



Angular 9

Aula #5

Vídeo Aula e Exercícios

Esta apostila é referente a vídeo aula #5 e a sequência de exercícios #5 do curso Angular 9. Não deixe de assistir a vídeo aula e fazer os exercícios propostos para esta aula em nossa Plataforma Online.

Sobre os materiais da Grande Porte

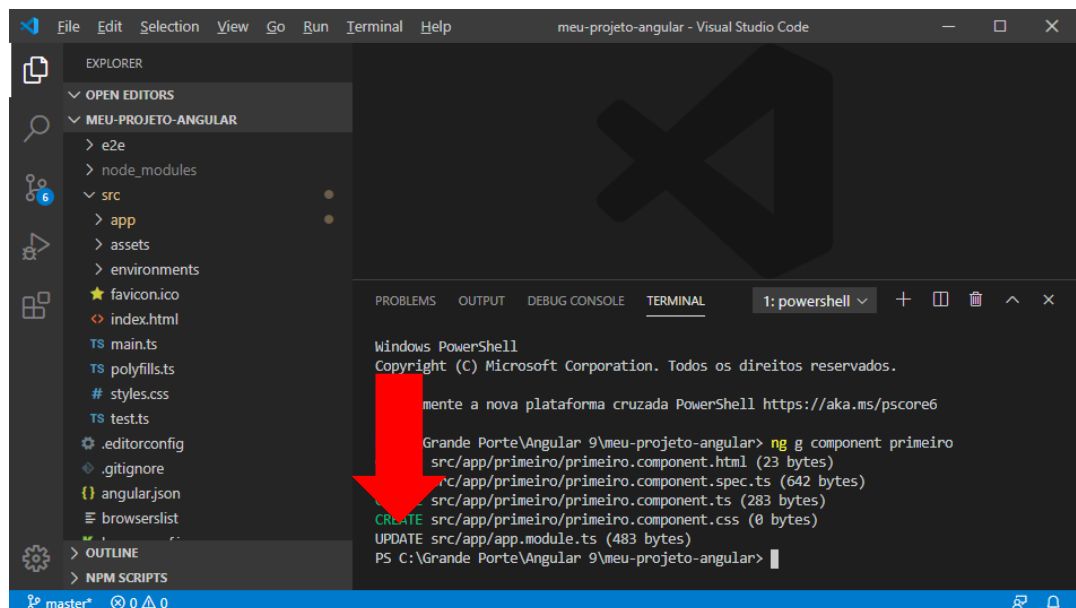
Todos os materiais desenvolvidos pela Grande Porte objetivam ensinar de maneira eficiente, explorando potenciais e sanando dificuldades. As atualizações são feitas constantemente para manter o conteúdo atualizado com as demandas de mercado e tendências para o futuro. Esperamos que você aproveite este material. Comentários, críticas e sugestões serão muito bem-vindos.

ATENÇÃO

Os materiais desenvolvidos pela Grande Porte são distribuídos através da Plataforma Online apenas para assinantes. Todos os direitos são reservados à Grande Porte. A distribuição, cópia, revenda e utilização para ministrar treinamentos, sem autorização, são absolutamente proibidas. Se você deseja obter autorização para usar nossos materiais comercialmente, por favor, entre em contato.

Chamando um componente

Após a criação de nosso primeiro componente, podemos observar que o arquivo **app.module.ts** foi atualizado, conforme figura 1.



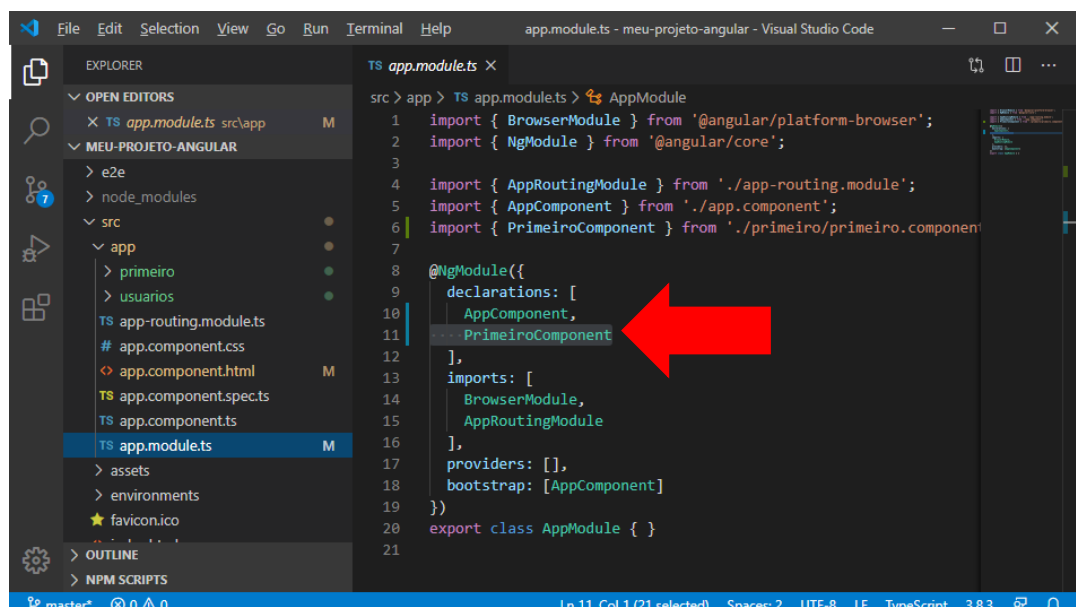
```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Atualmente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

Grande Porte\Angular 9\meu-projeto-angular> ng g component primeiro
src/app/primeiro/primeiro.component.html (23 bytes)
src/app/primeiro/primeiro.component.spec.ts (642 bytes)
src/app/primeiro/primeiro.component.ts (283 bytes)
CREATE src/app/primeiro/primeiro.component.css (0 bytes)
UPDATE src/app/app.module.ts (483 bytes)
PS C:\Grande Porte\Angular 9\meu-projeto-angular>
```

Figura 1. Console do VS Code mostrando a atualização do **app.module.ts**

Isso significa que, por default, todo novo componente é criado dentro do módulo **app.module**. Isso pode ser comprovado abrindo esse arquivo no VS Code, conforme figura 2.



```
src > app > TS app.module.ts x AppModule
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppRoutingModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6 import { PrimeiroComponent } from './primeiro/primeiro.component';
7
8 @NgModule({
9   declarations: [
10     AppComponent,
11     PrimeiroComponent
12   ],
13   imports: [
14     BrowserModule,
15     AppRoutingModule
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
19 })
20 export class AppModule { }
21
```

Figura 2. O arquivo **app.module.ts** com a declaração do **PrimeiroComponent**.

Então o componente **primeiro** pertence ao módulo **app**, juntamente com o componente **app**, pois em **declarations** temos **AppComponent** e **PrimeiroComponent**.

Ao abrir o **app.component.ts** no VS Code, observamos que neste componente temos o par nome/valor: **selector: 'app-root'**, conforme figura 3.

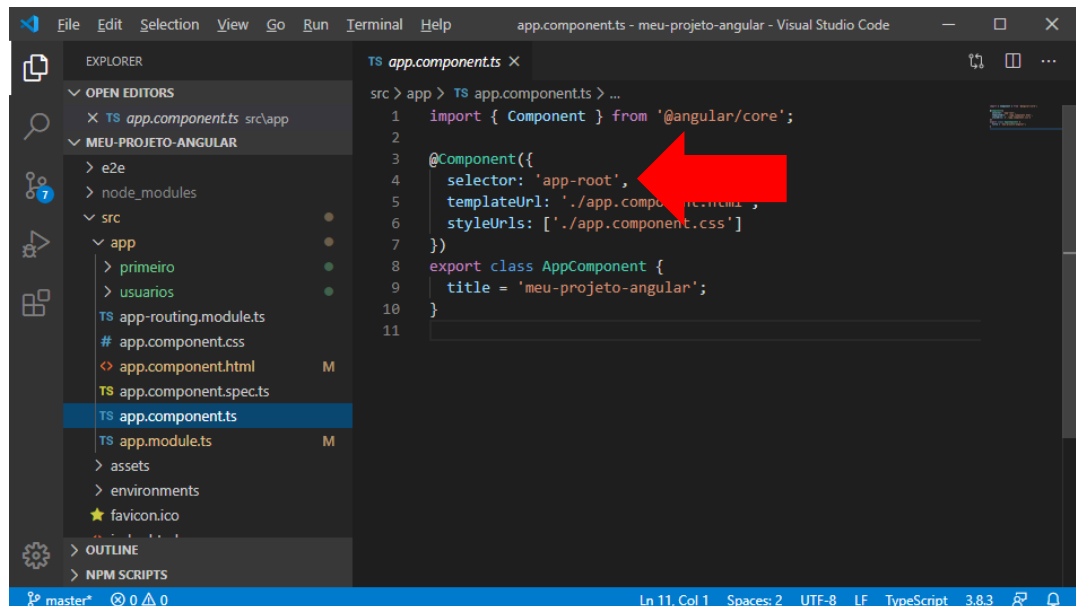


Figura 3. O seletor do componente **app**.

Abrindo também o **primeiro.component.ts** no VS Code, podemos ver que o par nome/valor é: **selector: 'app-primeiro'**, conforme figura 4.

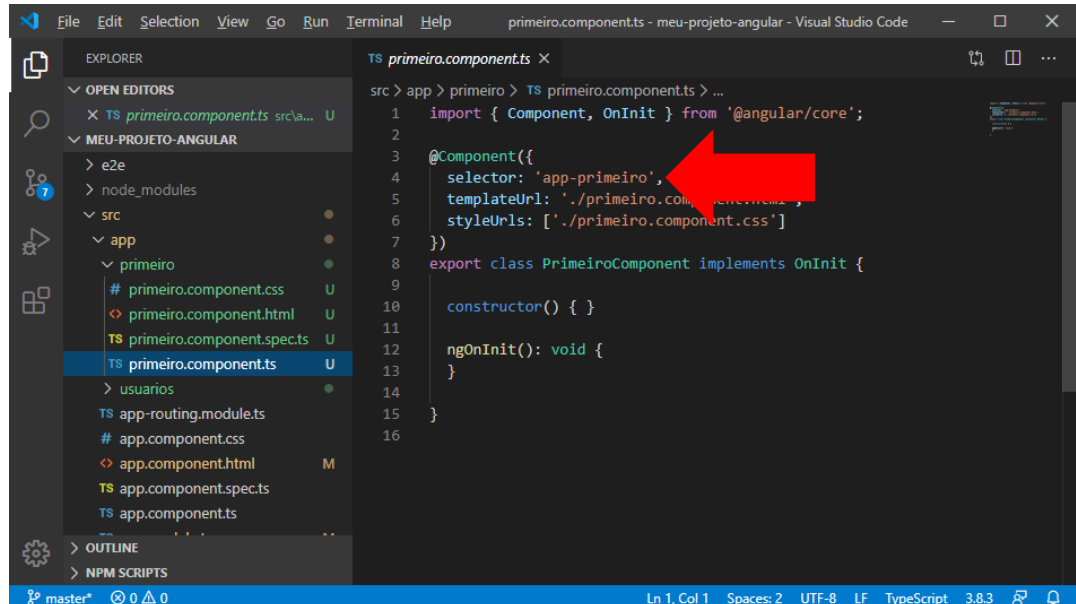


Figura 4. O seletor do componente **primeiro**.

É através desses nomes declarados no **selector** que os componentes podem ser chamados no arquivo **.html**.

Vamos editar o arquivo **app.component.html** que é o arquivo html do módulo raiz. **Exclua** todo o conteúdo deste arquivo e coloque o código abaixo:

```
<app-primeiro></app-primeiro>
```

Isso irá chamar o componente primeiro. A digitação deste código no VS Code pode ser vista na figura 5.

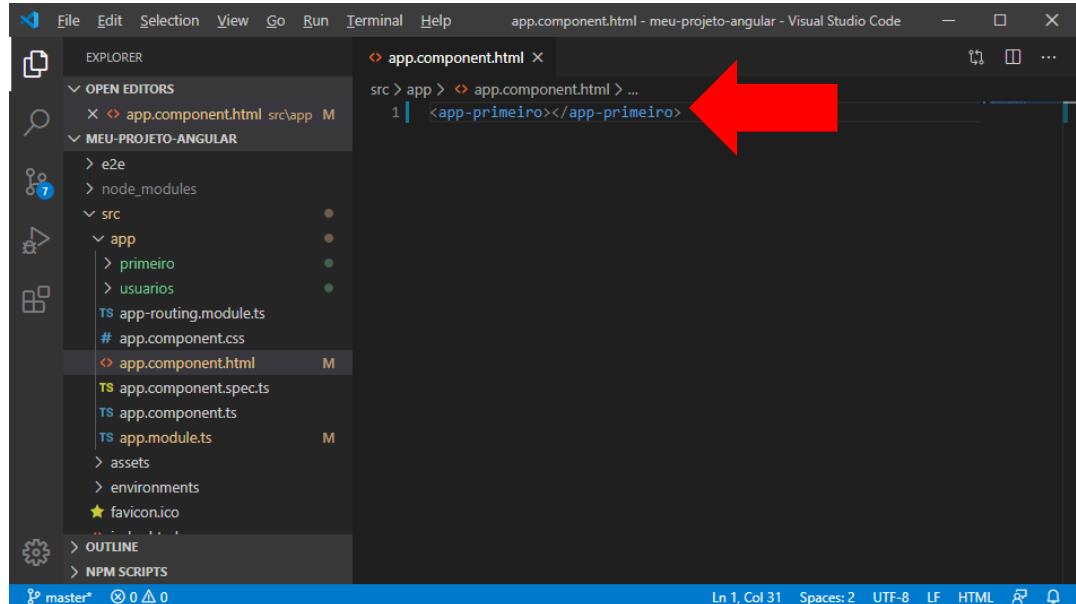


Figura 5. O html do componente raiz chamando o componente **primeiro**.

Ao salvar esse arquivo, o servidor fará as compilações necessárias e o browser refletirá esse resultado, conforme figura 6.

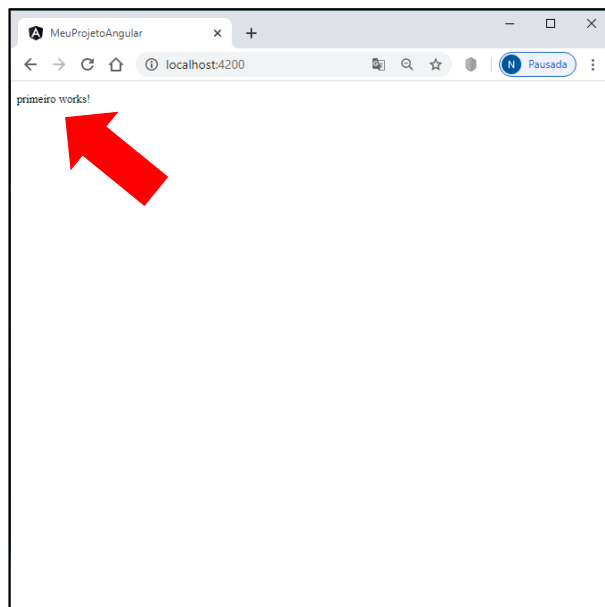


Figura 6. O html do componente **primeiro** exibido pelo browser.

Esse texto foi colocado como default no momento da criação do componente **primeiro** no arquivo **primeiro.component.html**, conforme figura 7.

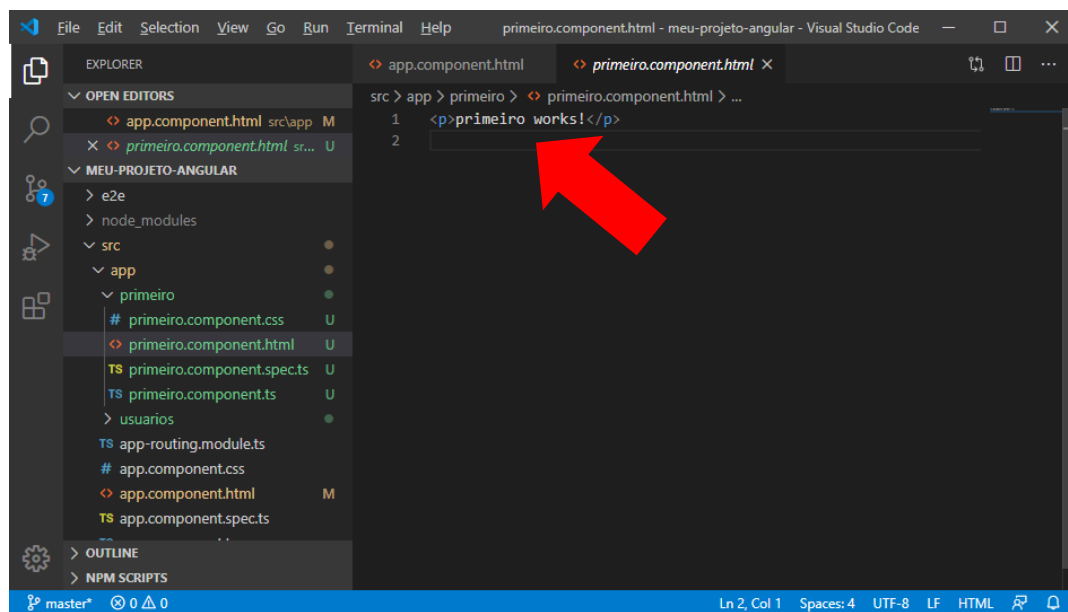


Figura 7. O código html do componente **primeiro**.

Data Binding (Vinculação)

O Data Binding ou Vinculação de dados é o conceito principal do Angular 9 e é usado para definir a comunicação entre os arquivos typescript e html do componente. Isso facilita o desenvolvimento de aplicativos interativos sem se preocupar em enviar e receber dados.

Aqui vale uma explicação sobre DOM (*Document Object Model*). Um documento HTML tem o que é chamado de uma estrutura hierárquica. Cada elemento (ou tag), exceto a tag <html> superior, está contido em outro elemento, seu **pai**. Este elemento, por sua vez, pode conter elementos **filho**. Você pode visualizar isso como uma espécie de árvore genealógica, conforme figura 8.

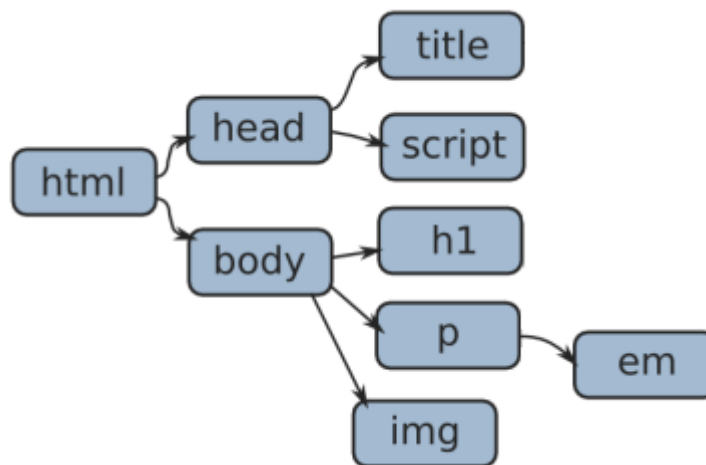


Figura 8. O código html e sua estrutura hierárquica.

O DOM define a estrutura lógica do html e como ele é acessado e manipulado. É muito comum utilizar a sigla DOM ao se referir a um arquivo html.

O Data Binding pode ser unidirecional ou bidirecional.

O **Data Binding unidirecional** é uma comunicação simples entre o arquivo typescript e o arquivo html. Quando essa comunicação tem origem no arquivo typescript e destino no arquivo html, o Data Bind pode ser de dois tipos:

- **Interpolation**
- **Property Binding**

Quando essa comunicação tem origem no arquivo html e destino no arquivo typescript, o Data Binding é do tipo:

- **Event Binding**

No **Data Binding bidirecional**, a sincronização automática de dados ocorre entre os arquivos typescript e html do componente. Sempre que você fizer alterações no typescript, ele será refletido no html e quando você fizer alterações no html ele será refletido no typescript. Esse tipo de Data Binding é do tipo:

- **Two-way binding.**

Esses quatro tipos de Data Binding são ilustrados na figura 9.

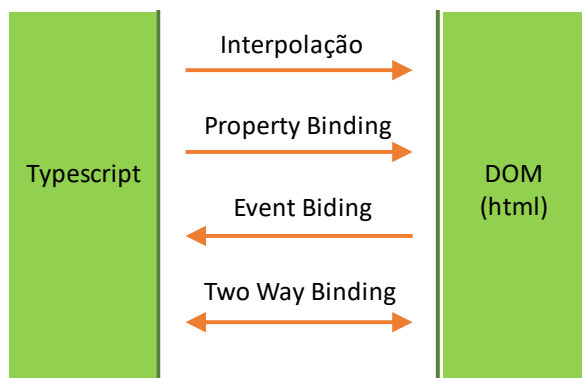


Figura 9. Os quatro tipos de Data Binding

Nesta aula veremos a Interpolação e o Property Binding, em que a origem dos dados é o arquivo typescript e o destino é o arquivo html.

Interpolação

A interpolação, também chamada de interpolação de string, é uma técnica de Data Binding unidirecional que é usada para exibir dados cuja origem é o arquivo typescript e o destino é o arquivo html, conforme figura 10.

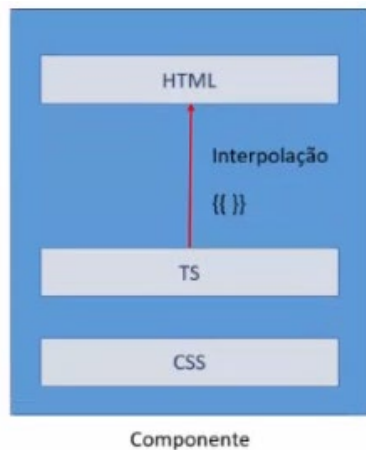


Figura 10. A interpolação – Data Binding unidirecional do TS para o HTML

A interpolação começa com a criação do dado no arquivo typescript **primeiro.component.ts**. Como exemplo, vamos criar a variável **nome** e atribuir a ela o valor “Grande Porte”. Isso é feito dentro da classe **PrimeiroComponent** antes da chamada do construtor, conforme figura 11.

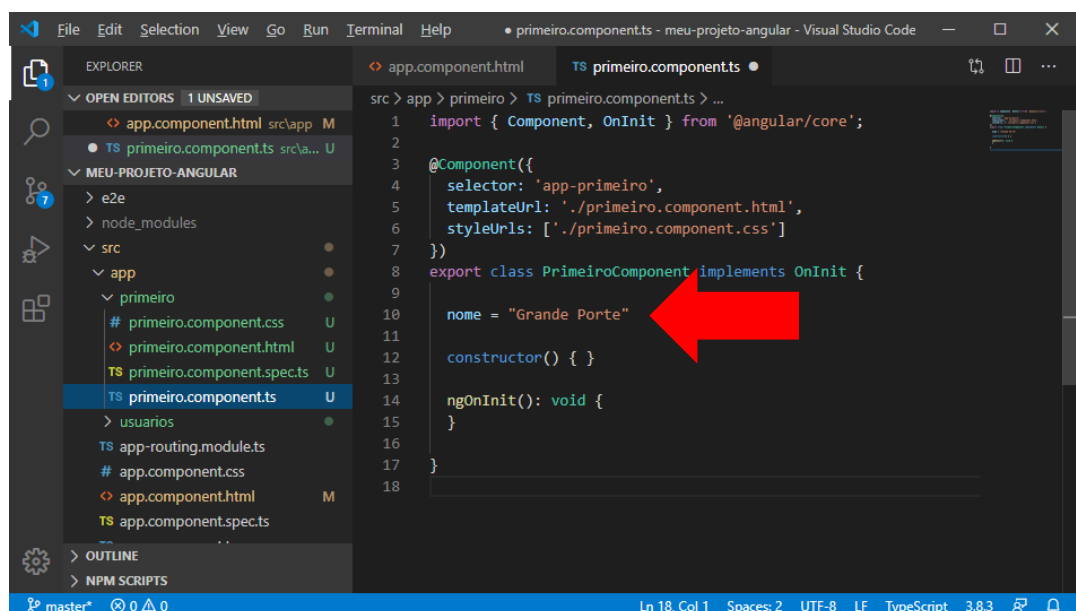


Figura 11. A interpolação – Criação da variável **nome** no arquivo TS

O próximo passo é exibir o conteúdo dessa variável no arquivo html. Isso é feito utilizando o nome da variável fechada entre dois pares de chaves:

```
{{ variável }}
```

Vamos editar o arquivo **primeiro.component.html**, conforme figura 12.

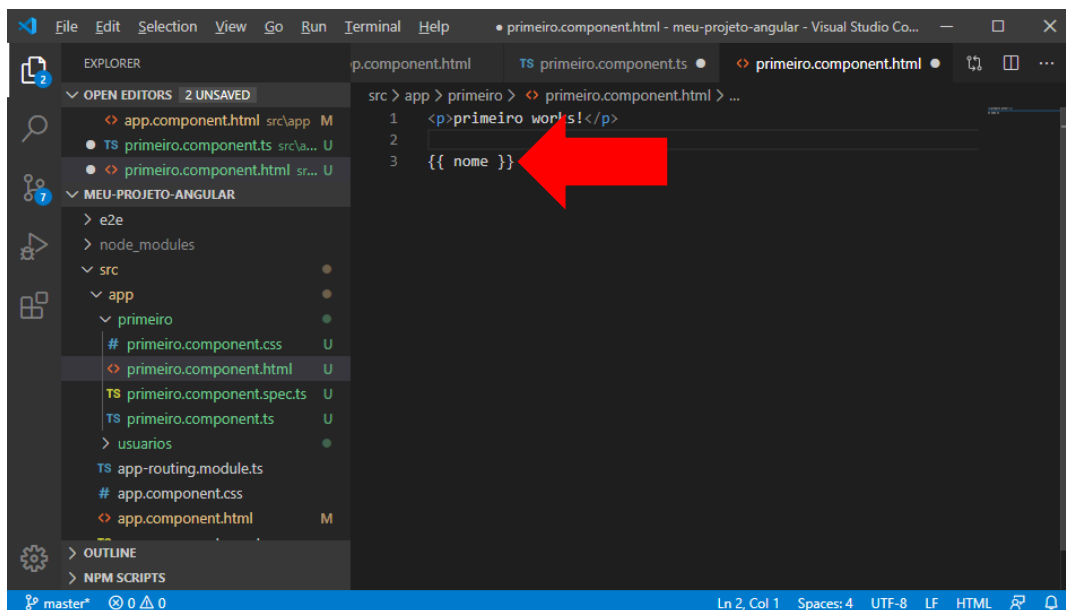


Figura 12. A interpolação – Chamada da variável nome no arquivo HTML

Após salvarmos essa alteração, o NodeJS realizará a compilação e o resultado refletirá imediatamente no browser, conforme figura 13.

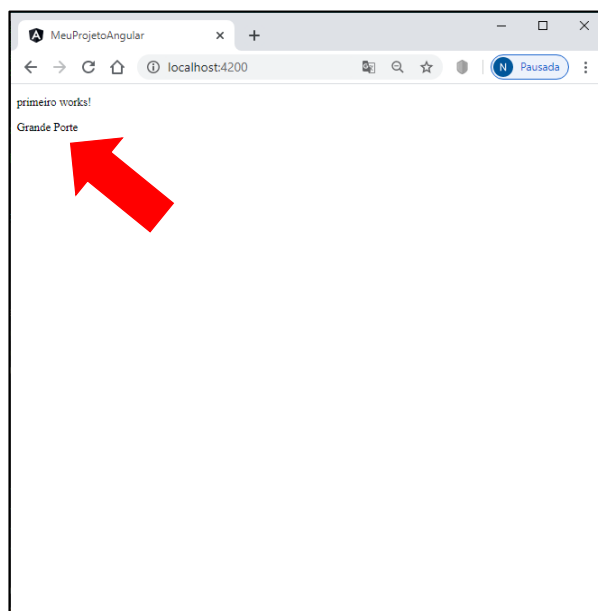


Figura 13. A interpolação – Browser Google Chrome exibindo a variável nome

Property Binding

o Property Binding também é uma técnica de Data Binding unidirecional, onde ocorre a vinculação da propriedade de um elemento DOM (tag html) com uma variável definida no arquivo typescript, conforme figura 14.

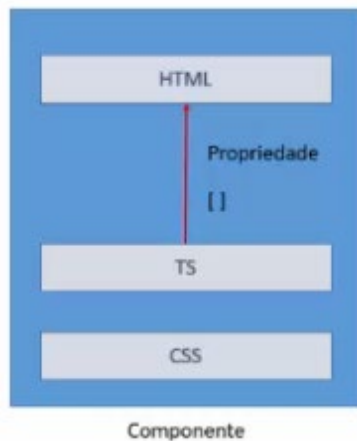
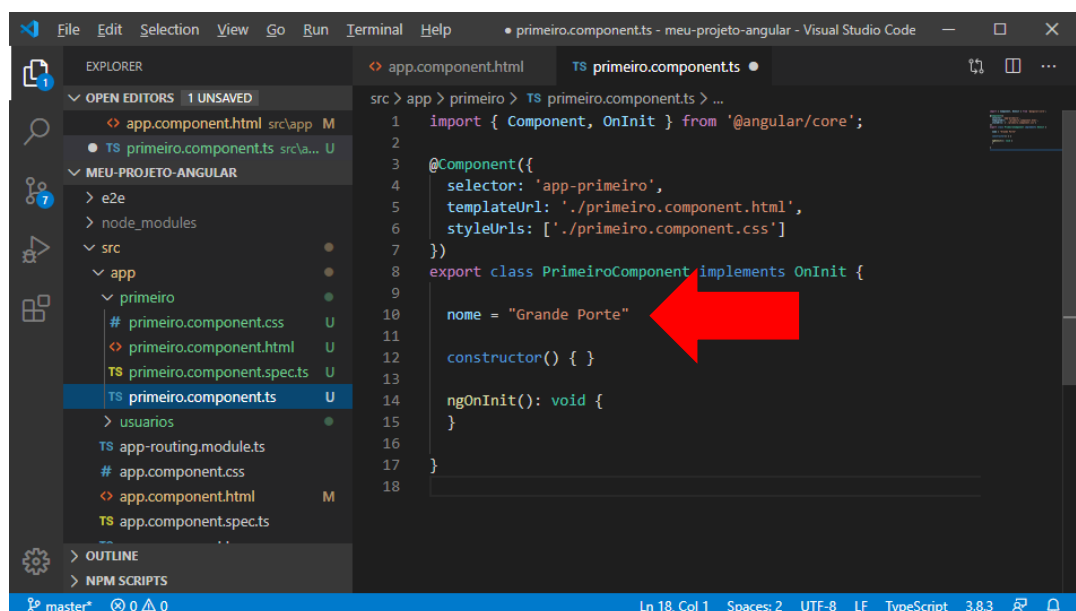


Figura 14. O Property Binding – Data Binding unidirecional do TS para o HTML

A Property Binding começa com a criação do dado no arquivo typescript **primeiro.component.ts**. No tópico anterior (interpolação) nós já criamos a variável **nome** e atribuímos a ela o valor “Grande Porte”. Isso foi feito dentro da classe **PrimeiroComponent** antes da chamada do construtor, conforme figura 15.



A imagem mostra a interface do Visual Studio Code com o arquivo **primeiro.component.ts** aberto. O código TypeScript define uma classe **PrimeiroComponent** que implementa **OnInit**. A linha 10 contém a declaração da variável **nome = "Grande Porte"**, que é destacada por uma seta vermelha. O código também inclui a configuração do componente com **selector**, **templateUrl** e **styleUrls**.

```

1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-primeiro',
5   templateUrl: './primeiro.component.html',
6   styleUrls: ['./primeiro.component.css']
7 })
8 export class PrimeiroComponent implements OnInit {
9
10  nome = "Grande Porte"
11
12  constructor() { }
13
14  ngOnInit(): void {
15  }
16
17 }
18

```

Figura 15. O Property Binding – Criação da variável **nome** no arquivo TS

O próximo passo é editar uma tag html, por exemplo **<p>** (parágrafo) e atribuir a uma propriedade desta tag uma variável. Vamos utilizar a propriedade **innerHTML** da

tag <p>. A forma de usar o Property Binding é escrever a propriedade fechada entre colchetes e atribuir uma variável entre aspas, conforme abaixo:

```
<tag [propriedade]='variável'></tag>
```

Vamos editar o arquivo **primeiro.component.html**, conforme figura 16.

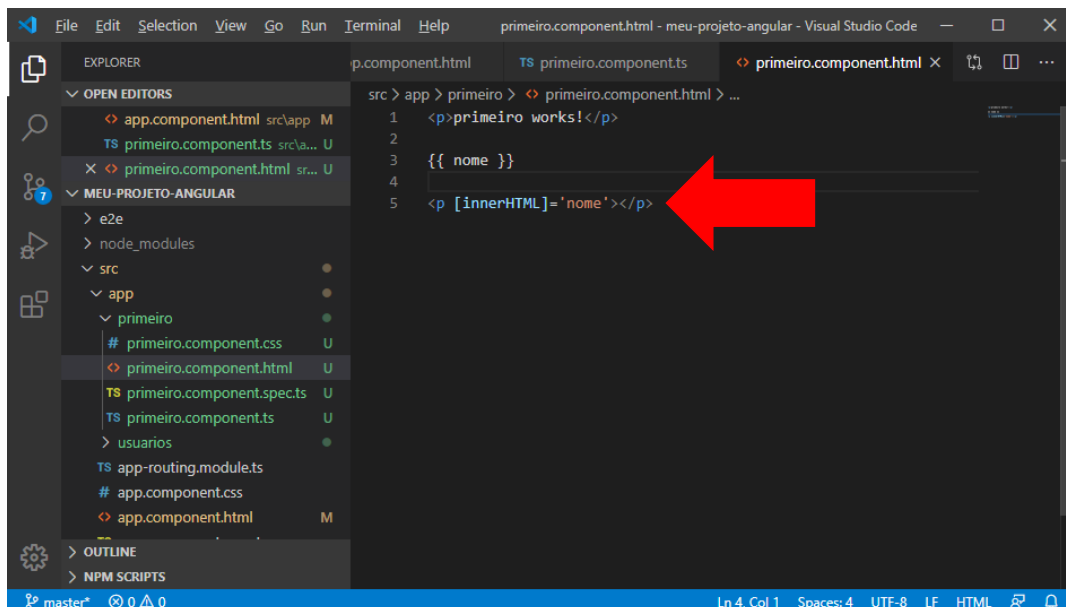


Figura 16. O Property Binding – Chamada da variável nome na propriedade da tag HTML

Após salvarmos essa alteração, o NodeJS realizará a compilação e o resultado refletirá imediatamente no browser, conforme figura 17.

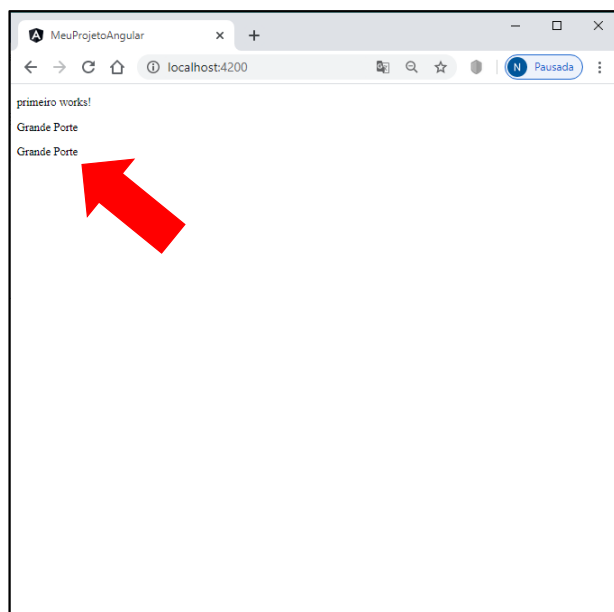


Figura 17. O Property Binding – Browser Google Chrome exibindo a variável nome

Vídeo Aula e Exercícios

Esta apostila é referente a vídeo aula #5 e a sequência de exercícios #5 do curso Angular 9. Não deixe de assistir a vídeo aula e fazer os exercícios propostos para esta aula em nossa Plataforma Online.

Sobre o autor

Fabrizio Borelli



Doutorando em Engenharia da Informação pela UFABC

Mestre em Ciência da Computação pela UFABC

Bacharel em Ciência da Computação pela UFABC

Professor Universitário

Empreendedor em TI