



Pipes



Pipes

- Pipes são funções simples que podem ser usadas em expressões de templates (interpolação);
- Podem transformar strings, datas, moedas e vários outros tipos de dados;
- Aceitam um valor de entrada e retornam um valor transformado;
- São úteis porque evitam repetição de código, um pipe pode ser utilizado na aplicação inteira



Pipes built-in

- `DatePipe`: formata um valor de data, de acordo com as regras de localização;
- `UpperCasePipe`: transforma um texto para maiúsculo;
- `LowerCasePipe`: transforma um texto para minúsculo;
- `CurrencyPipe` transforma um número em uma string monetária, de acordo com as regras de localização;
- `DecimalPipe`: transforma um número em uma string com ponto decimal, respeitando as regras de localização;
- `PercentPipe`: transforma um número em uma string percentual, respeitando as regras de localização;



Localização

- Importante se o projeto necessitar suportar diversas linguagens;
- Separador decimal, símbolo da moeda e formato da data podem variar.

en-US	pt-BR
April 15, 1988	15 de abril de 1988
\$10.00	R\$10,00
1,000.50	1.000,50



Utilizando pipes built-in

<p>The hero's birthday is {{ birthday | date }}</p>

Saída na locale en-US:
The hero's birthday is Apr 15, 1988

O que aconteceria nesse caso?

<p>The hero's birthday is {{ birthday }}</p>



Pipes

- Caso necessário, é possível passar parâmetros para os pipes:

```
{{ amount | currency:'EUR' }}
```

Renderiza *amount* com símbolo do Euro

- É possível passar mais parâmetro separando os valores com dois pontos:

```
{{ amount | currency:'EUR':'Euros' }}
```



Pipes

- É possível cascadear pipes:

```
{{ birthday | date | uppercase  
    }}
```

Primeiro formata a data e em seguida deixa maiúsculo, ou seja:

1. Entrada: `Date(1988, 3, 15)`;
2. Saída DatePipe: Apr 15, 1988
3. Saída UpperCasePipe: APR 15, 1988



Pipes customizados

- Para criar um Pipe customizado, basta criar uma classe anotada com @Pipe, implementar PipeTransform e escrever a função de transformação:

```
@Pipe({name: 'addAsusPipe'})
export class AddAsusPipe implements PipeTransform {
  transform(value: string): string {
    Return 'ASUS ' + value;
  }
}
```




Pipe puro vs impuro

- Ao detectar uma mudança no valor passado ao pipe, o Angular automaticamente chama *transform* novamente para atualizar a saída;
- Por padrão, é verificada apenas a **referência** a esse valor, e não se realmente o conteúdo mudou. Esse é o caso do pipe puro;
- Para contornar esse problema existem duas soluções:
 - Alterar a referência do objeto/array passado para aquele pipe;
 - Tornar o pipe impuro (`pure: false` no decorador `@Pipe`);
- É preciso cautela ao utilizar pipes impuros, se a função de transformação for muito complexa, a aplicação pode ficar muito lenta.



Observações:

- Questionário: <https://forms.gle/zo4dqyWwEcvwozX99>
- Código fonte: <https://github.com/GabriellBP/Angular-course>
- Dúvidas:
 - gabriel.pereira@edge.ufal.br
 - lucas.amorim@edge.ufal.br



Obrigado!