PROGRAMAÇÃO WEBII

1

Typescript

Criando variáveis

Criação de variáveis primitivas

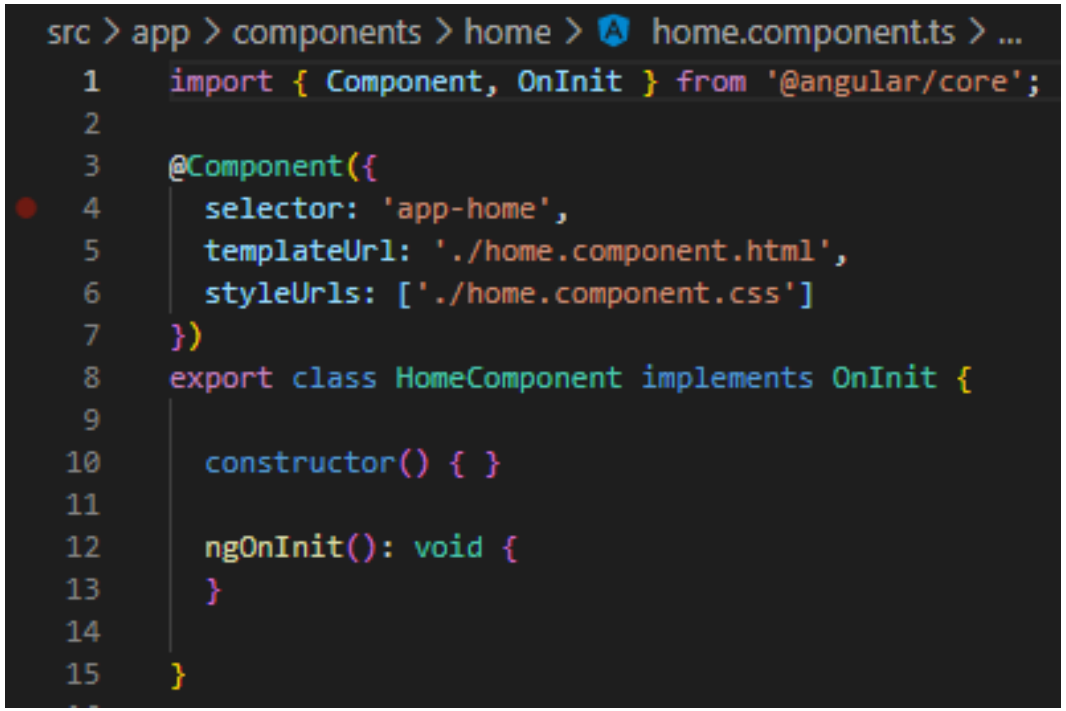
- O TypeScript possui 3 tipos primitivos:
  - string -> para textos
  - number -> para números inteiros ou ponto flutuante
  - boolean -> para T ou F, verdadeiro ou falso

Na prática

- Vamos começar...
  - Vamos navegar até a pasta do nosso projeto
    - Acessar prompt de comando
    - cd\angular\cursoPWEBII\frontend <ENTER>
    - code . <ENTER>
    - ng serve –o <ENTER> (Como não iremos alterar nada na infraestrutura, já podemos inicializar.

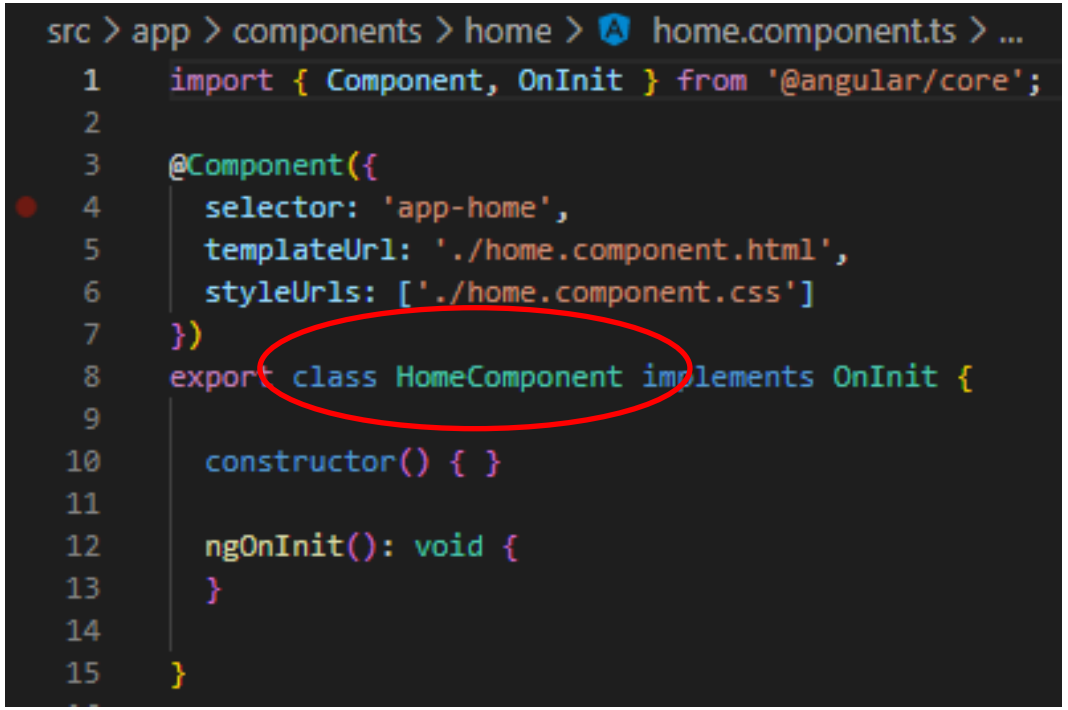
Na prática

- Vamos trabalhar no arquivo:
  - src
  - app
  - components
  - home
  - home.component.ts



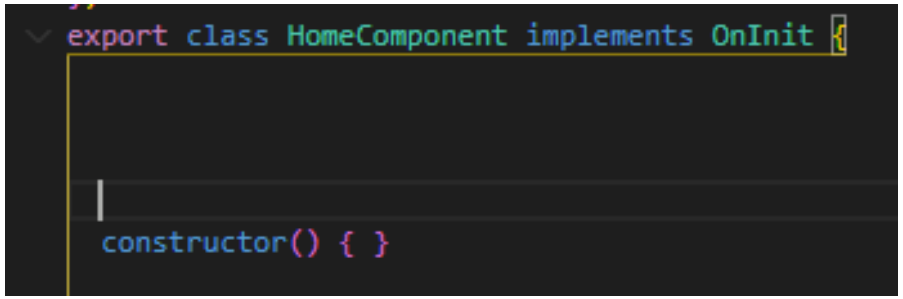
Na prática

- Estamos trabalhando com orientação a objetos e estamos dentro de uma classe.



Na prática

- Quem já vem de Java ou C# está acostumado a trabalhar com orientação a objeto e quem não vem e está começando agora verá que é super tranquilo trabalhar principalmente dentro do Angular e TypeScript.
- As variáveis devem ser criadas antes do construtor. Vamos criar algumas linhas em branco e começar.



Na prática

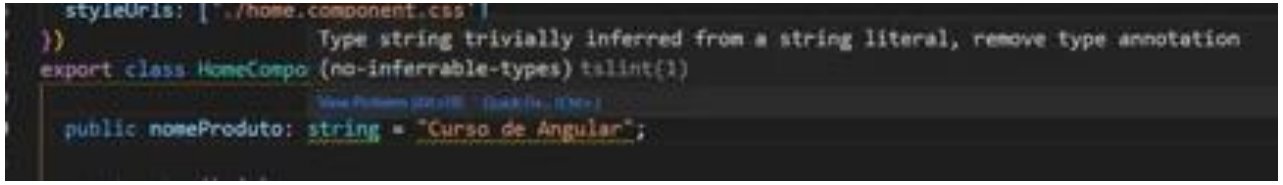
Vamos começar escrevendo public, o nome da variável. Se estivesse em Java ou C#, aqui seria o tipo da variável, mas como estamos dentro do TypeScript /Javascript, vamos utilizar direto o nome da variável, :, tipo da variável e o conteúdo.

public nomeProduto: string = “Curso de Angular”;

```
 8   export class HomeComponent implements OnInit {
 9
10     public nomeProduto: string = "Curso de Angular";
11
```

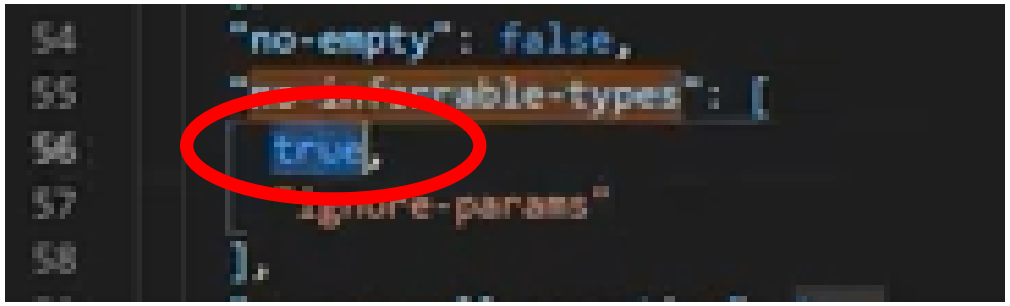
Na prática

- O preenchimento está correto, mas pode ser que vc esteja diante de 2 warnings, 2 avisos.
- O primeiro deles é no string. Nós estamos trabalhando com uma extensão chamada tslint e essa extensão nos ajuda a colocar em prática as boas práticas da programação. Uma delas é que o typeScript diz que se criarmos uma variável e preenchermos o conteúdo dela no momento da criação, não precisamos colocar o tipo da variável, porque ele já consegue identificar o conteúdo da variável e já vai saber o tipo da variável. Mas, é uma boa prática para quem está começando com TypeScript deixar o tipo da variável aqui para que você não se perca durante a programação.
- Para resolver isso:
  - Coloque o cursor em cima do tipo de variável (string)



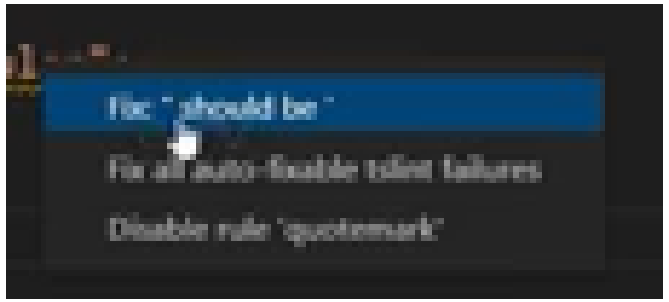
Na prática

- Copie o erro: no-inferrable-types <CTRL><C>
- Vamos na raiz do projeto, dentro do arquivo tslint.Jason.
- <CTRL><F> para procurar
- <CTRL><V> para colar
- A propriedade deve estar como true, só substituir por false
- Salve o arquivo e quando retornar para o arquivo home.component.ts o primeiro erro de warning não estará mais aparecendo.



Na prática

- O segundo erro estará informando que utilizamos aspas, mas é convenção dentro do TypeScript utilizar apóstrofe. Se deixar como aspas, irá ter algum problema, irá aparecer algum erro? Não, mas é de convenção que utilizemos apostrofe.
- Como corrigir?
  - Só posicionando o mouse em cima, já irá aparecer mensagem quick fix. Clique n quick fix
  - Depois irá exibir outra janela e clique na 1ª opção: fix “ should be ‘



Na prática

- Outra convenção que iremos utilizar no curso do Angular, porque é do Angular, iremos retirar a palavra public. Toda vez que uma variável ter seu tipo de acesso como public, não precisamos escrever public aqui. Se fosse do tipo private teríamos que escrever o private.

```
8  export class HomeComponent implements OnInit {
9
10   nomeProduto: string = 'Curso de Angular';
11
```

Na prática

- Outra coisa legal para trabalhar com string dentro do TypeScript é a possibilidade de trabalhar com template strings, o que seria isso? Podemos colocar variáveis dentro do texto da nossa string. Vamos ver como isso acontece.

Na prática

- Vamos criar uma variável chamada anuncio com tipo string = Ao invés de colocar aspas ou apóstrofe iremos colocar crase. Dentro da crase, podemos criar um texto onde pode conter uma variável no meio do texto. Ao invés e utilizar concatenação é possível colocar direto a variável no meio do texto.
- Vamos escrever a mensagem: O cifrão, chaves. Dentro da chaves, iremos inserir o nome da variável: this.nomeProduto. Após o fechamento da chaves iremos continuar com o texto: está em promoção.
- O $ {}
- ´O${this.nomeProduto} está em promoção´

Na prática

- Criamos a variável string, utilizando o template string.

```
8  export class HomeComponent implements OnInit {
9      nomeProduto: string = 'Curso de Angular';
10     anuncio: string = `O ${this.nomeProduto} está em promoção`;
```

Conceito Template Literals Interpolação

- Tanto no JavaScript como no TypeScript, as strings definidas com *backticks*/crases (`) permitem interpolação (embutir expressões no meio da string sem precisar fechar sua delimitação e concatená-la com outra string), multilinhas (não precisar escapar uma quebra de linha dentro da string ou concatenar duas strings em linhas diferentes) e processamento (*tagging/parse*, onde você pode usar uma função pré-definida).
- Esta forma de definição é nativa do JavaScript e é chamada de *template literals* a partir do ES2015 e de *template strings* anteriormente.

https://medium.com/collabcode/template-string-%C3%A9-top-demais-02-5d8964726a32

(https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Tem

Template Literals

- Exemplos de uso de *template literals*:

```
`texto` # simples 'string text'

`texto linha 1
texto linha 2` # multilinha equivalente a 'texto linha 1\ntexto linha 2'

`texto ${variavel} texto` # interpolação

function tag(literais, ...expressoes) {
    let resultado = "";

    for (let i = 0; i < expressoes.length; i++) {
        resultado += literais[i];
        resultado += expressoes[i].toUpperCase();
    }
    resultado += literais[literais.length - 1];

    return resultado;
}

tag `texto ${variavel} texto` # parse
```

Na prática

- E como seria com concatenação?
- Vamos utilizar o construtor para ver como ficaria:

```
9     nomeProduto: string = 'Curso de Angular';
10    anuncio: string = `O ${this.nomeProduto} está em promoção`;
11
12    constructor() {
13      // variáveis de string com concatenação
14      this.anuncio = 'O' + this.nomeProduto + 'está em promoção';
15
16    }
```

Na prática

- Com o template string fica muito mais fácil entender o que está acontecendo dentro da nossa variável, como ficará nosso texto final.
- Sempre que for possível utilizaremos template string.
- Vamos comentar o uso de concatenação no nosso código:

```
 9    nomeProduto: string = 'Curso de Angular';
10    anuncio: string = `O ${this.nomeProduto} está em promoção`;
11
12    constructor() {
13      // variáveis de string com concatenação
14     // this.anuncio = 'O' + this.nomeproduto + 'está em promoção';
15
16    }
```

Na prática

- O próximo tipo de variável será as variáveis tipo number:

idProduto: number = 123;

- O próximo tipo de variável será number com ponto flutuante:

precoProduto: number = 2.59;

Não importa se será com ponto flutuante ou não, sempre será number. Podemos inicialmente informar que o conteúdo é sem ponto flutuante e depois alterar, desde que definimos como number.

```
9     nomeProduto: string = 'Curso de Angular';
10    anuncio: string = `O ${this.nomeProduto} está em promoção`;
11    idProduto: number = 123;
12    precoProduto: number = 2.59;
```

Na prática

- O último tipo primitivo que iremos verificar é o tipo boolean.

promocao: boolean = true;

```
 9    nomeProduto: string = 'Curso de Angular';
10    anuncio: string = `O ${this.nomeProduto} está em promoção`;
11    idProduto: number = 123;
12    precoProduto: number = 2.59;
13    promocao: boolean = true;
```

Na prática

- Criamos nossas variáveis e vamos ver se elas estão funcionando...
- Dentro do construtor, vamos chamar o console.log. Quem trabalha com o javascript está acostumado a utilizar o console. Vamos digitar:

console.log('Nome do Produto', this.nomeProduto);

Salvar o arquivo

```
13  constructor() {
14    // variáveis de string com concatenação
15    // this.anuncio = 'O' + this.nomeProduto + 'está em promoção';
16
17    console.log('Nome do Produto: ', this.nomeProduto);
18
19  }
```

Na prática

- Se o servidor não estiver rodando, deixar rodando...
- Abrir a página da aplicação: localhost:4200
- Teclar <F12>
- Vocês irão ver dentro do console o nome do produto: Curso de Angular.

Na prática

- Por que estamos utilizando , ao invés do sinal de +?
  - A virgula (,) permite que possamos passar vários argumentos para ele. Cada argumento que colocarmos será exibido na tela.
  - Nesse caso, se trocarmos por + teremos o mesmo resultado da , (virgula).
  - Porém se a variável (this.nomeProduto) fosse do tipo objeto ou um array, ao invés de imprimir o conteúdo desse objeto ou desse array, iria imprimir apenas o tipo da variável. Sendo assim, sempre iremos trabalhar dentro do console com a , (virgula). Ele imprime a string e o conteúdo da variável.

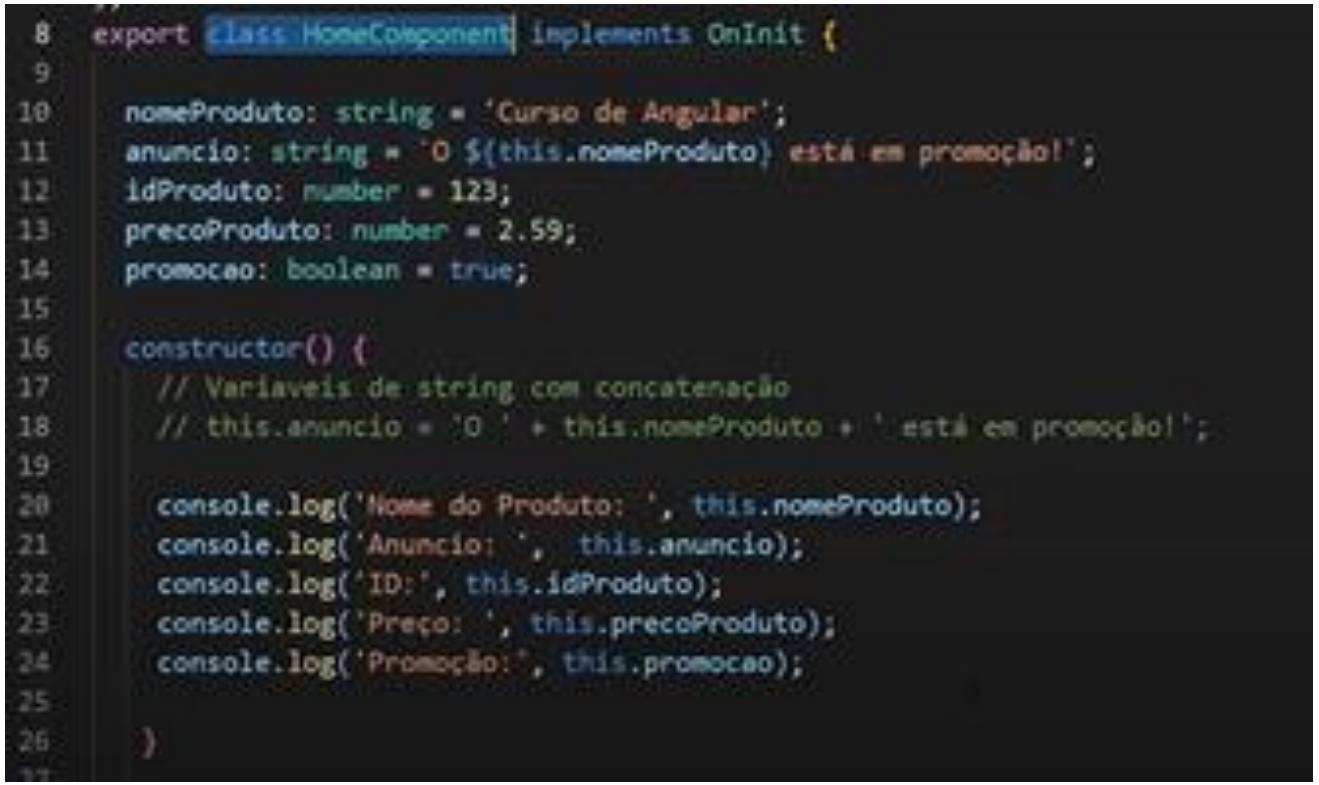
Na prática

- Vamos fazer para as outras variáveis também.

```
20      console.log('Nome do Produto: ', this.nomeProduto);
21      console.log('Anuncio: ',  this.anuncio);
22      console.log('ID:', this.idProduto);
23      console.log('Preço: ', this.precoProduto);
24      console.log('Promoção:', this.promocao);
25
```

Na prática

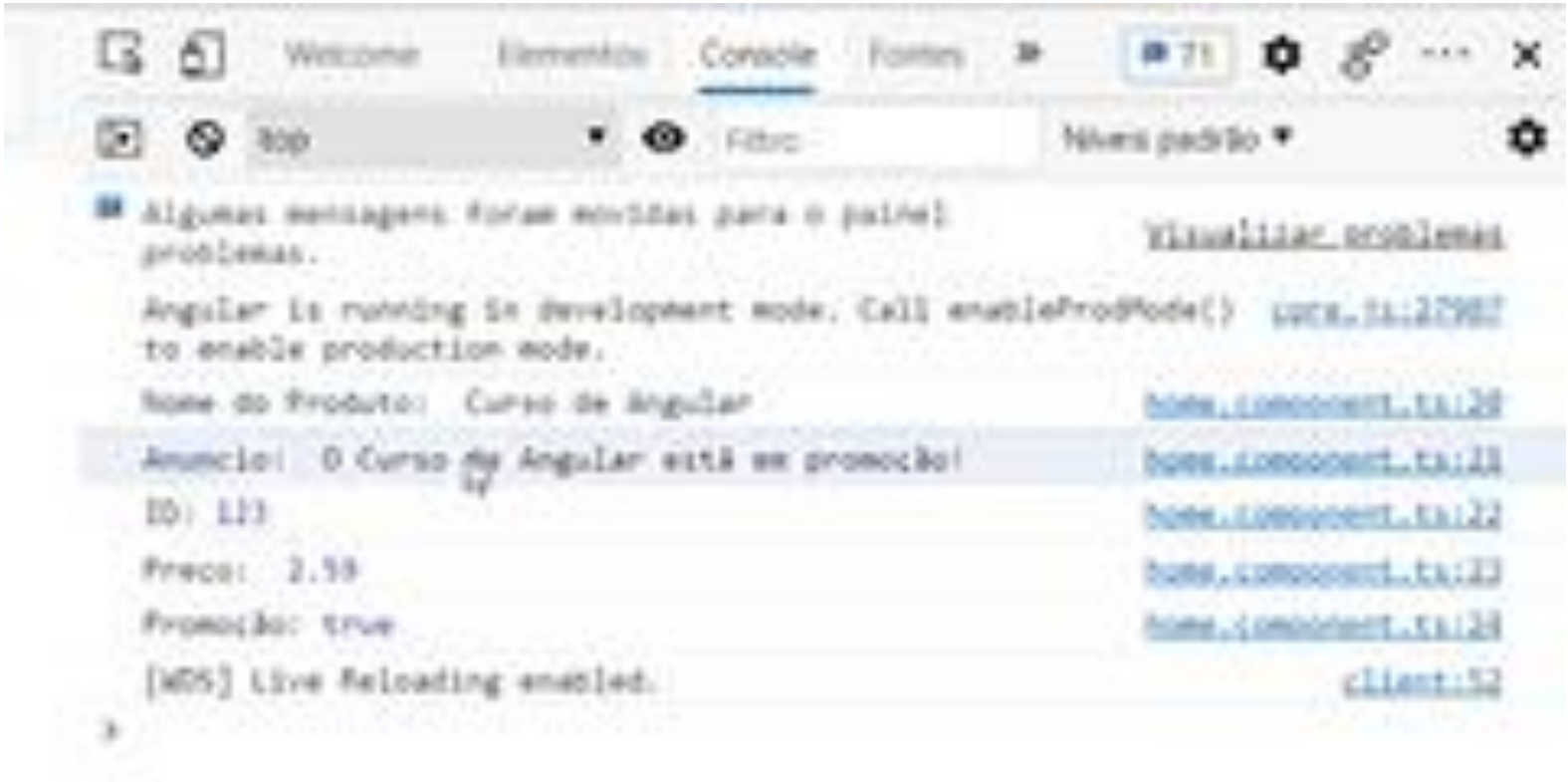
- Estamos com todas as variáveis criadas. Elas são globais e podem ser utilizadas em qualquer lugar dentro da classe home.component. Basta referenciá-las utilizando o this.



```
8  export class HomeComponent implements OnInit {
9
10   nomeProduto: string = 'Curso de Angular';
11   anuncio: string = `O ${this.nomeProduto} está em promoção!`;
12   idProduto: number = 123;
13   precoProduto: number = 2.59;
14   promocao: boolean = true;
15
16   constructor() {
17     // Variaveis de string com concatenação
18     // this.anuncio = 'O ' + this.nomeProduto + ' está em promoção!';
19
20     console.log('Nome do Produto: ', this.nomeProduto);
21     console.log('Anuncio: ', this.anuncio);
22     console.log('ID:', this.idProduto);
23     console.log('Preço: ', this.precoProduto);
24     console.log('Promoção:', this.promocao);
25
26   }
```

Na prática

- Vamos retornar na página, já recarregou...



Na prática

- Vamos falar um pouco sobre o escopo das variáveis dentro do código.
- Vocês devem ter visto em outros vídeos, variáveis carregadas com var, let ou const:
- Esses 3 tipos de variáveis nós utilizaremos dentro das nossas funções.
- Iremos trabalhar com elas quando chegar o momento de utilizar ela e depois verificaremos de maneira mais detalhada.

```
20      console.log('Nome do Produto: ', this.nomeProduto);
21      console.log('Anuncio: ', this.anuncio);
22      console.log('ID:', this.idProduto);
23      console.log('Preco: ', this.precoProduto);
24      console.log('Promocão:', this.promocao);
25
26      // Escopo das variaveis dentro do código
27      var variavel1;
28      let variavel2;
29      const variavel3;
```

Na prática - Resumo

keyword	const	let	var
global scope	NO	NO	YES
function scope	YES	YES	YES
block scope	YES	YES	NO
can be reassigned	NO	YES	YES

Na prática

- Qual a saída do código abaixo?

```
var idade = 10
console.log('Minha idade é:', idade)
```

Na prática

- Qual a saída do código abaixo?

```
let idade = 10
console.log('Minha idade é:', idade)
```

Na prática

- Qual a saída do código abaixo? Testar com var e let:

```
 function imprimeIdade() {
        var idade = 20
        console.log('Minha idade é:', idade)

    imprimeIdade()
```

Na prática

- Qual a saída do código abaixo, com var?

```
function imprimeIdade() {
    for (var idade = 30; idade <= 40; idade++) {
      console.log('Idade dentro do for:', idade)
    }
    console.log('Idade fora do for:', idade)
}
imprimeIdade()
```

Na prática

- Qual a saída do código abaixo, com let?

```
  function imprimeIdade() {
      for (let idade = 30; idade <= 40; idade++) {
        console.log('Idade dentro do for:', idade)
      }
      console.log('Idade fora do for:', idade)
    }
    imprimeIdade()
```

Na prática

```
23      console.log('Nome do Produto', this.nomeProduto);
24      console.log('Anuncio', this.anuncio);
25      console.log('ID', this.idProduto);
26      console.log('Preço', this.precoProduto);
27      console.log('Promoção', this.promocao);
28
29      var variavel1;
30      let variavel2;
31     // const variavel3;
32
33      var idade = 10
34      console.log('Minha idade é:', idade)
35
36  /*    function imprimeIdade() {
37          var idade = 50
38          console.log('Minha idade é:', idade)
39      }
40      imprimeIdade() */
41
42
43      function imprimeIdade() {
44          for (let idade = 30; idade <= 40; idade++) {
45              console.log('Idade dentro do for:', idade)
46          }
47          console.log('Idade fora do for:', idade)
48      }
49      imprimeIdade()
50  }
```

Na prática

- Outra particularidade que envolve let e var.
- Quando declaramos a mesma variável com var, o último conteúdo é o que prevalece.

Na prática - const

- Assim como as variáveis declaradas com a palavra-chave let, constantes também tem escopo de bloco.
- Além disso, constantes devem ser inicializadas obrigatoriamente no momento de sua declaração.

Na prática - const

- Por que está com erro?

```
const variavel3;
```

Na prática - const

- Com a correção:

```
const variavel3 = 1;
```

- O que acontece se tentarmos alterar o valor da constante?

```
const variavel3 = 1;

var variavel3 = 2;
```

Utilizar var ou let?

- Bom, se você irá escrever seu todo seu código em ES6 (lembre-se: nem todas as features da ES6 são suportadas pelos browsers: É recomendado usar let, pois assim você conseguirá prevenir alguns bugs causados por erros de escopos e também o deixará mais fácil de ler.
- Não escreva partes com var e partes com let, ou um ou outro. Por outro lado, se você já está habituado com var e isto não te causa problemas, então continue usando.

Na prática

- Acessem o site:

https://www.alura.com.br/artigos/entenda-diferenca-entre-var-let-e-const-no-javascript?gclid=Cj0KCQjwqKuKBhCxARIsACf4XuFDEDCvctxyYVWq4NZUyxFPzvc1Qq9cRKDlogyM744Zm_vuOT0Nm40aAiiAEALw_wcB