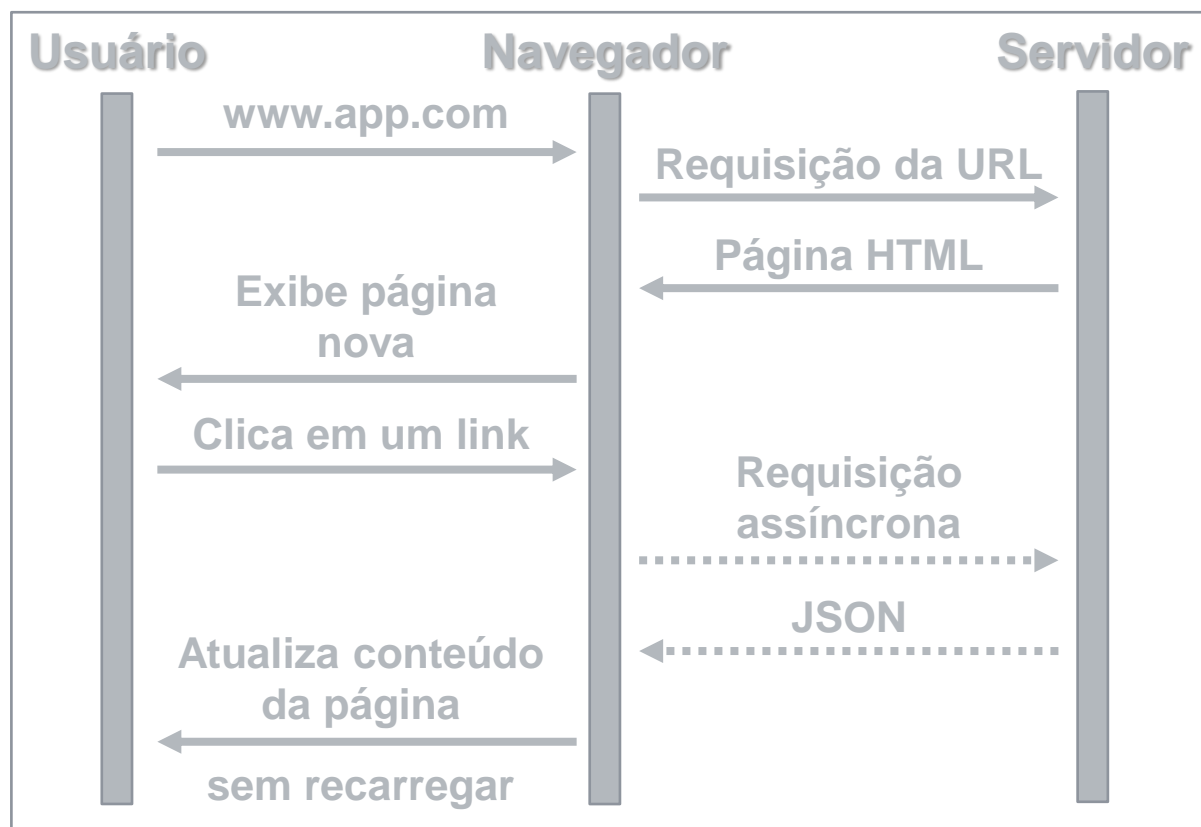
## Roteamento no servidor

Páginas HTML



## Roteamento no cliente

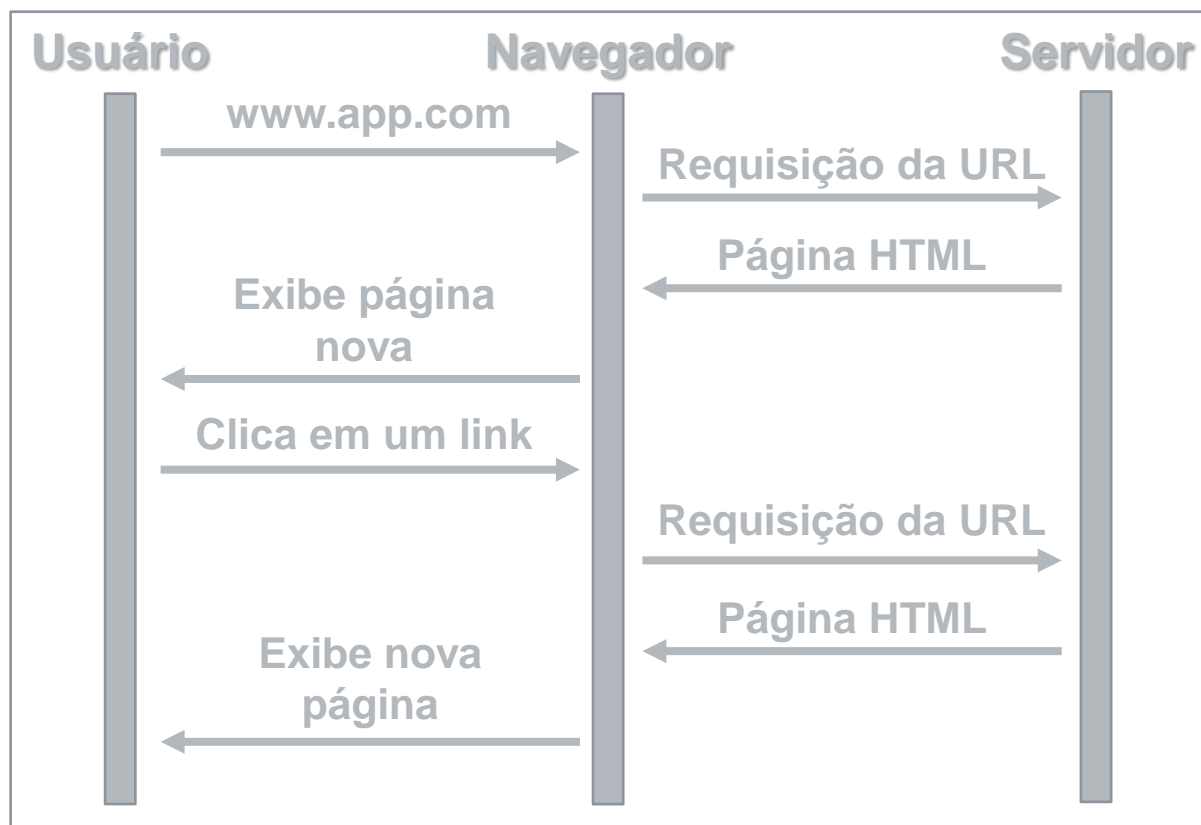
Single Page Application (SPA)



# Roteamento no lado cliente e no servidor

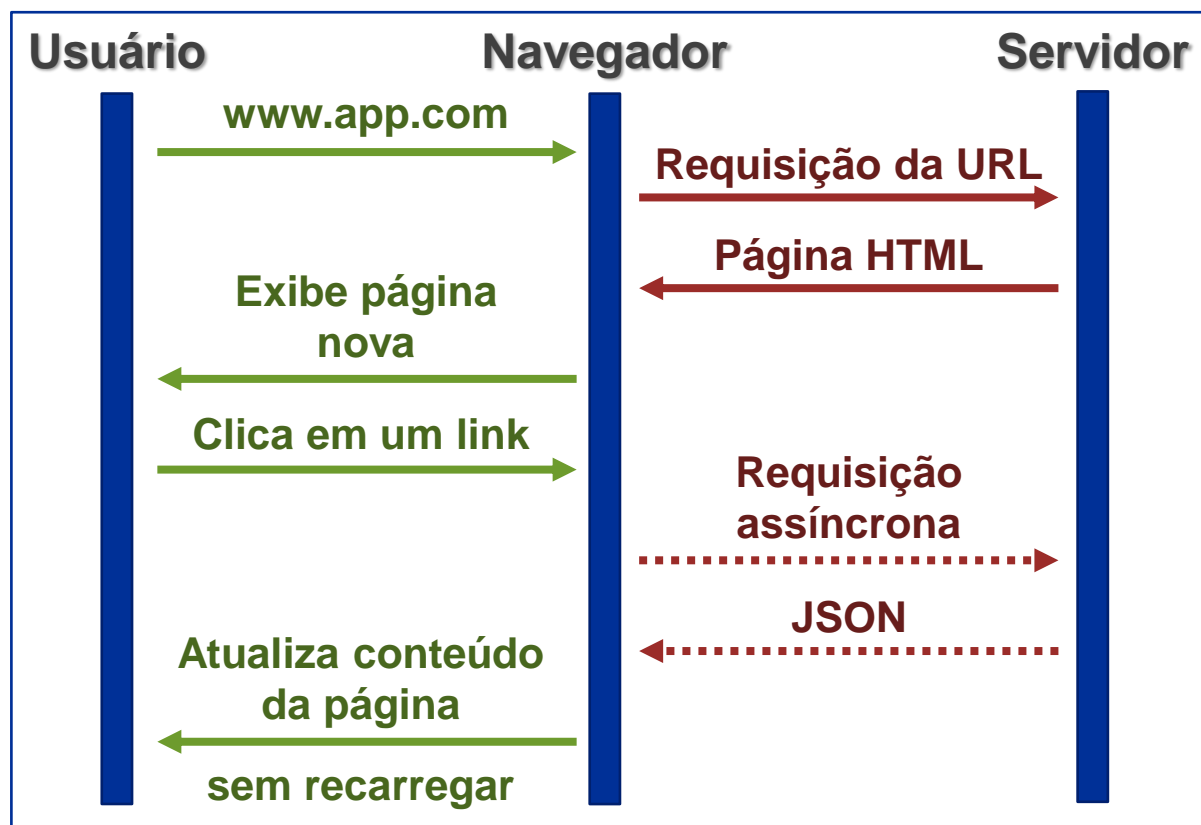
## Roteamento no servidor

Coleção de páginas HTML



## Roteamento no cliente

Single Page Application (SPA)



# Introdução ao TypeScript


- **TypeScript** é uma linguagem de programação desenvolvida pela Microsoft que pode ser considerada um **superset do JavaScript**, adicionando algumas funcionalidades.
- Maior parte de um código JavaScript válido também é um código TypeScript válido.
- No processo de **compilação**, o código **TypeScript é convertido para JavaScript**.
- Sendo o mesmo código em tempo de execução, não existem problemas de compatibilidade nos navegadores ao migrar a aplicação de uma linguagem para outra.
- TypeScript = JavaScript + **Verificação de tipos** (análise estática).



# TypeScript: definição de tipos

## Inferência de tipos do JavaScript

1. `let stringA = "WebAcademy";`
2. `let stringB: string = "WebAcademy";`

  
*stringA* e *stringB*  
são do mesmo tipo.

## Definição de tipos

1. `class Usuario {`
2.     `id: number | undefined;`
3.     `nome: string | undefined;`
4. `}`
5. `const usuario: Usuario = {`
6.     `id: 1,`
7.     `nome: "Daniel"`
8. `};`

# TypeScript: definição de tipos

## Inferência de tipos do JavaScript

1. `let stringA = "WebAcademy";`
2. `let stringB: string = "WebAcademy";`

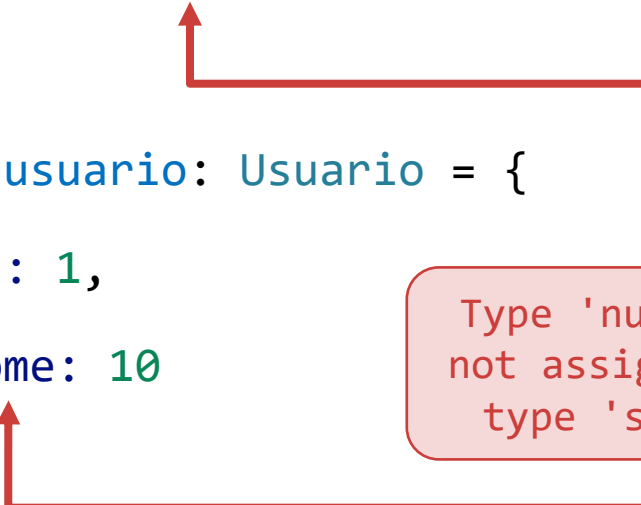
*stringA* e *stringB*  
são do mesmo tipo.



## Definição de tipos

1. `class Usuario {`
2.  `id: number | undefined;`
3.  `nome: string | undefined;`
4. `}`
5. `const usuario: Usuario = {`
6.  `id: 1,`
7.  `nome: 10`
8. `};`

Type 'number' is  
not assignable to  
type 'string'.





# TypeScript: definição de tipos

## Interface

```
1. interface Usuario {  
2.     id: number;  
3.     nome: string;  
4. }
```

## Type

```
1. type Usuario = {  
2.     id: number;  
3.     nome: string;  
4. }
```

- Diferença entre **Type** e **Interface**:
  - <https://www.typescriptlang.org/docs/handbook/2/everyday-types.html#differences-between-type-aliases-and-interfaces>
  - Why use Type and not Interface in TypeScript: <https://youtu.be/ldf0zh9f3qQ>

# Fundamentos

# O que é o Angular?

- **Framework criado pelo Google**, baseado em **TypeScript**.
- Ambiente de desenvolvimento, que inclui:
  - Uma **estrutura baseada em componentes** (facilita escalabilidade).
  - Uma **coleção de bibliotecas** integradas que cobrem uma ampla variedade de recursos, incluindo roteamento, gerenciamento de formulários, comunicação cliente-servidor, etc.
  - Um **conjunto de ferramentas** que ajudam na construção, teste e atualização do código.



# Angular CLI

- Ferramenta de **interface de linha de comando** usada para criar e manter aplicações Angular.
- Principais comandos:
  - **ng new**: cria um espaço de trabalho.
  - **ng build**: compila uma aplicação em um diretório de saída (dist/).
  - **ng serve**: compila a aplicação e cria um servidor, recompilando e atualizando os arquivos sempre que houver modificações.
  - **ng generate**: gera ou modifica arquivos com base em um esquema (componentes, classes, etc.).
- Referência de comandos: <https://v17.angular.io/cli#command-overview>

```
ng new <nome-projeto>  
cd <nome-projeto>  
ng serve
```

# Espaços de trabalho

- Um **espaço de trabalho** (*workspace*) no Angular é o contexto **onde as aplicações são desenvolvidas**, podendo ser uma ou múltiplas aplicações no mesmo espaço de trabalho.
- Criação do espaço de trabalho:

**ng new** <nome\_app> --skip-git --defaults

- Referência: <https://v17.angular.io/cli/new>
- Múltiplos projetos: <https://v17.angular.io/guide/file-structure#multiple-projects>



# Espaços de trabalho

- Um **espaço de trabalho** (*workspace*) no Angular é o contexto **onde as aplicações são desenvolvidas**, podendo ser uma ou múltiplas aplicações no mesmo espaço de trabalho.

- Criação do espaço de trabalho:

`ng new <nome_app> --skip-git --defaults`

Não  
inicializa  
repositório

Ignora prompts  
selecioneando  
opções padrão

- Referência: <https://v17.angular.io/cli/new>
- Múltiplos projetos: <https://v17.angular.io/guide/file-structure#multiple-projects>

# Arquivos de configuração do espaço de trabalho

Arquivo/Diretório	Descrição
angular.json	Padrões de configuração do Angular CLI para todos os projetos no espaço de trabalho, incluindo localização de arquivos CSS e JavaScript.
package.json	Configurações aplicadas aos pacotes npm que estão disponíveis para todos os projetos no espaço de trabalho. Arquivo de configuração de projetos baseados em NodeJS, onde também são configurados os scripts NPM ( <i>start</i> , <i>build</i> , etc.).
src/	Código-fonte do projeto.
node_modules/	Local onde estão armazenados os pacotes npm para todo o espaço de trabalho.
tsconfig.json	Configuração básica do TypeScript para todos os projetos no espaço de trabalho. Todos os outros arquivos de configuração herdam deste arquivo base. A presença do arquivo em um diretório indica que o diretório é a raiz de um projeto TypeScript.

# Arquivos da aplicação (src/)

Arquivo/Diretório	Descrição
app/	Contém a lógica e os dados do projeto, incluindo componentes, templates HTML e arquivos de estilo (CSS).
assets/	Contém arquivos de imagem e outros conteúdos estáticos (CSS, JavaScript, etc.).
environments/	Contém opções de configuração de compilação para ambientes específicos. Normalmente, há um ambiente de desenvolvimento e um ambiente de produção.
index.html	A página HTML principal exibida quando a aplicação é acessada. Arquivos JavaScript e CSS são adicionados automaticamente no index.html no processo de <i>build</i> .
main.ts	Ponto de entrada da aplicação. Inicializa o componente principal (AppComponent) para ser executado no navegador.
styles.css	Reúne configurações CSS globais do projeto.

# Arquivos da aplicação (src/app/)

Arquivo/Diretório	Descrição
app.component.ts	Contém a lógica do componente principal da aplicação (AppComponent).
app.component.html	Define o documento HTML (template) associado ao AppComponent.
app.component.css	Define as propriedades estilo CSS para o AppComponent.
app.component.spec.ts	Arquivo criado para realizar teste de unidade do AppComponent.
app.routes.ts	Define as configurações de roteamento para a aplicação.
app.config.ts	Gerencia configurações da aplicação, incluindo o fornecimento das rotas definidas no arquivo <i>app.routes.ts</i> .

# Executando o projeto

- Iniciar a aplicação:
  - **ng serve**
- Sempre que um arquivo da aplicação for alterado, o projeto é recompilado e a modificação é refletida no cliente.

```
Initial chunk files | Names | Raw size
polyfills.js      | polyfills | 86.27 kB |
main.js           | main      | 8.40 kB  |
styles.css        | styles    | 4.48 kB  |
scripts.js        | scripts   | 648 bytes |
                  | Initial total | 99.78 kB

Application bundle generation complete. [2.233 seconds]

Watch mode enabled. Watching for file changes...
→ Local: http://localhost:4200/
→ press h + enter to show help
```



# Deploy

- O servidor de aplicação do Angular ([webpack](#)) não é recomendado para o modo de produção.
- Para um *deployment* simples são suficientes os seguintes passos:
  1. Compilar o projeto: **ng build**
  2. Copiar todo o conteúdo do diretório **dist/<nome\_app>/browser** para o servidor.
  3. **(Opcional, mas importante)** Configurar o servidor para **redirecionar solicitações de arquivos ausentes para *index.html***. (<https://v17.angular.io/guide/deployment#routed-apps-must-fall-back-to-indexhtml>)
- Opções de *deployment* automático:
  - <https://v17.angular.io/guide/deployment#automatic-deployment-with-the-cli>

**Continua...**

# Referências

- DUCKETT, Jon. **Javascript e JQuery: desenvolvimento de interfaces web interativas**. 1. ed. [S. l.]: Alta Books, 2016. 640 p.
- GOOGLE (ed.). **Angular Docs**. [S. l.], 2024. Disponível em: <https://v17.angular.io/docs>.
- MARCO TULIO VALENTE. **Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade**, 2020. Disponível em: <https://engsoftmoderna.info/>
- MICROSOFT (ed.). **The TypeScript Handbook**. [S. l.], 2024. Disponível em: <https://www.typescriptlang.org/docs/handbook/intro.html>.
- MOZILLA (ed.). **MDN Web Docs: Aprendendo desenvolvimento web**. [S. l.], 2024. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn>.