

PROGRAMAÇÃO WEBII

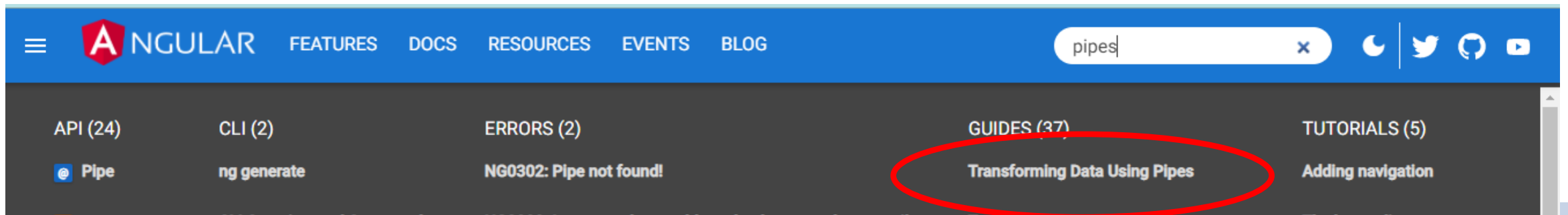
1

Formatando variáveis com Pipes

no Angular Data Binding

Vamos a documentação...

- Vamos aprender como formatar as variáveis que colocamos no html para nosso usuário ver utilizando Pipes do Angular.
- Os Pipes permitem que formatemos nossas variáveis de uma forma muito fácil e muito prática.
- Vamos acessar a documentação do Angular
 - ▶ Buscar por: Pipes
 - ▶ Selecionar: Transforming data using Pipes



Vamos a documentação...

- Ele já traz alguns pipes pré-programados para nós e nós podemos programar os nossos próprios pipes.
- Os pipes pré-programados que teremos aqui, como na documentação, temos alguns exemplos.
- O DatePipe – permite formatar datas
- UpperCasePipe – permite formatar textos deixando tudo em maiúsculo
- LowerCasePipe – deixa todas as letras em minúsculo
- CurrencyPipe – permite que formate em moedas
- DecimalPipe – para formatarmos números
- Percentpipe – para formatarmos números de porcentagem

- `DatePipe` : Formats a date value according to locale rules.
- `UpperCasePipe` : Transforms text to all upper case.
- `LowerCasePipe` : Transforms text to all lower case.
- `CurrencyPipe` : Transforms a number to a currency string, formatted according to locale rules.
- `DecimalPipe` : Transforms a number into a string with a decimal point, formatted according to locale rules.
- `PercentPipe` : Transforms a number to a percentage string, formatted according to locale rules.

Vamos a documentação...

Não temos somente esses pipes, podemos acessar a documentação completa acessando pipes API documentation. Clicando na página, teremos todos os pipes que o Angular nos fornece. Nós iremos aprender alguns, mas aprendendo esses daqui vocês já conseguiram utilizar qualquer outro pipe que você precise. Basta consultar a documentação do Angular.

- `DatePipe` : Formats a date value according to locale rules.
- `UpperCasePipe` : Transforms text to all upper case.
- `LowerCasePipe` : Transforms text to all lower case.
- `CurrencyPipe` : Transforms a number to a currency string, formatted according to locale rules.
- `DecimalPipe` : Transforms a number into a string with a decimal point, formatted according to locale rules.
- `PercentPipe` : Transforms a number to a percentage string, formatted according to locale rules.

- For a complete list of built-in pipes, see the [pipes API documentation](#).

Praticando...

- Vamos abrir o código fonte: [home.component.ts](#)
- Lá temos nossa variável nomeProduto, que está como Curso do Angular
- Percebam que não está tudo em maiúsculo, vamos transformar utilizando pipes

```
nomeProduto: string = 'Curso de Angular';  
anuncio: string = `O ${this.nomeProduto} está em promoção`;  
idProduto: number = 123;  
precoProduto: number = 2.59;  
promocao: boolean = true;  
foto: string = 'assets/img/crud.png'  
  
constructor() {  
  // variáveis de string com concatenação  
  // this.anuncio = 'O ' + this.nomeProduto + 'está em promoção!';  
  
  console.log('Nome do Produto', this.nomeProduto);  
}
```

Praticando...

- Vamos abrir o código fonte: [home.component.html](#)
- Precisamos encontrar onde estamos exibindo variável nomeProduto
 - ▶ Dentro da interpolação, inserir o símbolo do pipe | (por isso ele tem esse nome)
 - ▶ Vamos escrever uppercase para que deixe todo o texto em maiúsculo
- Salve e verifique como ficou a sua página.

```
8 <h5 class="card-title">  
9   {{ nomeProduto | uppercase }}
```

Seja bem vindo ao curso de Angular!

O Curso de Angular está em promoção



Praticando...

- Os pipes só servem para formatar variáveis. Não adianta colocar um pipe dentro do html. O Angular não irá entender, não conseguirá formatar outros textos.

Praticando...

■ Agora vamos para a variável `idProduto`. Atualmente está com o valor 123, vamos alterar para 3. Vamos deixar somente o numero 3 para vocês perceberem melhor a formatação que iremos utilizar agora.

```
11 nomeProduto: string = 'Curso de Angular';  
12 anuncio: string = `0 ${this.nomeProduto} está em promoção`;  
13 idProduto: number = 3;
```

Praticando...

- Vamos dentro da documentação novamente sobre pipes.
- Vamos ver como funciona a formação do DecimalPipe, porque ele permite que inserimos algumas propriedades para nós formatarmos, personalize a nossa formatação.
- Clique no DecimalPipe e será exibida a documentação.
- Quer que passe a nossa variável, um pipe, number. O restante é meio estranho a documentação aqui, então vamos ver os exemplos mais abaixo.

DecimalPipe

PIPE

Formats a value according to digit options and locale rules. Locale determines group sizing and separator, decimal point character, and other locale-specific configurations.

```
{{ value_expression | number [ : digitsInfo [ : locale ] ] }}
```

Praticando...

- Nos exemplos, vamos ter que escrever number:
- Utilizando apostrofe vamos passar a quantidade de dígitos mínimos de inteiros que vamos ter aqui (minIntegerDigits):
 - ▶ Do lado esquerdo, se tivermos apenas o número 3 e colocarmos aqui ao invés do 1 o número 2 para números mínimos inteiro, ele irá exibir (03)

- `minIntegerDigits`: The minimum number of integer digits before the decimal point. Default is 1.
- `minFractionDigits`: The minimum number of digits after the decimal point. Default is 0.
- `maxFractionDigits`: The maximum number of digits after the decimal point. Default is 3.

If the formatted value is truncated it will be rounded using the "to-nearest" method:

```
{3.6 | number: '1.0-0'}
```

~~<!--will output '4'-->~~

```
{-3.6 | number: '1.0-0'}
```

<!--will output '-4'-->

Praticando...

Depois inserimos o ponto (.)

- ▶ A quantidade mínima de frações que queremos ter. (minFractionDigits)
 - ▶ Se o nosso número fosse 3.52 qual seria a quantidade mínima de decimais que iríamos exibir
 - ▶ Se deixar 0, não mostra decimal
 - ▶ se digitar 1 vai exibir apenas o primeiro decimal
 - ▶ Se não tiver decimal também complementa com zero

If the formatted value is truncated it will be rounded using the "to-nearest" method:

```
{{3.6 | number: '1.0-0'}}
```

```
<!--will output '4'-->
```

```
{{-3.6 | number: '1.0-0'}}
```

```
<!--will output '-4'-->
```

Praticando...

E o máximo de decimal (maxFractionDigits)

▶ Caso tenha 5 decimais nós podemos resumir nossos decimais a apenas 2, no máximo. Vamos ver isso na prática.

- `minIntegerDigits`: The minimum number of integer digits before the decimal point. Default is 1.
- `minFractionDigits`: The minimum number of digits after the decimal point. Default is 0.
- `maxFractionDigits`: The maximum number of digits after the decimal point. Default is 3.

If the formatted value is truncated it will be rounded using the "to-nearest" method:

```
{{3.6 | number: '1.0-0'}}  
<!--will output '4'-->  
  
{{-3.6 | number: '1.0-0'}}  
<!--will output '-4'-->
```

Praticando...

Vamos na prática... Abra o código [home.component.html](#)

- ▶ Temos aqui o id do produto (idProduto)
- ▶ Vamos inserir um pipe (|)
- ▶ Vamos escrever number, dois pontos : para podermos formatar e apostrofe
- ▶ Quantidade de números inteiros mínimos: Vamos inserir 2, ponto (.)
- ▶ Quantidade de dígitos mínimo: Posso não ter nenhum, vamos digitar 0
- ▶ Quantidade de dígitos máximo: Será 3



Vamos salvar e ver o resultado



```
17 <p class="card-text">Identificação: {{ idProduto | number:'2.0-3' }}</p>
```

Praticando...

- Como nossa variável só tinha o número 3, ele mostrou 03



CREATE READ UPDATE DELETE

C R U D

CURSO DE ANGULAR

Promoção

Identificação: 03

R\$ 2.59

Praticando...

- Vamos alterar o conteúdo da variável idProduto para 3.5123
- Vamos salvar e ver o resultado
- Não exibiu o ultimo numero 3, porque informamos que era para colocar no máximo 3 dígitos na fração.

```
13 idProduto: number = 3.5123;
```

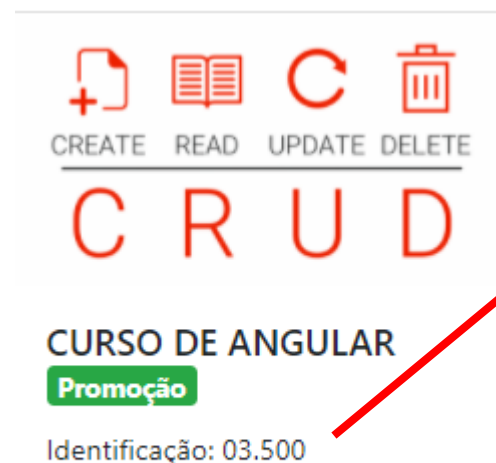


Praticando...

- Podemos informar que a fração mínima deve ser 3, independente das casas decimais que estiverem contidas na variável.
- Vamos alterar o conteúdo para 3.5
- Vamos salvar e ver o resultado

```
17 <p class="card-text">Identificação: {{ idProduto | number:'2.3-3' }}</p>
```

```
13 idProduto: number = 3.5;
```



Completo com zeros

R\$ 2.59

Praticando...

- Vamos alterar e deixar a formatação correta:
- Vamos alterar o conteúdo para 3
- Vamos deixar a formação conforme segue abaixo
- Vamos salvar e ver o resultado

```
idProduto: number = 3;
```

<p class="card-text">Identificação: {{ idProduto | number: '2.0-0' }}</p>



Praticando...

- Agora vamos trabalhar com currency, para isso vamos trabalhar com a variável preço
- Na documentação, vamos acessar o CurrencyPipe.
- Como ele funciona:
- Vamos passar qual moeda ele irá trabalhar (currencyCode)
- Se vai exibir ou não o símbolo da moeda (display)
- Quantidade de dígitos (digitsInfo)
- Localização (locale)

Parameters

currencyCode `string` The [ISO 4217](#) [🔗](#) currency code, such as `USD` for the US dollar and `EUR` for the euro. The default currency code can be configured using the `DEFAULT_CURRENCY_CODE` injection token.

Optional. Default is `this._defaultCurrencyCode`.

Praticando...

- Vamos ver isso na prática, trabalhando com o precoProduto
- Após a variável precoProduto, vamos inserir um | (pipe),
- Após o pipe, digitar currency
- Salvar e verificar como ficou na página
- Inserindo só isso já aparece o \$ na tela



```
<a href="#" class="btn btn-primary float-right">R$ {{ precoProduto | currency }}</a>
```

Praticando...

- Vamos retirar o R\$, que tínhamos inserido manualmente
- Salvar e verificar como ficou na página



```
<a href="#" class="btn btn-primary float-right"> {{ precoProduto | currency }}</a>
```

Praticando...

- Vamos informar que nossa moeda é brasileira:
 - ▶ Após o currency, digitar : 'BRL'
- Salvar e verificar como ficou na página
- Agora já temos o R\$



```
<a href="#" class="btn btn-primary float-right"> {{ precoProduto | currency:'BRL' }}</a>
```

Praticando...

- Ainda podemos colocar outras formatações, por exemplo, qual simbolo quero utilizar.
 - ▶ Após o 'BRL', digitar : 'M\$'
- Salvar e verificar como ficou na página
- Agora temos o símbolo M\$



```
<a href="#" class="btn btn-primary float-right"> {{ precoProduto | currency:'BRL':'M$' }}</a>
```

Praticando...

- Mas, podemos colocar também o symbol, se você acessar a documentação será possível visualizar.
 - ▶ Após o 'BRL', digitar : 'symbol'
- Salvar e verificar como ficou na página
- Agora retornou para o R\$



```
<a href="#" class="btn btn-primary float-right"> {{ precoProduto | currency:'BRL':'symbol' }}</a>
```


Praticando...

- Agora vamos formatar essa informação:
 - ▶ Após o 'symbol', digitar : '1:2-3), lembrando...
 - ▶ 1 – número inteiro
 - ▶ 2 – mínimo de casas decimais
 - ▶ 3 – máximo de casas decimais
- Vamos formatar o código <ALT><SHIFT><F>
- Vamos alterar o conteúdo do preço para: 2.5123
- Salvar e verificar como ficou na página



```
<a href="#" class="btn btn-primary float-right">
  {{ precoProduto | currency: "BRL":"symbol":"1.2-3" }}</a>
```

```
precoProduto: number = 2.5123;
```

Praticando...

- Vamos alterar o conteúdo do preço para: 2.51
- Salvar e verificar como ficou na página
- Assim como no number, irá assumir que o mínimo é 2 casas decimais

```
precoProduto: number = 2.51;
```



Praticando...

- Vamos deixar somente até o BRL
- Salvar e verificar como ficou na página

```
{{ precoProduto | currency: "BRL" }}
```



CURSO DE ANGULAR

Promoção

Identificação: 03

R\$2.51

Praticando...

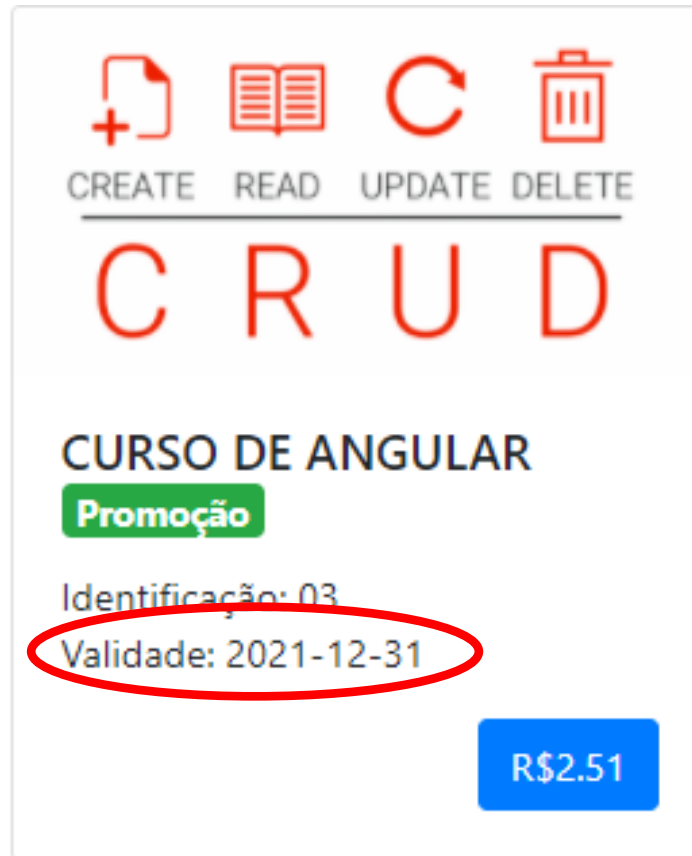
- Agora vamos ver a formatação de datas
- Para isso, vamos criar uma nova variável
 - ▶ `dataValidade = "2021-12-31"` -> utilizando a data invertida (ano-mês-dia)
- Salvar
- Vamos exibir a data:
 - ▶ Após identificação, após a tag `,p>`, vamos inserir a tag `
`
 - ▶ Digitar: Validade: `{{dataValidade}}`

```
nomeProduto: string = 'Curso de Angular';
anuncio: string = `0 ${this.nomeProduto} está em promoção`;
idProduto: number = 3;
precoProduto: number = 2.51;
promocao: boolean = true;
foto: string = 'assets/img/crud.png';
dataValidade = '2021-12-31'
```

```
<p class="card-text">Identificação: {{ idProduto | number: "2.0-0" }} <br>
| Validade: {{dataValidade}}
</p>
```

Praticando...

Vamos ver como ficou na página



Praticando...

- Como formatamos com o pipe?
- Após a data, digitamos o | (pipe), date
- Salvar e verificar como ficou na página
- Mudou para um formato padrão

```
<p class="card-text">Identificação: {{ idProduto | number: "2.0-0" }} <br>  
| Validade: {{dataValidade | date}}  
</p>
```



Praticando...

- Como podemos ter outros tipos de formatação?
- Vamos acessar a documentação DatePipe - Pre-defined format options
- Podemos formatar um desses exemplos
- Vamos selecionar o shortDate, por exemplo

- **DatePipe** : Formats a date value according to locale rules.
- **UpperCasePipe** : Transforms text to all upper case.
- **LowerCasePipe** : Transforms text to all lower case.
- **CurrencyPipe** : Transforms a number to a currency string, formatted according to locale rules.
- **DecimalPipe** : Transforms a number into a string with a decimal point, formatted according to locale rules.
- **PercentPipe** : Transforms a number to a percentage string, formatted according to locale rules.

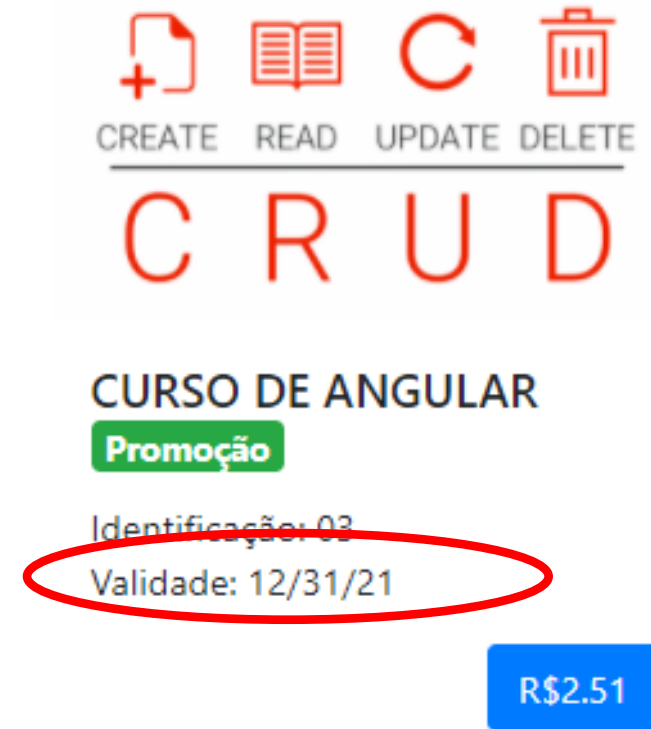
Pre-defined format options

Option	Equivalent to	Examples (given in en-US locale)
'short'	'M/d/yy, h:mm a'	6/15/15, 9:03 AM
'medium'	'MMM d, y, h:mm:ss a'	Jun 15, 2015, 9:03:01 AM
'long'	'MMMM d, y, h:mm:ss a z'	June 15, 2015 at 9:03:01 AM GMT+1
'full'	'EEEE, MMMM d, y, h:mm:ss a zzzz'	Monday, June 15, 2015 at 9:03:01 AM GMT+01:00
'shortDate'	'M/d/yy'	6/15/15
'mediumDate'	'MMM d, y'	Jun 15, 2015

Praticando...

- Após o date, digitar : 'shortDate'
- Salvar e verificar como ficou na página

```
<p class="card-text">Identificação: {{ idProduto | number: "2.0-0" }} <br>  
| Validade: {{dataValidade | date : 'shortDate'}}</p>
```



Praticando...

- Podemos criar uma formatação manual, na documentação tem como fazer isso.
- Sendo assim, vamos criar a nossa formatação, no padrão brasileiro
- Após o date:, digitar 'dd/MM/yyyy'
- Salvar e verificar como ficou na página

```
<p class="card-text">Identificação: {{ idProduto | number: "2.0-0" }} <br>
| Validade: {{dataValidade | date : 'dd/MM/yyyy'}}
</p>
```

MM – em maiúsculo



Praticando...

- Quando formatamos com o `shortDate`, ele formatou as datas todas no padrão americano.
- Vamos fazer algumas alterações para deixar nosso programa no modo “brasileiro”
- Para isso, tem uma colinha que irá nos ajudar nisso.
 - ▶ Arquivo: `local_BR.txt`

Praticando...

```
// PIPES: https://angular.io/guide/pipes
```

```
import { LOCALE_ID } from '@angular/core';
import localePt from '@angular/common/locales/pt';
import { registerLocaleData } from '@angular/common';
```

```
registerLocaleData(localePt);
```

```
providers: [
  { provide: LOCALE_ID, useValue: 'pt-BR' },
],
```

Praticando...

- Para isso, precisamos parar a execução do nosso servidor `<CTRL> <C>`
- Vamos fazer alterações no arquivo `app.module.ts`
- Vamos copiar os `import's` e colocar abaixo dos imports que temos nesse arquivo:
- Vamos copiar a linha do register e inserir abaixo dos imports
- Dentro do providers, vamos inserir a linha que está no arquivo
- Vamos subir o servidor novamente
 - ▶ `ng serve -o`

```

10 import { CadastrarProdutoComponent } from './components/produtos/cadastrar-produto';
11
12 import { LOCALE_ID } from '@angular/core';
13 import localePt from '@angular/common/locales/pt';
14 import { registerLocaleData } from '@angular/common';
15
16 registerLocaleData(localePt);
17
18
19 @NgModule({
20   declarations: [
21     AppComponent,
22     HeaderComponent,
23     FooterComponent,
24     HomeComponent,
25     ListarProdutosComponent,
26     CadastrarProdutoComponent
27   ],
28   imports: [
29     BrowserModule,
30     AppRoutingModule
31   ],
32   providers: [
33     { provide: LOCALE_ID, useValue: 'pt-BR' },
34   ],
35   bootstrap: [AppComponent]
36 })
37 export class AppModule { }

```

Praticando...

- Vamos retornar para shortDate novamente
- Salvar e verificar como ficou na página

```
<p class="card-text">Identificação: {{ idProduto | number: "2.0-0" }} <br>  
| Validade: {{dataValidade | date : 'shortDate'}}  
</p>
```

Tudo mudou..
Data já apareceu em português
R\$ já deu um espaço e as casas decimais separando com virgula
Todo programa já está seguindo o padrão que é o português- brasileiro
Quando utilizarmos os pipes de números, de data de valores ele irá assumir o padrão brasileiro


CREATE READ UPDATE DELETE
C R U D

CURSO DE ANGULAR
Promoção

Identificação: 03
Validade: 31/12/2021

R\$ 2,51

