

# CURSO ANGULAR

## Sys4soft (João Ribeiro)

[https://www.youtube.com/watch?v=KZrMuOEK1cE&list=PLXik\\_5Br-zO9Rly\\_0vxD4p1TF8z7p6Dx5](https://www.youtube.com/watch?v=KZrMuOEK1cE&list=PLXik_5Br-zO9Rly_0vxD4p1TF8z7p6Dx5)

Resumo do curso feito por Roberto Pinheiro

## Aula 1 - Introdução

### Requisitos necessários

- Conhecimentos de HTML5 e CSS3
- Bons conhecimentos de JavaScript/TypeScript
- NodeJS (NPM) e Angular CLI
- Editor de código: Visual Studio Code
- Browser Google Chrome

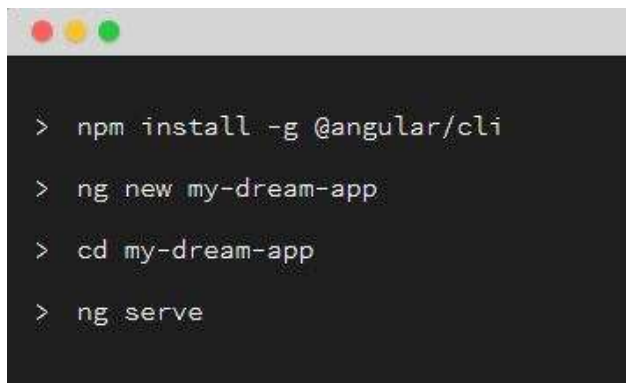
### O que fazer?

- Instalar o Visual Studio Code
- Instalar o NodeJS para ter acesso ao NPM
- Instalar o Angular CLI
- Instalar globalmente a framework
- Criar o primeiro projeto

## Aula 2 - Instalação do software necessário

- 1) Baixar e instalar o Visual Studio Code (<https://code.visualstudio.com>)
- 2) Dentro do Visual Studio Code, instalar a extensão vscode-icons
- 3) É necessário criar uma estrutura de pasta de arquivos que vai ser organizada pelo Angular CLI. E através dessa estrutura é que se começa a criar a aplicação.
- 4) Acessar a página oficial do Angular: <https://angular.io>. Na aba de recursos (Resources), na sessão Tooling, clicar em Angular CLI. Será redirecionado para a página do Angular CLI: <https://cli.angular.io>

Comandos necessários para criar a estrutura do aplicativo:

A screenshot of a terminal window with a dark background and light-colored text. The window has three colored window control buttons (red, yellow, green) in the top-left corner. The terminal displays four commands, each preceded by a prompt character '>'.

```
> npm install -g @angular/cli
> ng new my-dream-app
> cd my-dream-app
> ng serve
```

- 5) Baixar e instalar NodeJs em <https://nodejs.org>
- 6) Criar a pasta de aplicativos: C:/angular
- 7) Instalar npm a partir de qualquer pasta:

`npm install -g @angular/cli`

Os arquivos ficarão instalados em:

C:\Usuários\betorp\_000\AppData\Roaming\npm\node\_modules\@angular\cli

## Aula 3 – Criação do primeiro projeto de Angular

Na pasta C:\angular, criar o projeto "primeiro" com o comando:

```
ng new primeiro
```

Será criada a pasta "primeiro". Entrar dentro desta pasta e abrir o servidor, digitando:

```
ng serve
```

No Visual Studio Code, para abrir a pasta do aplicativo, entrar em "Open Folder", selecionar e clicar em C:\angular e depois primeiro.

Entrar na pasta src e abrir o arquivo index.html

### index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Primeiro</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Fazer uma cópia compactada, com o nome projeto.rar, das pastas e arquivos da pasta primeiro e mover esse arquivo para a pasta: C:\angular.

<app-root></app-root> é um elemento que pertence a framework do Angular.

Para carregar o aplicativo, no navegador, digitar:

localhost/4200



Na pasta `app` acessar o arquivo `app.component.html`

**app.component.html**

<!--The content below is only a placeholder and can be replaced.-->

1. Introduction

2. Background

3. Methodology

4. Results

5. Conclusion

**1. Introduction**

The purpose of this study is to investigate the impact of digital marketing strategies on consumer behavior. This research aims to provide insights into how various digital channels influence purchasing decisions and brand loyalty.

**2. Background**

In the current digital landscape, businesses are increasingly relying on online platforms to reach their target audience. Understanding the effectiveness of these strategies is crucial for sustained growth and competitive advantage.

# 

Welcome to {{ title }}!

```
<img width="300" alt="Angular Logo"
```

[illegible]

## Here are some links to help you start:

 $\langle i \rangle$ 

## [Tour of Heroes](https://angular.io/tutorial)

<|i>

## <a target="\_blank" rel="noopener" href="https://angular.io/cli">CLI Documentation</a></h2>

</i>

## <a target="\_blank" rel="noopener" href="https://blog.angular.io/">Angular blog</a></h2></li>

Excluir as linhas:

```

<h2>Here are some links to help you start: <
<ul>
  <li>
    <h2><a target="_blank" rel="noopener" href="#">
  </li>
  <li>
    <h2><a target="_blank" rel="noopener" href="#">
  </li>
  <li>
    <h2><a target="_blank" rel="noopener" href="#">
  </li>
</ul>

```

Abrir o arquivo `app.component.ts` e alterar o título (`title`) de app para `Primeiro`

#### **app.component.ts**

```
import { Component } from '@angular/core';
```

```

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  // title = 'app';
  title = 'Primeiro';
}

```

**Welcome to Primeiro!**

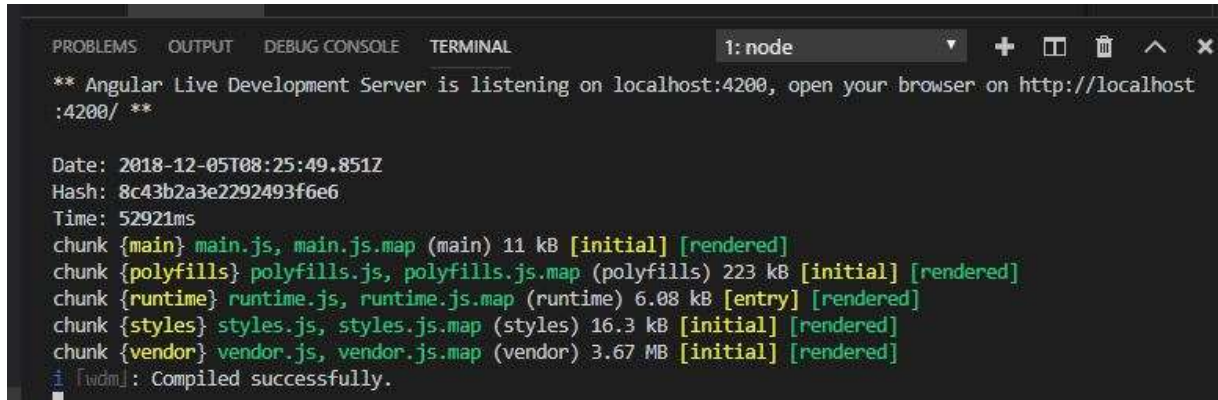


## Aula 4 – Breve análise - funcionamento do Angular

Vamos utilizar o terminal do Visual Studio Code para carregar o servidor:

No Visual Studio Code, clicar em [View → Output](#). Selecionar a aba [Terminal](#) e digitar o comando: `ng serve` e aguardar.

As informações que precisamos saber serão concentradas em poucas linhas:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: node + [icon] [icon] [icon] [icon] [icon]
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

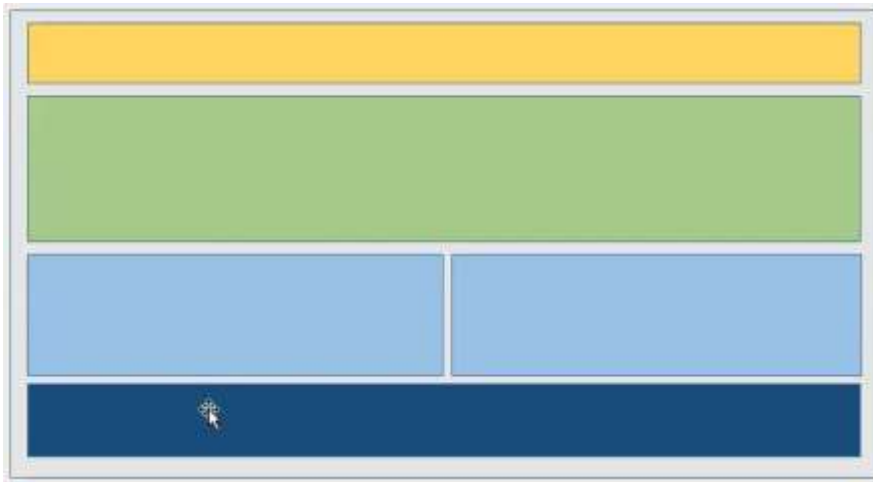
Date: 2018-12-05T08:25:49.851Z
Hash: 8c43b2a3e2292493f6e6
Time: 52921ms
chunk {main} main.js, main.js.map (main) 11 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 223 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.08 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 16.3 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 3.67 MB [initial] [rendered]
i [wdm]: Compiled successfully.
```

Se o painel for fechado e o terminal ficar oculto, para voltar a vê-lo, clique em [View → Integrated Terminal](#).

Para fechar o terminal e parar de rodar o aplicativo clique no ícone "Exit Terminal"

As alterações que o Angular faz são feitas no momento em que a página é carregada e não são exibidas no código fonte da página.

## Aula 5 – Criando o primeiro component



A vantagem dos components é que eles podem ser reutilizados.

O acréscimo de componentes não é feito dentro do arquivo [index.html](#) e sim no arquivo [app.component.html](#).

Dentro da pasta [app](#), crie uma nova pasta com o nome: [teste](#). Dentro dessa pasta criar os arquivos:

- teste.component.ts
- teste.component.html

### **teste.component.ts**

```
import { Component } from '@angular/core';
```

```
@Component({  
  selector: 'app-teste',  
  templateUrl: './teste.component.html',  
})  
export class testeComponent {  
  title = 'primeiro';  
}
```

### **teste.component.html**

```
<h3>Este é o componente de teste.</h3>
```

### **app.component.html**

```
<h3>Este é o componente de base.</h3>  
<app-teste></app-teste>
```

### app.component.ts

```
import { Component } from '@angular/core';
```

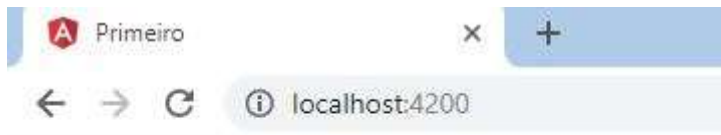
```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})  
export class AppComponent {  
  title = 'primeiro';  
}
```

### app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';  
import { NgModule } from '@angular/core';
```

```
import { AppComponent } from './app.component';  
import { testeComponent } from './teste/teste.component';
```

```
@NgModule({  
  declarations: [  
    AppComponent,  
    testeComponent  
  ],  
  imports: [  
    BrowserModule  
  ],  
  providers: [],  
  bootstrap: [AppComponent]  
})  
export class AppModule { }
```



**Este é o componente de base.**

**Este é o componente de teste.**



## Aula 6 – Exercício prático de criação de components

### Exercício:

1. Remover o componente do exercício anterior;
2. Adicionar dois novos componentes (**area1** e **area2**);
3. Colocar esses dois componentes visíveis na aplicação;

**Bônus:** No componente **area2**, criar ficheiro de CSS com formatação de um DIV com cor amarela de background e padding de 20px.

Em `app` criar duas pastas:

- `area1`
- `area2`

Na pasta `area1` criar dois arquivos:

- `area1.component.html`
- `area1.component.ts`

Na pasta `area2` criar três arquivos:

- `area2.component.html`
- `area2.component.ts`
- `area2.component.css`

**area1.component.html:**

```
<p>Area 1</p>
```

**area2.component.html:**

```
<div class='caixa'>
  <p>Area 2</p>
</div>
```

### **area1.component.ts**

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-area1',
  templateUrl: './area1.component.html',
})
export class area1 {
  title = 'area 1';
}
```

### **area2.component.ts**

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-area2',
  templateUrl: './area2.component.html',
  styleUrls: ['./area2.component.css']
})
export class area2 {
  title = 'area 2';
}
```

### **area2.component.css**

Curso\_Angular-sys4soft.docx

### **app.component.html**

<h3>Este é o componente base</h3>

<app-area1></app-area1>

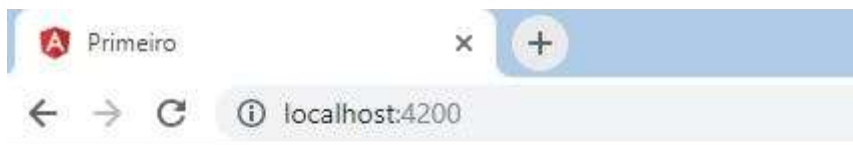
<app-area2></app-area2>

### app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
```

```
import { AppComponent } from './app.component';
import { area1 } from './area1/area1.component';
import { area2 } from './area2/area2.component';
```

```
@NgModule({
  declarations: [
    AppComponent,
    area1,
    area2
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```



### Este é o componente base

Area 1



## Aula 7 – Criar componentes com a CLI

Na pasta `C:\angular`, criar o projeto "exemplos" com o comando:

```
ng new exemplos
```

No Visual Studio Code adicionar um novo terminal (botão +)

Criar um novo component chamado socio com o comando:

```
ng g c socio --spec false
```

Dentro da pasta `app` será criada a pasta `socio` com os seguintes arquivos:

- `socio.component.css`
- `socio.component.html`
- `socio.component.ts`

### **socio.component.html**

```
<p>
  Esse é o componente sócio
</p>
```

### **socio.component.ts**

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-socio',
  templateUrl: './socio.component.html',
  styleUrls: ['./socio.component.css']
})
export class SocioComponent {

}
```

### **app.component.html**

```
<h3>Este é o componente base</h3>

<hr />

<app-socio></app-socio>
```

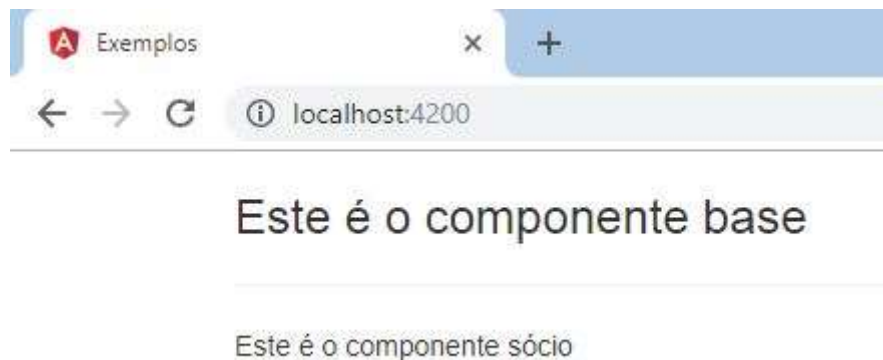
### app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
```

```
import { AppComponent } from './app.component';
import { SocioComponent } from './socio/socio.component';
```

```
@NgModule({
  declarations: [
    AppComponent,
    SocioComponent
  ],
  imports: [
    BrowserModule,
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Rodando no navegador (localhost:4200)



## Aula 8 – Nesting Components e Component Templates

### Nesting components

Alterar **socio.component.html** para:

```
<!-- <p>
  Esse é o componente sócio
</p> -->

<p>
  Uma ficha de inscrição de um sócio.
</p>
```

Criar o componente clube:

```
C:\angular\exemplos>ng g c clube --spec false
```

Será criada a pasta clube com os arquivos:

- clube.component.css
- clube.component.html
- clube.component.ts

**clube.component.html:**

```
<h3>Este é o meu clube</h3>
<hr />
<app-socio></app-socio>
<hr />
<app-socio></app-socio>
<hr />
<app-socio></app-socio>
<hr />
<app-socio></app-socio>
```

### clube.component.ts

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-clube',
  templateUrl: './clube.component.html',
  styleUrls: ['./clube.component.css']
})
export class ClubeComponent implements OnInit {

  constructor() { }

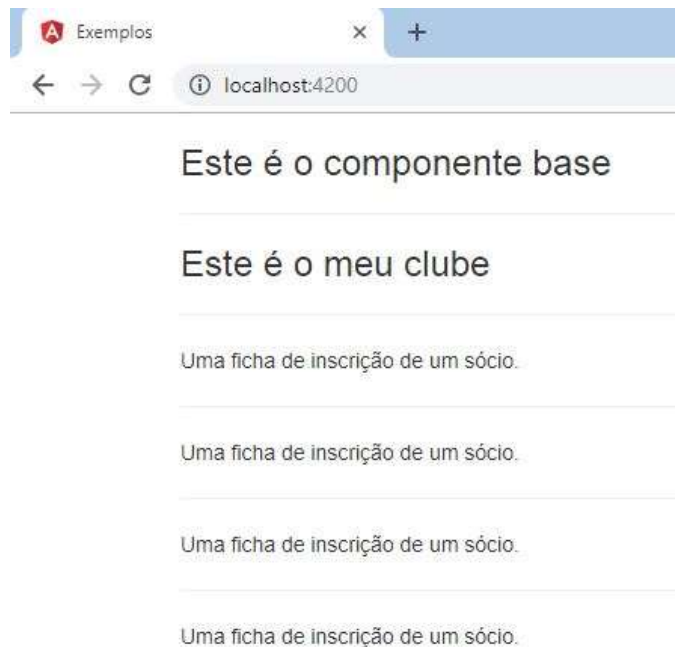
  ngOnInit() {
  }

}
```

### app.component.html:

```
<h3>Este é o componente base</h3>
<hr />
<app-clube></app-clube>
```

O resultado é:



## Component templates

Apagar os arquivos:

- socio.component.css
- socio.component.html

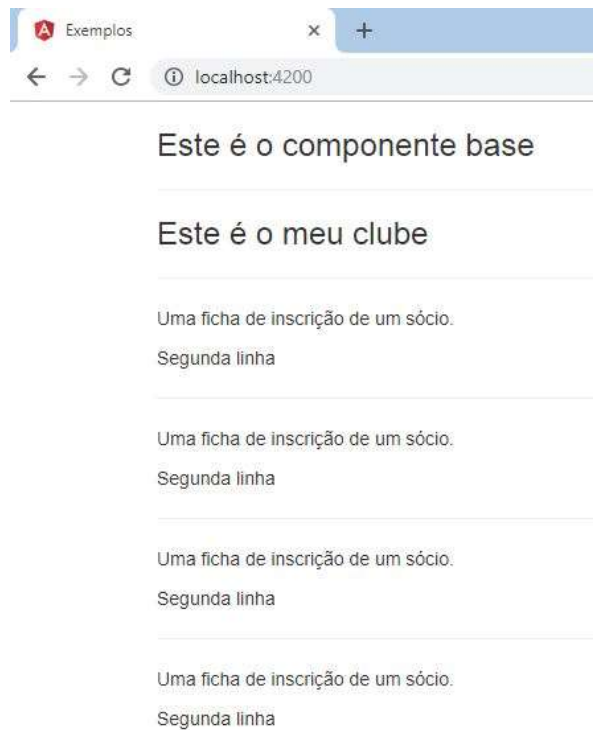
Alterar **socio.component.ts** para:

```
import { Component, OnInit } from '@angular/core';
```

```
@Component({  
  selector: 'app-socio',  
  template: `<p>Segunda linha</p>`  
})  
export class SocioComponent {  
  
}
```

Observe que em **template**, aspas simples são substituídas por acento grave.

O resultado é:





## Aula 9 – Utilizar o Bootstrap no Angular

### Instalando o Bootstrap no Angular

`npm install --save bootstrap`

O Bootstrap será instalado na pasta `node_modules`

Editar o arquivo `angular.json` alterando `styles` para:

```
"styles": [  
  "node_modules/bootstrap/dist/css/bootstrap.min.css",  
  "src/styles.css"  
],
```

Alterar `index.html` para:

```
<!doctype html>  
<html lang="pt-br">  
<head>  
  <meta charset="utf-8">  
  <title>Exemplos</title>  
  <!-- <link rel="stylesheet" href="assets/bootstrap/dist/css/bootstrap.min.css"> -->  
  <base href="/">  
  <meta name="viewport" content="width=device-width, initial-scale=1">  
  <link rel="icon" type="image/x-icon" href="favicon.ico">  
</head>  
<body>  
  
  <!-- <app-root></app-root> -->  
  
  <div class="container">  
    <div class="jumbotron">  
      <app-root></app-root>  
      <button class="btn btn-primary">Enviar</button>  
    </div>  
  </div>  
  
</body>  
</html>
```

Resultado:



**Este é o componente base**

Enviar

## Aula 10 – Utilização de CSS nos Components

Estilos CSS podem ser inseridos em qualquer um dos seguintes arquivos:

- app.component.css
- styles.css
- app.component.ts

### index.html

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>Exemplos</title>
  <link rel="stylesheet" href="assets/bootstrap/dist/css/bootstrap.min.css">
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

### app.component.html:

```
<div class="container panel panel-default">
  <h3 class="cor1">Este é o componente base</h3>
</div>
```

### Primeira forma:

#### app.component.css:

```
.cor1{
  color: blue;
}
```

Resultado:



## Segunda forma

Aplica as configurações de modo global.

Apague o estilo do arquivo `app.component.css` e altere o arquivo `styles.css` para:

```
.cor1{  
  color: red;  
}
```

Resultado:



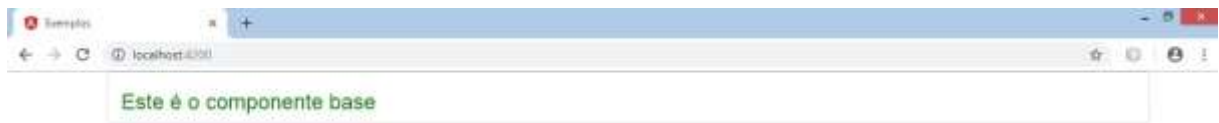
## Terceira forma

Apague o estilo do arquivo `styles.css` e altere o arquivo `app.component.ts` para:

```
import { Component } from '@angular/core';
```

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styles: ['.cor1{ color: green;}']  
})  
export class AppComponent {  
}
```

Resultado:



## Aula 11 – Uma atenção especial ao elemento Selector

Existem três formas de se utilizar selector:

### Primeira forma (como um elemento HTML)

#### app.component.ts

```
import { Component } from '@angular/core';

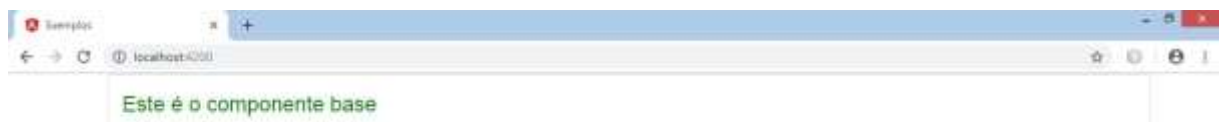
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styles: ['.cor1{ color: green;}']
})
export class AppComponent {

}
```

#### arquivo index.html

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>Exemplos</title>
  <link rel="stylesheet" href="assets/bootstrap/dist/css/bootstrap.min.css">
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```



## Segunda forma (como um atributo)

### app.component.ts

```
import { Component } from '@angular/core';

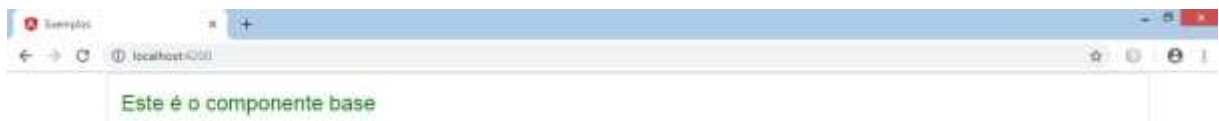
@Component({
  selector: '[app-root]',
  templateUrl: './app.component.html',
  styles: ['.cor1{ color: green;}']
})
export class AppComponent {

}
```

### arquivo index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Exemplos</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <div app-root></div>
</body>
</html>
```



## Terceira forma (como uma classe)

### app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styles: ['.cor1{ color: green;}']
})
export class AppComponent {

}
```

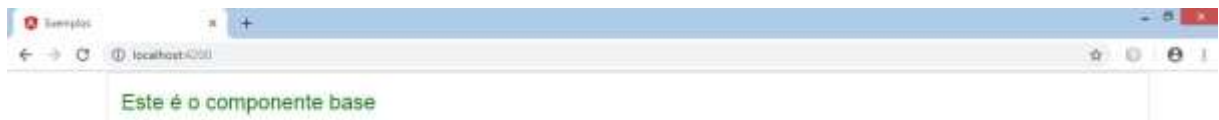
### index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Primeiro</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>

  <!-- <app-root></app-root> -->
  <!-- <div app-root></div> -->
  <div class="app-root"></div>

</body>
</html>
```



## Aula 12 – Databinding – String Interpolation

Databinding são mecanismos de comunicação entre a lógica e o template.



### Primeira forma

#### app.component.css

```
.espaco{
  margin: 20px;
  padding: 20px;
}
p{
  color: blue;
}
input[type=text]{
  padding: 10px;
}
button {
  margin: 10px;
}
```

#### app.component.ts

```
import { Component } from '@angular/core';
```

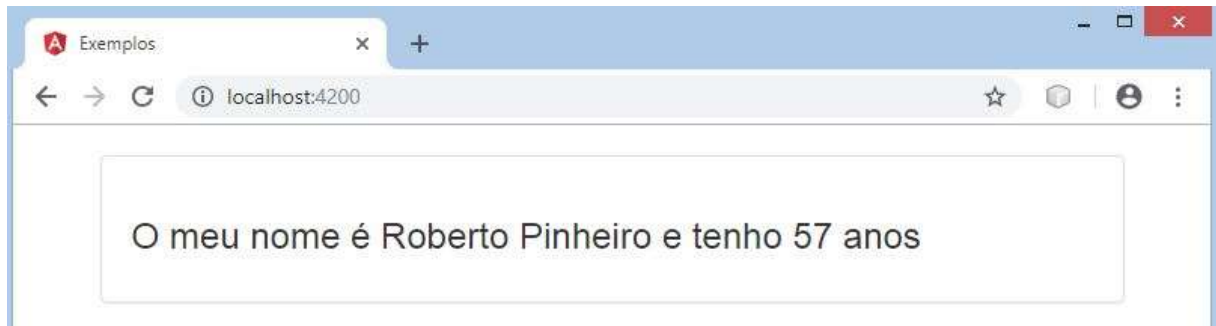
```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  nome: string = "Roberto";
  sobrenome: string = "Pinheiro";
  idade: number = 57;
}
```



### app.component.html

```
<div class="container panel panel-default espaco">
  <h3>O meu nome é {{nome}} {{sobrenome}} e tenho {{idade}} anos</h3>
</div>
```

### Resultado:



## Segunda forma

### app.component.ts

```
import { Component } from '@angular/core';
```

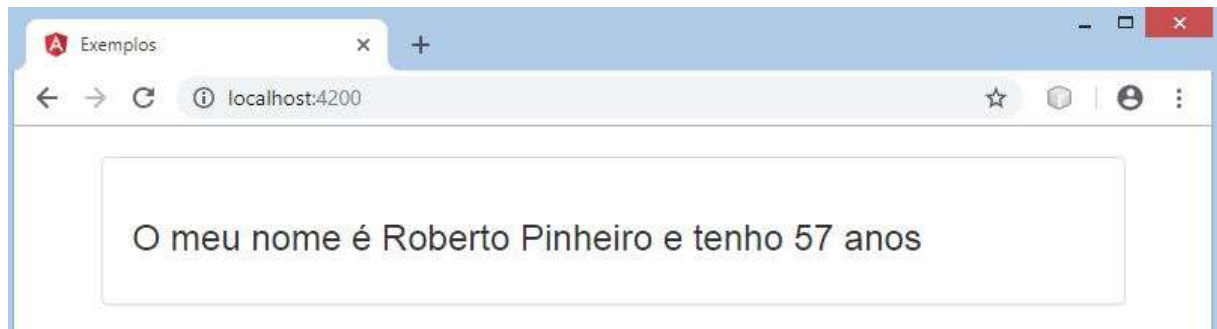
```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  nome: string = "Roberto";
  sobrenome: string = "Pinheiro";
  idade: number = 57;

  nomeCompleto() {
    return this.nome + ' ' + this.sobrenome;
  }
}
```

### app.component.html

```
<div class="container panel panel-default espaco">
  <h3>O meu nome é {{nomeCompleto()}} e tenho {{idade}} anos</h3>
</div>
```

Resultado:



## Aula 13 – Databinding - Property Binding

Durante os 5 primeiros segundos o botão e a caixa de textos ficarão desativados. Depois desse tempo, eles serão ativados.

### app.component.css

```
.espaco{
  margin: 20px;
  padding: 20px;
}
input[type=text]{
  padding: 10px;
}
button {
  margin: 10px;
}
```

### app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {
  inativo: boolean = true;
  texto: string = "Aguarde!";

  constructor(){
    this.colocarAtivo();
  }

  colocarAtivo(){
    setTimeout(() => {
      this.inativo = false;
      this.texto = "Digite o seu nome";
    }, 5000);
  }
}
```

## app.component.html

```
<div class="container">
  <div class="container panel panel-default espaco">
    <input type="text" [disabled]="inativo" [placeholder]="texto"><br /><br />
    <button class="btn btn-primary" [disabled]="inativo">Executar</button>
  </div>
</div>
```

## Resultado:

Tela inicial:



Depois de 5 segundos:



## Aula 14 – Databinding - Event Binding

### app.component.css

```
.espaco{  
  margin: 20px;  
  padding: 20px;  
}
```

```
input[type=text]{  
  padding: 10px;  
}
```

```
button {  
  margin: 10px;  
}
```

### app.component.ts

```
import { Component } from '@angular/core';
```

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

```
export class AppComponent {  
  texto: string = "Texto inicial";  
  inativo: boolean = false;
```

```
  onClick(){  
    this.texto = "Novo texto depois do click";  
    this.inativo = true;  
  }  
}
```

### app.component.html

```
<div class="container">  
  <div class="box">  
    <button class="btn btn-primary" (click)="onClick()" >Clicar</button>  
    <button class="btn btn-primary" [disabled]="inativo" >Outro botão</button>  
    <p>{{texto}}</p>  
  </div>  
</div>
```

## Resultado:

Antes do click no botão Clicar



Depois do click no botão Clicar



## Aula 15 – Exercício prático de Databinding

### String Interpolation, Property Binding e Event Binding

#### app.component.css

```
.espaco{  
  margin: 20px;  
  padding: 20px;  
}
```

```
input[type=text]{  
  padding: 10px;  
}
```

```
button {  
  margin: 10px;  
}
```

#### app.component.ts

```
import { Component } from '@angular/core';
```

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

```
export class AppComponent {
```

```
  // ativo e inativo  
  cmd2_inativo: boolean = false;  
  cmd3_inativo: boolean = false;
```

```
  // texto dos comandos  
  texto_cmd1: string = "Comando 1";  
  texto_cmd2: string = "Comando 2";  
  texto_cmd3: string = "Comando 3";
```

```
  cmd1_Click(){  
    // ativa e desativa os outros comandos  
    if(!this.cmd2_inativo){  
      this.cmd2_inativo = true;  
      this.cmd3_inativo = true;  
    } else {  
      this.cmd2_inativo = false;  
      this.cmd3_inativo = false;  
    }  
  }  
}
```

```

cmd2_Click(){
  // Muda o texto dos botões para '...'
  this.texto_cmd1 = "...";
  this.texto_cmd2 = "...";
  this.texto_cmd3 = "...";
}

cmd3_Click(){
  // Muda o texto dos botões para o texto original
  this.texto_cmd1 = "Comando 1";
  this.texto_cmd2 = "Comando 2";
  this.texto_cmd3 = "Comando 3";
}
}

```

### app.component.html

```

<div class="panel panel-default espaco">
  <div class="row">
    <!-- comando 1 -->
    <div class="col-sm-4 text-center">
      <p>Ativa e inativa os outros comandos</p>
      <button class="btn btn-primary" (click)="cmd1_Click()">{{texto_cmd1}}</button>
    </div>
    <!-- comando 2 -->
    <div class="col-sm-4 text-center">
      <p>Altera o texto dos comandos para '...'</p>
      <button class="btn btn-primary" (click)="cmd2_Click()"
[disabled]="cmd2_inativo">{{texto_cmd2}}</button>
    </div>
    <!-- comando 3 -->
    <div class="col-sm-4 text-center">
      <p>Coloca o texto original nos comandos</p>
      <button class="btn btn-primary" (click)="cmd3_Click()"
[disabled]="cmd3_inativo">{{texto_cmd3}}</button>
    </div>
  </div>
</div>

```



## Resultado:

Tela inicial



Clique no botão Comando 1



Outro clique no botão Comando 1



Clique no botão Comando 2



Clique no botão Comando 3



## Aula 16 – Event Binding e o \$event

### app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

  valor: string = "texto inicial";

  btn_Click(event: any){
    this.valor = "novo texto";
    console.log(event);
  }

  teclaPressionada(event: any){
    // console.log(event);
    this.valor = event.target.value;
  }
}
```

### app.component.html

```
<div class="panel panel-default espacio">
  <button class="btn btn-default" (click)="btn_Click($event)">Clicar</button>
  <input type="text" id="text_texto" name="text_texto" (input)="teclaPressionada($event)">
  <p>{{valor}}</p>
</div>
```

### Resultado:

Inicial:



Clicando no botão "Clicar":



Digitando na caixa de texto:



## Aula 17 – Event Binding e o \$event - parte 2

### app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

  texto: string = "";

  clicar(event: any){
    console.log(event);
    this.texto = "Clicou";
  }
  mouseover(event: any){
    this.texto = event.screenX;
  }
  mousemove(event: any){
    this.texto = event.screenX + ' - ' + event.screenY;
  }
  executar(event: Event){
    this.texto = (<HTMLInputElement>event.target).value;
  }

}
```

### app.component.html

```
<div class="panel panel-default espaco">
  <p>{{texto}}</p>
  <button class="btn btn-primary" (click)="clicar($event)" (mouseover)="mouseover($event)"
(mousemove)="mousemove($event)">EXPERIMENTAR
    ESTE BOTÃO</button>
  <input type="text" id="text" name="text" (input)="executar($event)">
</div>
```

**Resultado:**

Inicial:



Clicando no botão:



Movendo o mouse sobre o botão



## Aula 18 – Databinding nos dois sentidos

### app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {
  valor: string = "";

  // text_Changed(event: any){
  //   this.valor = event.target.value;
  // }

  alterarTexto(){
    this.valor = "Valor alterado com o botão.";
  }
}
```

### app.component.html

```
<div class="panel panel-default espaco">
  <!-- <input type="text" class="form-control" (input)="text_Changed($event)"> -->

  <input type="text" class="form-control" [(ngModel)]=valor">
  <p>{{ valor }}</p>
  <button class="btn btn-primary" (click)="alterarTexto()">Clicar</button>
</div>
```

Para trabalhar com databinding nos dois sentidos é necessário importar FormsModule, no arquivo app.module.ts

### app.module.ts

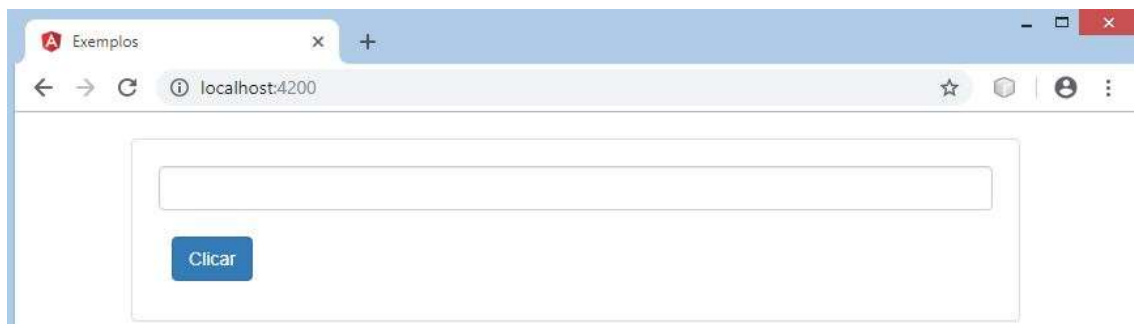
```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
```

```
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';
```

```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

### Resultado:

Tela inicial:



Digitando na caixa de texto





Clicando no botão



## Aula 19 – Introdução às Directives

### O que é uma Directive?

- São instruções inseridas dentro da DOM.
- Componentes são casos específicos de uma Directive.
- Podemos usar Directives existentes ou construir as nossas.
- Uma Directive é uma classe com o decorator @Directive

Existem outros dois tipos de Directives:

- **Structural Directives:** Alteram o layout da DOM, adicionando, removendo ou substituindo elementos;
- **Attribute Directives:** Alteram o aspecto e o comportamento de elementos existentes.

#### app.component.css

```
.espaco{  
  margin: 20px;  
  padding: 20px;  
}
```

```
input[type=text]{  
  padding: 10px;  
}
```

```
button {  
  margin: 10px;  
}
```

### app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

  apresentar: boolean = false;
  texto_botao: string = "Apresentar";

  btn_Click() {

    // Adicionar ou remover o parágrafo

    if(this.apresentar){
      this.apresentar = false;
      this.texto_botao = "Apresentar";
    } else{
      this.apresentar = true;
      this.texto_botao = "Remover";
    }
  }
}
```

### app.component.html

```
<div class="panel panel-default espaco">
  <p *ngIf="apresentar">Este é um parágrafo.</p>
  <button class="btn btn-primary" (click)="btn_Click()">{{ texto_botao }}</button>
</div>
```

### Resultado:

Inicial



Clicando no botão "Apresentar" será incluído o parágrafo:



Clicando novamente no botão "Remover", o parágrafo será removido:



## Aula 20 – Directive ngIf com uma condição Else

### app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

  apresentar: boolean = false;

  btn_Click() {
    // Adicionar ou remover o parágrafo
    if(this.apresentar){
      this.apresentar = false;
    } else{
      this.apresentar = true;
    }
  }
}
```

### app.component.html

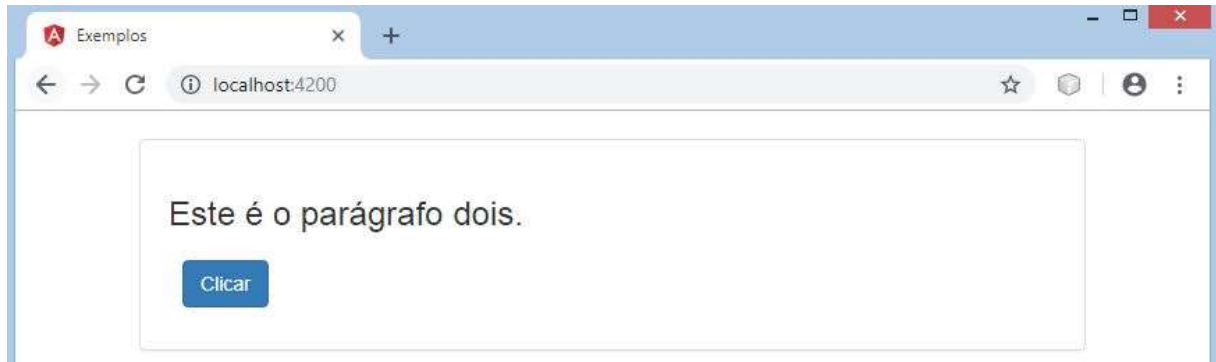
```
<div class="panel panel-default espaco">
  <p *ngIf="apresentar; else outroParagrafo">Este é o parágrafo um.</p>

  <ng-template #outroParagrafo>
    <h3>Este é o parágrafo dois.</h3>
  </ng-template>

  <button class="btn btn-primary" (click)="btn_Click()">Clicar</button>
</div>
```

## Resultado:

Tela Inicial



Clicando no botão "Clicar":



## Aula 21 – Attribute Directive ngStyle

### app.component.ts

```
import { Component } from '@angular/core';
```

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

```
export class AppComponent {
```

```
  altera_cor1: boolean = false;  
  altera_cor2: boolean = false;
```

```
  cor_texto1: string = 'white';  
  cor_fundo1: string = 'blue';  
  cor_texto2: string = 'black';  
  cor_fundo2: string = 'yellow';
```

```
  btn_Click1() {  
    if(!this.altera_cor1){  
      this.cor_texto1 = 'white';  
      this.cor_fundo1 = 'green';  
      this.altera_cor1 = true;  
    } else {  
      this.cor_texto1 = 'white';  
      this.cor_fundo1 = 'blue';  
      this.altera_cor1 = false;  
    }  
  }
```

```
  btn_Click2() {  
    if(!this.altera_cor2){  
      this.cor_texto2 = 'white';  
      this.cor_fundo2 = 'black';  
      this.altera_cor2 = true;  
    } else {  
      this.cor_texto2 = 'black';  
      this.cor_fundo2 = 'yellow';  
      this.altera_cor2 = false;  
    }  
  }  
}
```

```
}
```

## app.component.html

```
<div class="panel panel-default espaco">
  <div [ngStyle]='{color:cor_texto1,backgroundColor:cor_fundo1,'padding':'5px'}">
    <span>Texto da caixa 1</span>
  </div>

  <br />

  <div [ngStyle]="{'color':cor_texto2,backgroundColor:cor_fundo2,'padding':'5px'}">
    <span>Texto da caixa 2</span>
  </div>

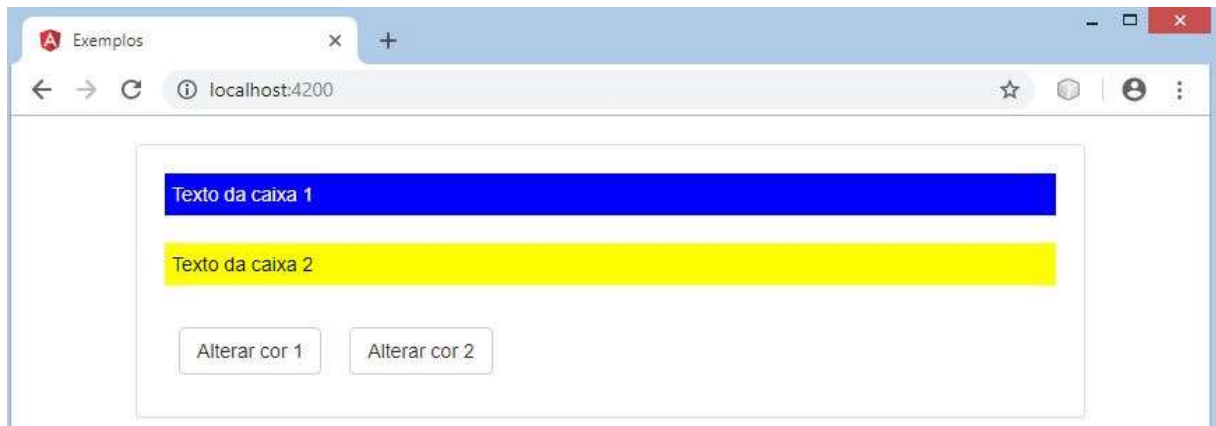
  <br />

  <button class="btn btn-default" (click)="btn_Click1()">Alterar cor 1</button>
  <button class="btn btn-default" (click)="btn_Click2()">Alterar cor 2</button>

</div>
```

## Resultado:

Tela inicial





Clicando no botão "Alterar cor 1"



Clicando no botão "Alterar cor 2"



## Aula 22 – Attribute Directive ngClass

**app.component.css**

```
.espaco{  
  margin: 20px;  
  padding: 20px;  
}
```

```
input[type=text]{  
  padding: 10px;  
}
```

```
button {  
  margin: 10px;  
}
```

```
.ativo {  
  opacity: 1;  
  font-weight: bold;  
}
```

```
.inativo {  
  opacity: 0.2;  
}
```

### app.component.ts

```
import { Component } from '@angular/core';
```

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

```
export class AppComponent {
```

```
  estado: boolean = true;
```

```
  alterarEstilo() {  
    return {  
      color: 'white',  
      backgroundColor: 'blue',  
      padding: '10px'  
    };  
  }  
}
```

```
  Alternar(){  
    if(this.estado){  
      this.estado = false;  
    } else {  
      this.estado = true;  
    }  
  }  
}
```

```
}
```

### app.component.html

```
<div class="panel panel-default espaço">
```

```
  <p [ngStyle]="alterarEstilo()">Texto de um parágrafo.</p>
```

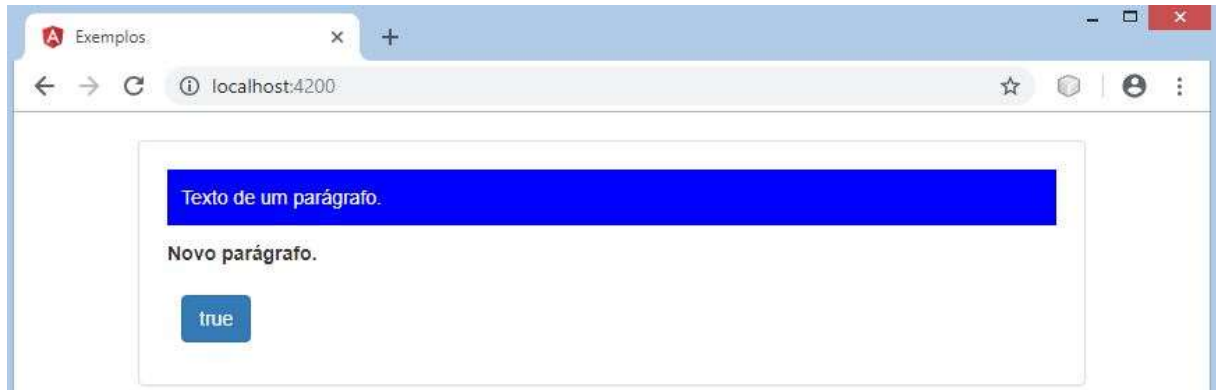
```
  <p [ngClass]="{ativo: estado === true, inativo: estado === false}">Novo parágrafo.</p>
```

```
  <button class="btn btn-primary" (click)="Alternar()">{{ estado }}</button>
```

```
</div>
```

## Resultado:

Tela inicial



Clicando no botão:



Clicando novamente no botão;



## Aula 23 – Structure Directive ngFor

### app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

  lista_nomes: string[] = [
    'João', 'Carlos', 'Ana'
  ];

}
```

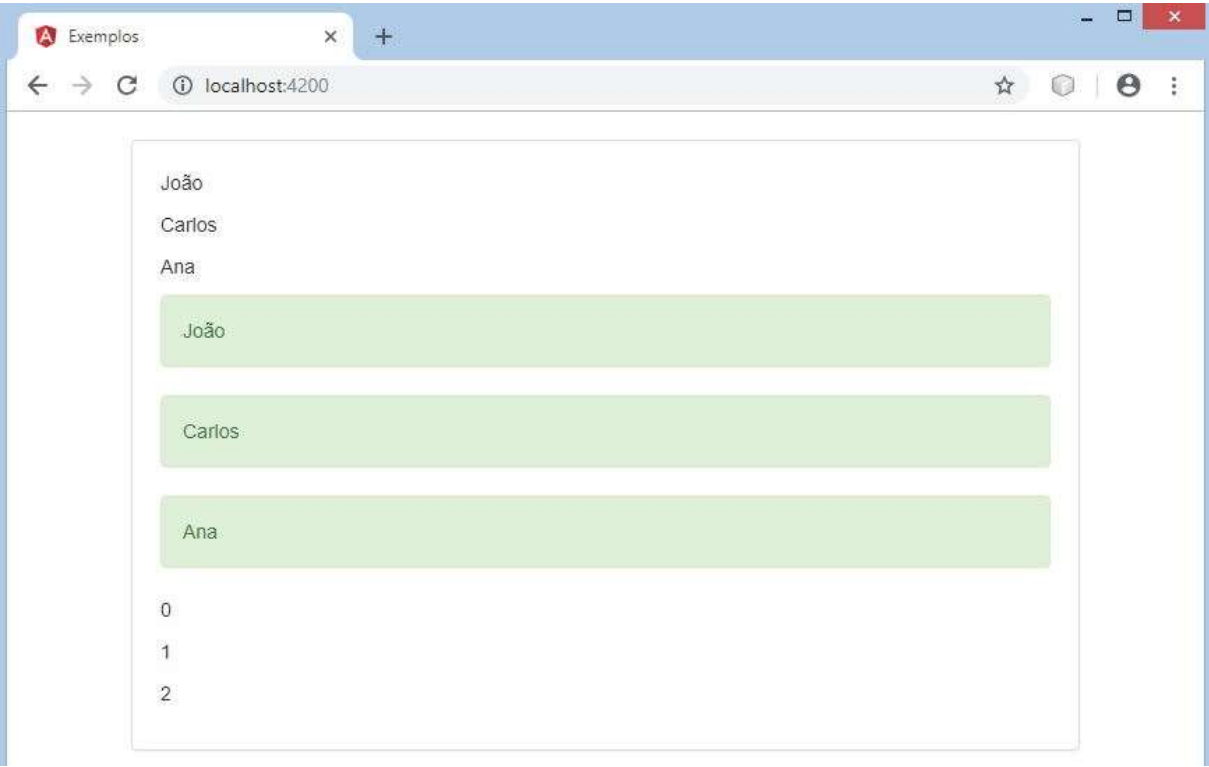
### app.component.html

```
<div class="panel panel-default espaco">
  <p *ngFor="let nome of lista_nomes">
    {{ nome }}
  </p>

  <div class="alert alert-success" *ngFor="let nome of lista_nomes">
    <p>{{ nome }}</p>
  </div>

  <div *ngFor="let nome of lista_nomes; let indice = index">
    <p>{{ indice }}</p>
  </div>
</div>>
```

Resultado:



## Aula 24 – Exercício micro loja - parte 1

### Criação do segundo projeto de Angular

Na pasta `C:\angular`, criar o projeto "micro-loja" com o comando:

```
ng new micro-loja
```

Será criada a pasta "micro-loja".

No Visual Studio Code, para abrir a pasta do aplicativo, entrar em "Open Folder" e selecionar e clicar em `C:\angular` e depois "micro-loja".

### Instalando o Bootstrap no Angular

Copiar a pasta bootstrap (bootstrap 3.3.7) para dentro da pasta assets.

Alterar o arquivo `index.html` para:

```
<!doctype html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>Micro Loja</title>
  <link rel="stylesheet" href="assets/bootstrap/dist/css/bootstrap.min.css">
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>

  <app-root></app-root>

</body>
</html>
```

Entrar dentro desta pasta e abrir o servidor, digitando:

```
ng serve
```

Abrir o arquivo `app.component.html` e trocar seu conteúdo para:

```
<div class="container panel panel-default">
  <h2>Teste</h2>
</div>
```

Para carregar o aplicativo, no navegador, digitar:

`localhost/4200`



## Aula 25 – Exercício micro loja - parte 2

- Criando o componente "loja"

```
C:\angular\micro-loja>ng g c loja
```

- Criando o componente "stock"

```
C:\angular\micro-loja>ng g c stock
```

### loja.component.html

```
<p>Loja</p>
```

### stock.component.html

```
<p>Stock</p>
```

### app.component.html

```
<div class="container panel panel-default espaco">
```

```
  <app-loja></app-loja>
```

```
  <app-stock></app-stock>
```

```
</div>
```

### app.component.css

```
.espaco{  
  margin: 20px;  
  padding: 20px;  
}
```

### app.component.html

```
<div class="container panel panel-default espaco">
```

```
  <p>Teste</p>
```

```
</div>
```

### app.component.ts

```
import { Component } from '@angular/core';
```

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

```
export class AppComponent {
```

```
}
```



## Aula 26 – Exercício micro loja - parte 3

Copie as quatro imagens para a pasta assets:



### app.component.css

```
.espaco{
  margin: 20px;
  padding: 20px;
}
.btn-size{
  width: 150px;
  margin: 5px;
}
```

### app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  // Variáveis que controlam a visibilidade dos componentes

  loja_visivel: boolean = true;
  stock_visivel: boolean = false;

  MostrarLoja(){
    // Apresenta a loja e remove o stock
    this.stock_visivel = false;
    this.loja_visivel = true;
  }

  MostrarStock(){
    // Apresenta o stock remove a loja
    this.loja_visivel = false;
    this.stock_visivel = true;
  }
}
```

## app.component.html

```
<div class="panel panel-default espaco">
  <div class="row">
    <div class="col-sm-6 col-xs-12">
      
    </div>
    <div class="col-sm-6 col-xs-12 text-right">
      <button class="btn btn-primary btn-size" (click)="MostrarLoja()">Loja</button>
      <button class="btn btn-primary btn-size" (click)="MostrarStock()">Stock</button>
    </div>
  </div>

  <app-loja *ngIf="loja_visivel"></app-loja>
  <app-stock *ngIf="stock_visivel"></app-stock>
</div>
```

Tela inicial:



Clicando no botão "Stock"



Clicando no botão "Loja"



## Aula 27 – Exercício micro loja - parte 4

loja.component.css

```
.espaco{  
  margin: 20px;  
  padding: 20px;  
}
```

```
button {  
  margin: 10px;  
}
```

## loja.component.ts

```
import { Component } from '@angular/core';
```

```
@Component({
  selector: 'app-loja',
  templateUrl: './loja.component.html',
  styleUrls: ['./loja.component.css']
})
export class LojaComponent {

  produtos = [];

  // Adiciona um ananás ao componente

  AdicionarAnanas(){
    this.produtos.push({
      'icon':'ico_ananas.png',
      'nome':'Ananás'
    });
  }

  // Adiciona uma banana ao componente

  AdicionarBanana(){
    this.produtos.push({
      'icon':'ico_banana.png',
      'nome':'Banana'
    });
  }

  // Adiciona uma laranja ao componente

  AdicionarLaranja(){
    this.produtos.push({
      'icon':'ico_laranja.png',
      'nome':'Laranja'
    });
  }
}
```

## loja.component.html

```
<div class="panel panel-default espaço">

  <!-- Título -->
  <div><h3>LOJA</h3></div>

  <!-- Conteúdo dinâmico -->
  <div class="row">
    <i *ngFor="let produto of produtos">
      <div class="col-xs-3">
        <span>{{ produto.nome}}</span>
      </div>
    </i>
  </div>

  <!-- Botões -->
  <div class="text-right">
    <button class="btn btn-success" (click)="AdicionarAnanas()">+ Ananás</button>
    <button class="btn btn-success" (click)="AdicionarBanana()">+ Banana</button>
    <button class="btn btn-success" (click)="AdicionarLaranja()">+ Laranja</button>
  </div>

</div>
```

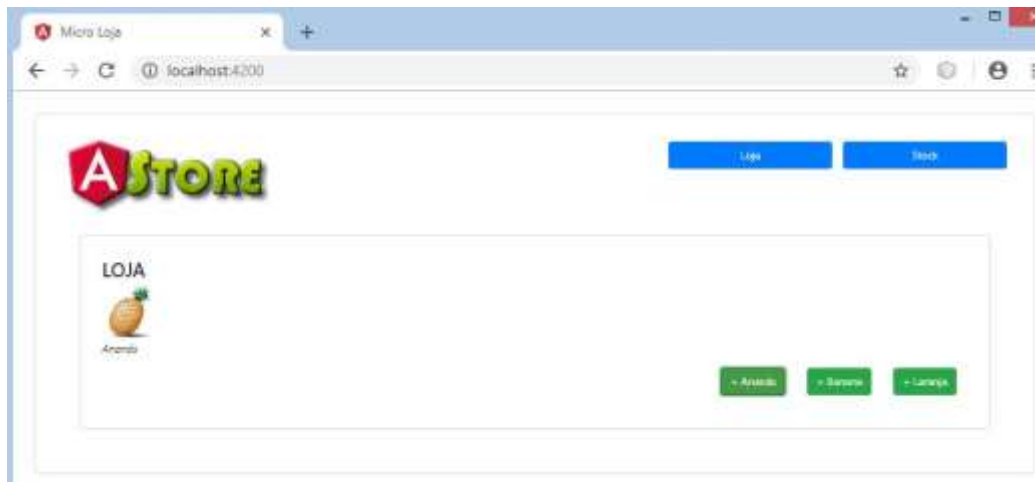
## Resultado:

Tela inicial

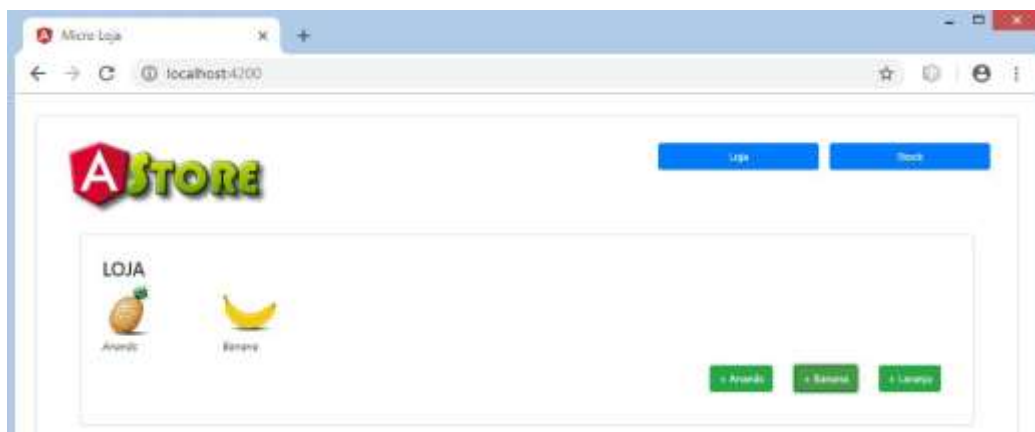




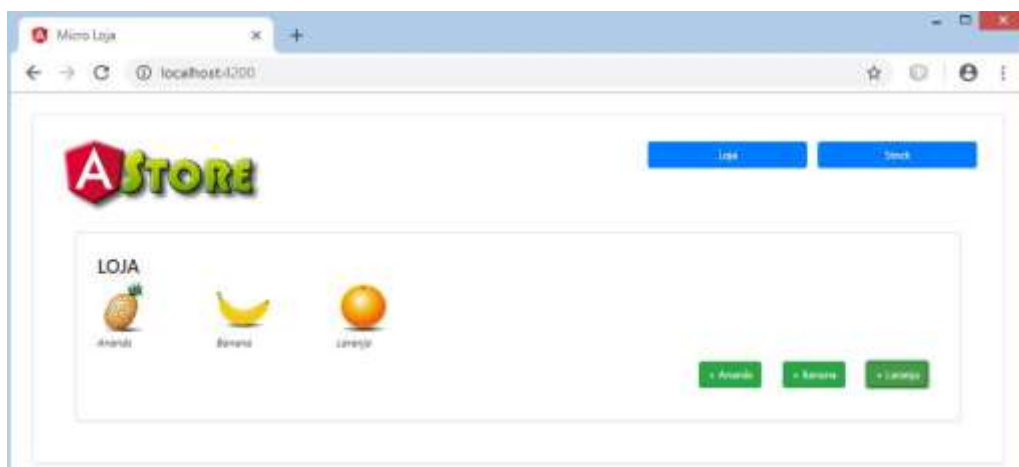
Clicando no botão +Ananás:



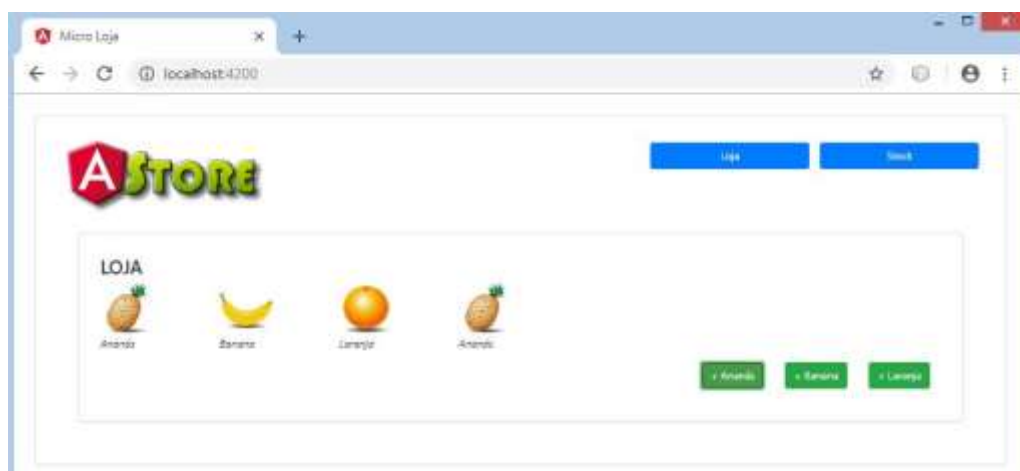
Clicando no botão +Banana:



Clicando no botão +Laranja



Clicando no botão +Ananás:



## Aula 28 – Exercício micro loja - parte 5

Atalho - Adicionar comentário:  
Shift + Alt + A

### stock.component.css

```
.espaco-stock{  
  margin: 20px;  
  padding: 20px;  
}
```

```
button {  
  margin: 10px;  
}
```

### stock.component.ts

```
import { Component } from '@angular/core';
```

```
@Component({  
  selector: 'app-stock',  
  templateUrl: './stock.component.html',  
  styleUrls: ['./stock.component.css']  
})
```

```
export class StockComponent {  
  total_ananases: number = 0;  
  total_bananas: number = 0;  
  total_laranjas: number = 0;
```

```
  AdicionarAnanas(){  
    this.total_ananases++;  
  }
```

```
  RemoverAnanas(){  
    if(this.total_ananases > 0){  
      this.total_ananases--;  
    }  
  }
```

```
  AdicionarBanana(){  
    this.total_bananas++;  
  }
```

```
  RemoverBanana(){  
    if(this.total_bananas > 0){  
      this.total_bananas--;
```

```
}  
}
```

```
AdicionarLaranja(){  
    this.total_laranjas++;  
}
```

```
RemoverLaranja(){  
    if(this.total_ananases > 0){  
        this.total_laranjas--;  
    }  
}
```

```
Calcular_Total(){  
    return this.total_ananases + this.total_bananas + this.total_laranjas;  
}
```

```
/* Estilos */
```

```
EstiloAnanases(){  
    let estilo = 'black';  
  
    if(this.total_ananases <= 0){  
        estilo = 'red';  
        return estilo;  
    }  
}
```

```
EstiloBananas(){  
    let estilo = 'black';  
    if(this.total_bananas <= 0){  
        estilo = 'red';  
        return estilo;  
    }  
}
```

```
EstiloLaranjas(){  
    let estilo = 'black';  
    if(this.total_laranjas <= 0){  
        estilo = 'red';  
        return estilo;  
    }  
}  
}
```

## stock.component.html

```
<div class="panel panel-default espaco-stock">

  <div class="row">
    <h3>STOCK</h3>
  </div>

  <!-- Quantidades de Produtos -->

  <div class="row">
    <p [ngStyle]="{color: EstiloAnanases()}">Ananases: {{ total_ananases }} </p>
  </div>

  <div class="row">
    <p [ngStyle]="{color: EstiloBananas()}">Bananas: {{ total_bananas }} </p>
  </div>

  <div class="row">
    <p [ngStyle]="{color: EstiloLaranjas()}">Laranjas: {{ total_laranjas }} </p>
  </div>

  <!-- Total dos Produtos -->

  <div class="row">
    <p>TOTAL: {{ Calcular_Total() }} </p>
  </div>

  <!-- Botões -->

  <div class="text-right">

    <button class="btn btn-success" (click)="AdicionarAnanas()">+ Ananás</button>
    <button class="btn btn-success" (click)="RemoverAnanas()">- Ananás</button>

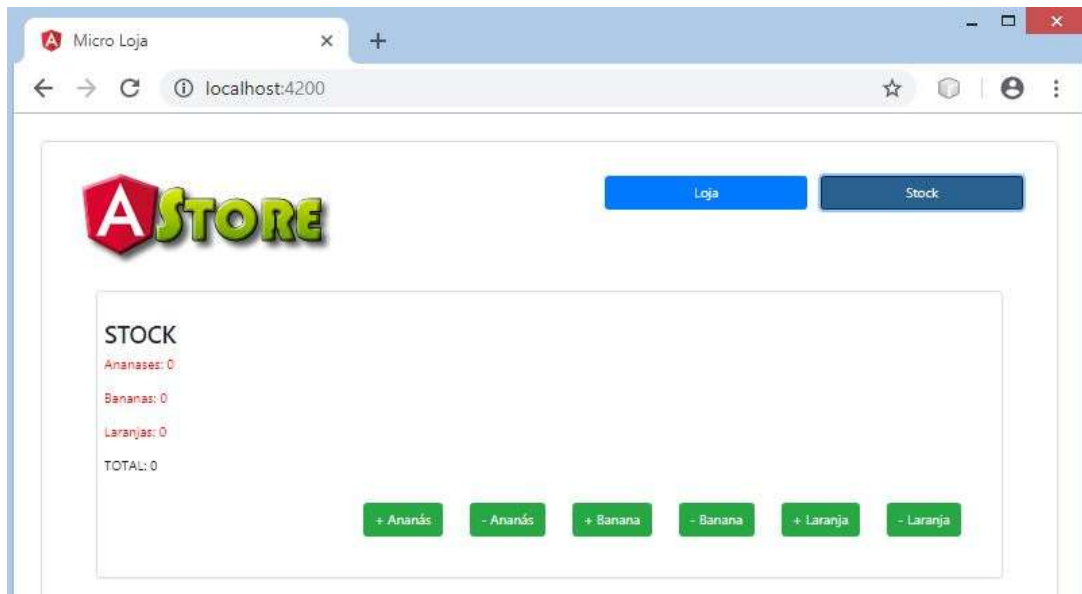
    <button class="btn btn-success" (click)="AdicionarBanana()">+ Banana</button>
    <button class="btn btn-success" (click)="RemoverBanana()">- Banana</button>

    <button class="btn btn-success" (click)="AdicionarLaranja()">+ Laranja</button>
    <button class="btn btn-success" (click)="RemoverLaranja()">- Laranja</button>

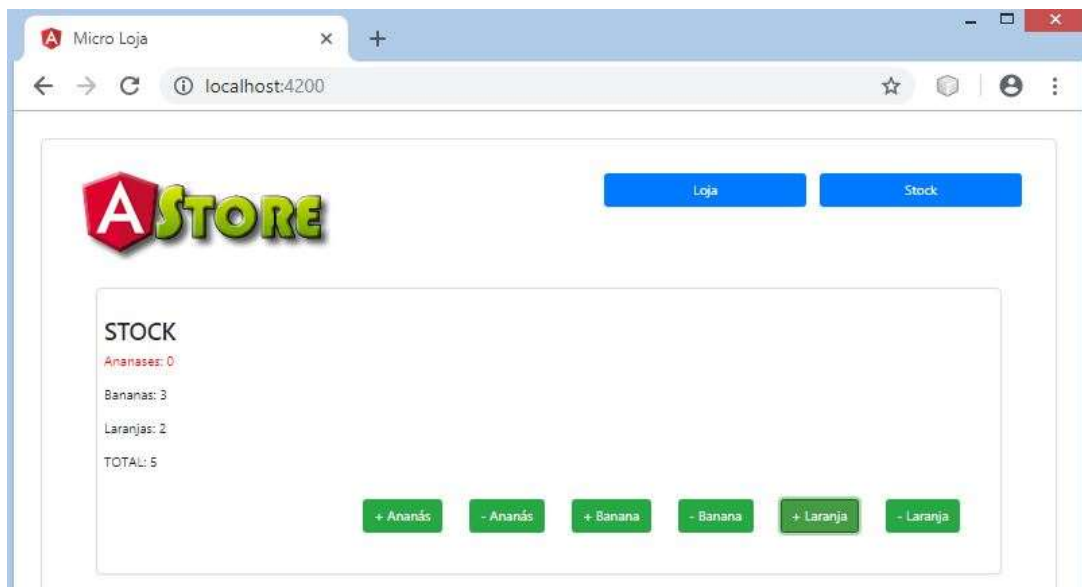
  </div>

</div>
```

## Resultado



## Adicionando bananas e laranjas



Se o valor de estoque chega a zero, a quantidade é apresentada em vermelho.

## Aula 29 – Exercício de adição de um Model

### Criação do terceiro projeto de Angular

Na pasta `C:\angular`, criar o projeto "exemplos2" com o comando:

```
ng new exemplos2
```

Será criada a pasta "exemplos2".

No Visual Studio Code, para abrir a pasta do aplicativo, entrar em "Open Folder" e selecionar e clicar em `C:\angular` e depois "exemplos2".

### Criando um modelo

Model é uma classe.

Crie dentro de app o arquivo `operacao.model.ts`

#### **operacao.model.ts**

```
export class cl_operacao {

    // valor1 + valor2 = resultado

    public valor1: number;
    public valor2: number;
    public tipo_operacao: number;
    public resultado: number;

    public str_operacao: string;

    //=====

    constructor(){
        // criar uma operação matemática aleatória
        this.valor1 = this.GerarValorAleatorio(1,10);
        this.valor2 = this.GerarValorAleatorio(1,10);
        this.tipo_operacao = this.GerarValorAleatorio(1,4);

        switch (this.tipo_operacao) {
            // adicao
            case 1:
                this.str_operacao = this.valor1 + ' + ' + this.valor2 + ' = ';
                this.resultado = this.valor1 + this.valor2;
                break;
            // subtração
```

```

    case 2:
        this.str_operacao = this.valor1 + " - " + this.valor2 + " = ";
        this.resultado = this.valor1 - this.valor2;
        break;
    // multiplicação
    case 3:
        this.str_operacao = this.valor1 + " x " + this.valor2 + " = ";
        this.resultado = this.valor1 * this.valor2;
        break;
    // divisão
    case 4:
        this.str_operacao = this.valor1 + " : " + this.valor2 + " = ";
        this.resultado = this.valor1 / this.valor2;
        break;
    default:
        break;
}
}

GerarValorAleatorio(min, max){
    return Math.floor(Math.random() * (max - min + 1) + min);
}
}

```

#### **app.component.html**

```

<div class="container panel panel-default espaco">

    <p>{{ operacao.str_operacao }} {{ operacao.resultado }}</p>
    <button (click)="CriarOperacao()" class="btn btn-primary">Criar Operação</button>

</div>

```



### app.component.ts

```
import { Component } from '@angular/core';  
import { cl_operacao } from './operacao.model';
```

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

```
export class AppComponent {
```

```
  operacao: cl_operacao;
```

```
  CriarOperacao(){  
    this.operacao = new cl_operacao;  
  }
```

```
}
```

### Resultado:

Tela inicial



Operações aleatórias:

Clique no botão "Criar Operação"





## Aula 30 – de parent component para child component

Comunicação entre componentes (parent e child) no Angular.

`@Input()`

Crie o componente "socios"

`ng g c socios --specs false`

### **socios.component.html**

```
<p>
  Nome: {{ nome }}
</p>
```

### **socios.component.ts**

```
import { Component, Input } from '@angular/core';
```

```
@Component({
  selector: 'app-socios',
  templateUrl: './socios.component.html',
  styleUrls: ['./socios.component.css']
})
```

```
export class SociosComponent {
```

```
  @Input() nome: string = "João"; // fica público para ser alterado pelo componente pai
```

```
  constructor() { }
```

```
}
```

### **app.component.html**

```
<div class="container panel panel-default espaco">
  <app-socios [nome] = 'novo_nome'></app-socios>
</div>
```

### app.component.ts

```
import { Component } from '@angular/core';  
import { cl_operacao } from './operacao.model';
```

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

```
export class AppComponent {  
  novo_nome: string = "JOÃO RIBEIRO";  
}
```

### Resultado:



### socios.component.ts

```
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-socios',
  templateUrl: './socios.component.html',
  styleUrls: ['./socios.component.css']
})

export class SociosComponent {
  socios = [];

  constructor() {
    this.socios.push({
      nome: 'João',
      contato: 12345
    });
  }
}
```

### socios.component.html

```
<p>Nome: {{ socios[0].nome }}</p>
<p>Contato: {{ socios[0].contato }}</p>
```

### app.component.html

```
<div class="container panel panel-default espaco">
  <app-socios></app-socios>
</div>
```

### Resultado:



### socios.component.ts

```
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-socios',
  templateUrl: './socios.component.html',
  styleUrls: ['./socios.component.css']
})

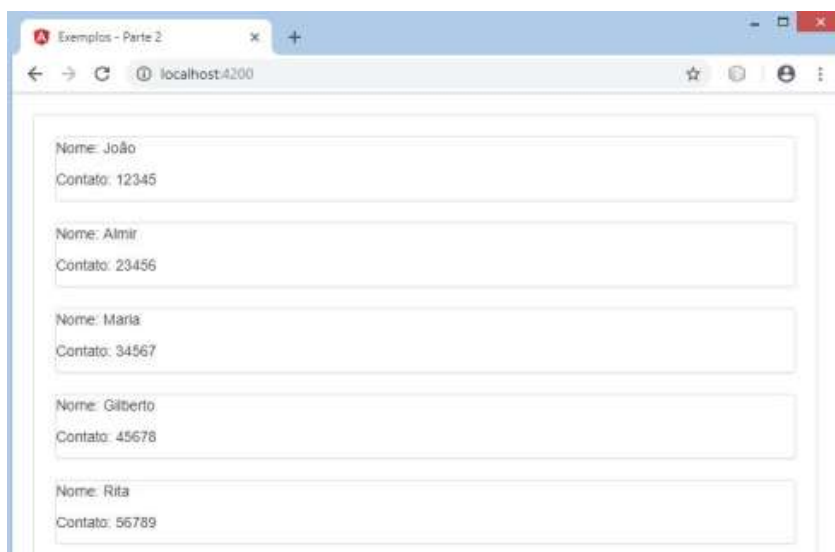
export class SociosComponent {
  socios = [];

  constructor() {
    this.socios.push({nome: 'João', contato: 12345});
    this.socios.push({nome: 'Almir', contato: 23456});
    this.socios.push({nome: 'Maria', contato: 34567});
    this.socios.push({nome: 'Gilberto', contato: 45678});
    this.socios.push({nome: 'Rita', contato: 56789});
  }
}
```

### socios.component.html

```
<div class="panel panel-default espacio" *ngFor = "let socio of socios" >
  <p>Nome: {{ socio.nome }}</p>
  <p>Contato: {{ socio.contato }}</p>
</div>
```

### Resultado:



### **app.component.ts**

```
import { Component, Input } from '@angular/core';
import { cl_operacao } from './operacao.model';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

  novos_socios = [];

  constructor(){
    this.novos_socios.push({nome: 'Sócio 1',contato: 11111});
    this.novos_socios.push({nome: 'Sócio 2',contato: 22222});
    this.novos_socios.push({nome: 'Sócio 3',contato: 33333});
    this.novos_socios.push({nome: 'Sócio 4',contato: 44444});
  }

}
```

### **app.component.html**

```
<div class="container panel panel-default espaco">
  <app-socios [socios] = 'novos_socios'></app-socios>
</div>
```

### socios.component.ts

```
import { Component, Input } from '@angular/core';

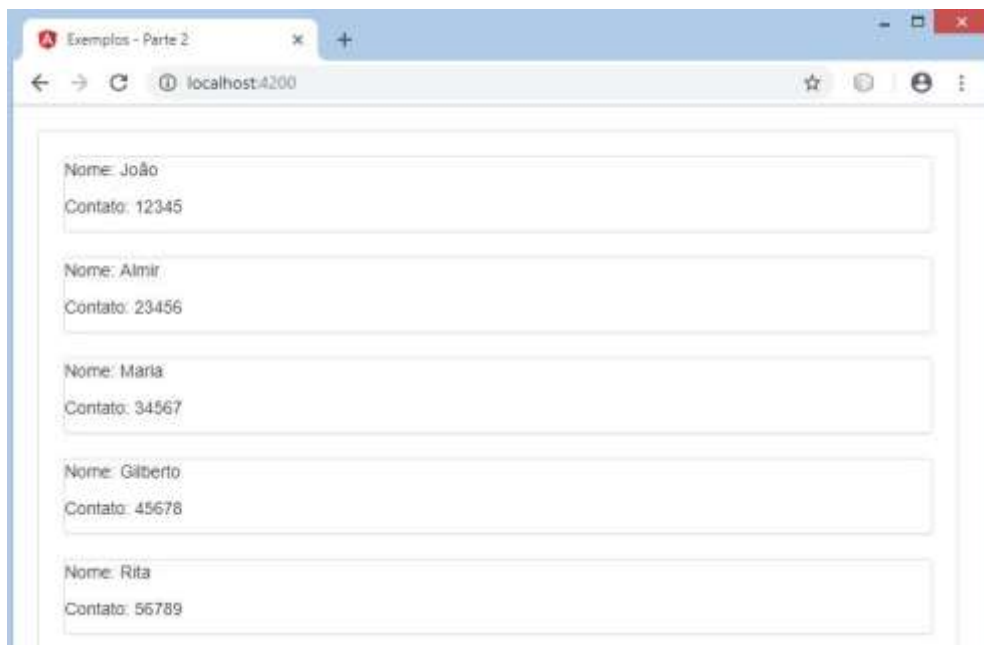
@Component({
  selector: 'app-socios',
  templateUrl: './socios.component.html',
  styleUrls: ['./socios.component.css']
})

export class SociosComponent {

  @Input() socios = [];

  constructor() {
    this.socios.push({nome: 'João', contato: 12345});
    this.socios.push({nome: 'Almir', contato: 23456});
    this.socios.push({nome: 'Maria', contato: 34567});
    this.socios.push({nome: 'Gilberto', contato: 45678});
    this.socios.push({nome: 'Rita', contato: 56789});
  }
}
```

### Resultado:





## Aula 31 – Usando um Alias no @Input

### socios.component.ts

```
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-socios',
  templateUrl: './socios.component.html',
  styleUrls: ['./socios.component.css']
})

export class SociosComponent {

  @Input('nome_novo') nome: string = "João";

}
```

### socios.component.html

```
<div class="panel panel-default espaco">
  <p>Nome: {{ nome }}</p>
</div>
```

### app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

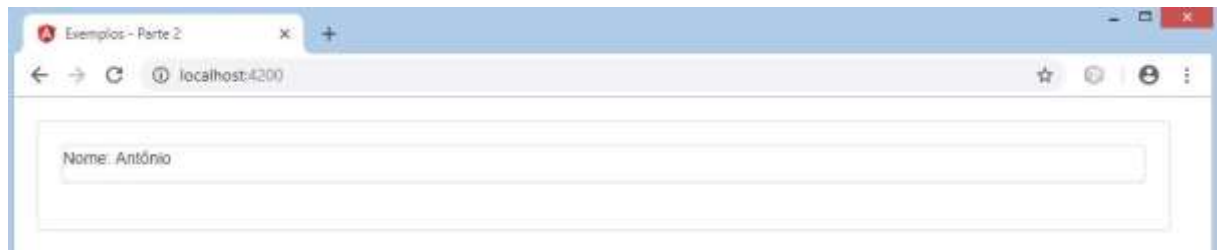
  valor: string = "Antônio";

}
```

### app.component.html

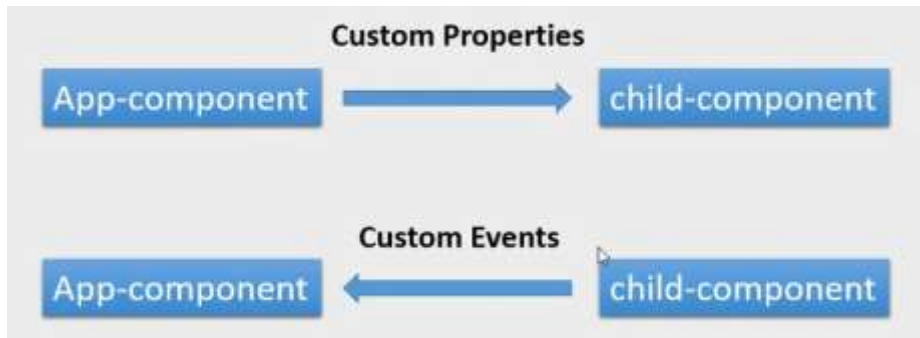
```
<div class="container panel panel-default espaco">
  <app-socios [nome_novo] = 'valor'></app-socios>
</div>
```

## Resultado:



## Aula 32 – Custom Events Binding

### Comunicação entre components



Crie o componente "meu":

```
ng g c meu --spec false
```

#### **meu.component.ts**

```
import { Component, EventEmitter, Output } from '@angular/core';
```

```
@Component({
  selector: 'app-meu',
  templateUrl: './meu.component.html',
  styleUrls: ['./meu.component.css']
})
```

```
export class MeuComponent {
```

```
  nome: string = "João";
  @Output() evento = new EventEmitter();
```

```
  AlterarNome(){
    this.nome = "Antônio";
    this.evento.emit();
  }
```

```
}
```

#### **meu.component.html**

```
<p>Nome: {{ nome }}</p>
```

```
<button (click)="AlterarNome()">Alterar</button>
```

### app.component.ts

```
import { Component } from '@angular/core';
```

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

```
export class AppComponent {
```

```
  info: string = "Tudo calmo";
```

```
  Escuta(){  
    this.info = "Nome alterado";  
  }  
}
```

### app.component.html

```
<div class="container panel panel-default espaco">
```

```
  <app-meu (evento)="Escuta()"></app-meu>
```

```
  <p>{{ info }}</p>
```

```
</div>
```

### Resultado

Tela inicial



Clicando no botão "Alterar"



## Aula 33 – Custom Events Binding - comunicando valores

### meu.component.ts

```
import { Component, EventEmitter, Output } from '@angular/core';

@Component({
  selector: 'app-meu',
  templateUrl: './meu.component.html',
  styleUrls: ['./meu.component.css']
})

export class MeuComponent {

  nome: string = "João";
  @Output() evento = new EventEmitter<string>();

  AlterarNome(){
    this.nome = "Antônio";
    this.evento.emit(this.nome);
  }
}
```

### app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

  info: string = "Tudo calmo";

  Escuta(valor: string){
    this.info = "O nome foi alterado para " + valor;
  }
}
```

## app.component.html

```
<div class="container panel panel-default espaço">  
  <app-meu (evento)="Escuta($event)"></app-meu>  
  <p>{{ info }}</p>  
</div>
```

## Tela inicial



Clicando no botão "Alterar":



## Aulas 34 e 35 – Jogo matemático com Angular

Crie o componente "jogo"

```
ng g c jogo --spec false
```

### jogo.component.ts

```
import { Component, EventEmitter, Output } from '@angular/core';
```

```
@Component({  
  selector: 'app-jogo',  
  templateUrl: './jogo.component.html',  
  styleUrls: ['./jogo.component.css']  
})
```

```
export class JogoComponent {
```

```
  // elementos que constituem a operação
```

```
  valor1: number = 0;
```

```
  valor2: number = 0;
```

```
  resultado: number = 0;
```

```
  str_operacao: string = "";
```

```
  valor_inserido: number;
```

```
  // event emitter para informar o app.component
```

```
  @Output() resposta_final = new EventEmitter();
```

```
  constructor() {
```

```
    this.CriarNovaOperacao();
```

```
  }
```

```
  CriarNovaOperacao(){
```

```
    // cria aleatoriamente uma adição
```

```
    this.valor1 = Math.floor(Math.random()*10);
```

```
    this.valor2 = Math.floor(Math.random()*10);
```

```
    this.resultado = this.valor1 + this.valor2;
```

```
    this.str_operacao = this.valor1 + " + " + this.valor2 + " = ";
```

```
  }
```

```
  DefinirResultado(Evento: any){
```

```
    // atualiza o valor inserido dentro do input
```

```
    this.valor_inserido = Evento.target.value;
```

```
  }
```

```
  AvaliarResultadoInserido(){
```

```
    // Avalia se a operação foi bem solucionada ou não
```

```
    if(this.valor_inserido == this.resultado){
```

```
      this.resposta_final.emit(true);
```

```
    } else {
```



```

        this.resposta_final.emit(false);
    }
    // criar uma nova operação
    this.CriarNovaOperacao();
}
}

```

### jogo.component.html

```

<div class="panel panel-default espaco">
  <h2>Jogo matemático</h2>
  <hr />

  <div class="form-inline form-group">
    <label><h3>{{ str_operacao }}</h3></label>
    <input type="number" (input) = "DefinirResultado($event)">
    <button class="btn btn-primary" (click)="AvaliarResultadoInserido()" >Ok</button>
  </div>

  <!-- <p>{{ valor_inserido }}</p> -->

</div>

```

### app.component.ts

```

import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {
  resposta: string = "";
  respostas_certas: number = 0;
  respostas_erradas: number = 0;
  AvaliarResposta(valor){
    if(valor){
      this.resposta = "Acertou :)";
      this.respostas_certas++;
    } else {
      this.resposta = "Errou :(";
      this.respostas_erradas++;
    }
  }
}

```

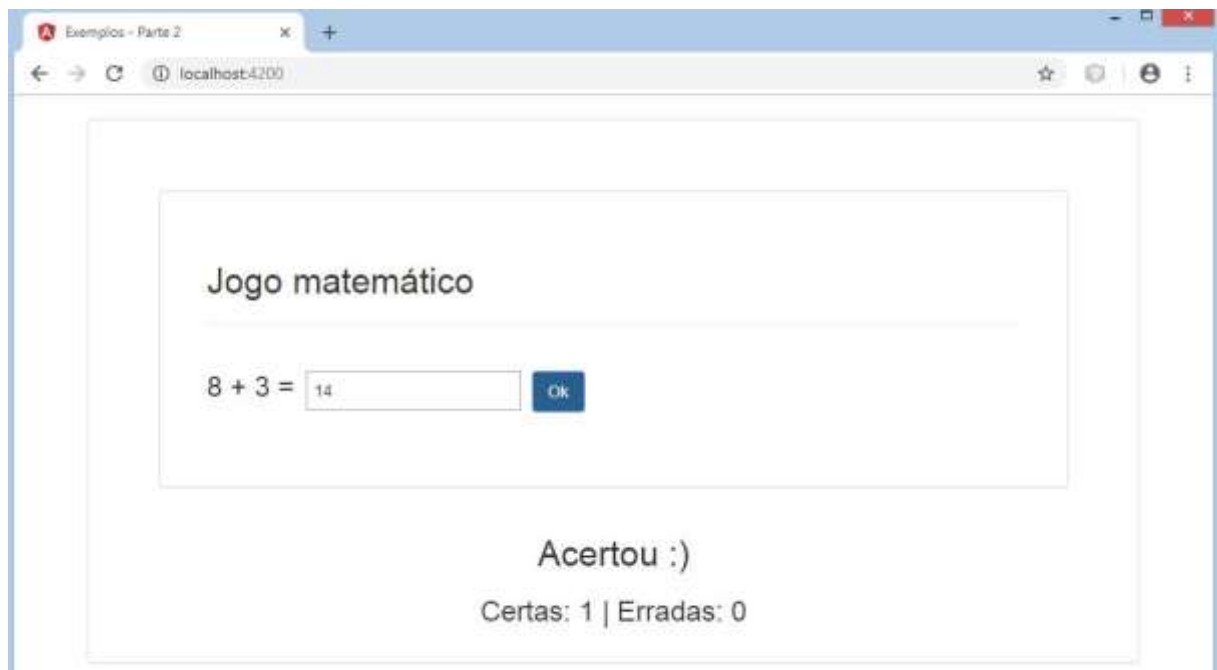
## app.component.html

```
<div class="container">
  <div class="panel panel-default espaco">
    <app-jogo (resposta_final) = "AvaliarResposta($event)"></app-jogo>

    <div class="text-center">
      <h2>{{ resposta }}</h2>
      <div class="text-center">
        <h3>Certas: {{ respostas_certas }} | Erradas: {{ respostas_erradas }}</h3>
      </div>
    </div>
  </div>
</div>
```

## Resultado:





## Aula 36 – Compreendendo o CSS e View Encapsulation

Crie os componentes "comp1" e "comp2":

```
ng g c comp1 --spec false
ng g c comp2 --spec false
```

### comp1.component.css

```
p{
  color: green;
  font-size: 2em;
}
```

### comp1.component.ts

```
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
```

```
@Component({
  selector: 'app-comp1',
  templateUrl: './comp1.component.html',
  styleUrls: ['./comp1.component.css'],
  encapsulation: ViewEncapsulation.None
})
export class Comp1Component implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

### comp1.component.html

```
<p>
  comp1 works!
</p>
```

### comp2.component.ts

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-comp2',
  templateUrl: './comp2.component.html',
  styleUrls: ['./comp2.component.css']
})
export class Comp2Component implements OnInit {

  constructor() { }
  ngOnInit() {
  }

}
```

### comp2.component.html

```
<p>
  comp2 works!
</p>
```

### app.component.html

```
<div class="container">
  <div class="panel panel-default espacio">
    <p>Componente principal</p>
    <app-comp1></app-comp1>
    <app-comp2></app-comp2>
  </div>
</div>
```

Se comp2.component.css estiver vazio:



Caso contrário:

**comp2.component.css**

```
p{  
  color: yellow;  
}
```



## Aula 37 – Utilização de Local References

### app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

  valor: string = 'mensagem';

  AlterarTexto(e:HTMLInputElement, ee:HTMLElement){
    // this.valor = e.target.value;
    this.valor = e.value;
    ee.className = "alert alert-success";
    // console.log(e);
  }

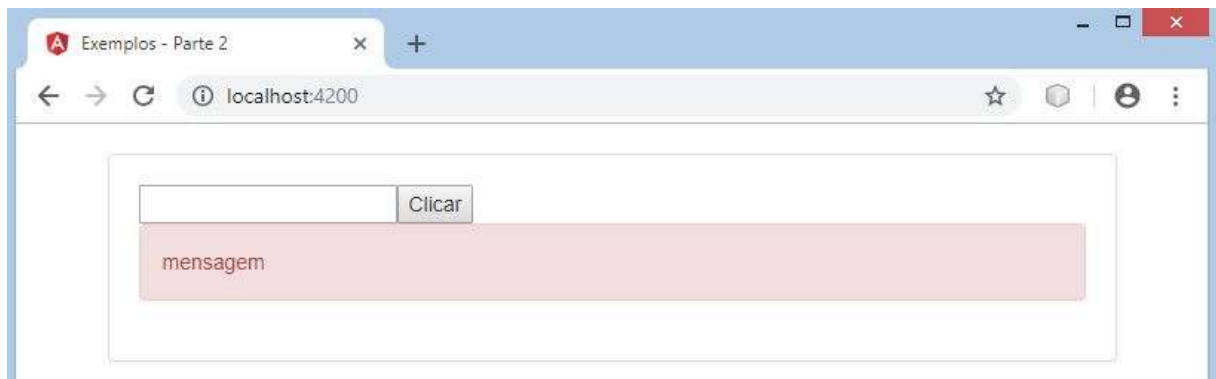
}
```

### app.component.html

```
<div class="container">
  <div class="panel panel-default espaco">
    <input type="text" #elementoInput>
    <button (click)=AlterarTexto(elementoInput,paragrafo)>Clicar</button>
    <p class="alert alert-danger" #paragrafo>{{valor}}</p>
  </div>
</div>
```

## Resultado:

Tela inicial:



Inserindo o texto "Sucesso" na caixa de texto e em seguida clicando no botão "Clicar":





## Aula 38 – Acesso a elementos da DOM com ViewChild

### app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

  valor: string = 'mensagem';

  @ViewChild('meuInput') meuElemento: ElementRef;
  @ViewChild('meuParagrafo') meuParagrafo: ElementRef;

  /* Executar(e: HTMLInputElement) {
    this.valor = e.value;
  }
  */

  Executar() {
    this.valor = this.meuElemento.nativeElement.value;
    this.meuParagrafo.nativeElement.className = "alert alert-success";
  }

}
```

### app.component.html

```
<div class="container">
  <div class="panel panel-default espaco">
    <input type="text" #meuInput>
    <button (click)=Executar(meuInput)>Clicar</button>
    <p class="alert alert-danger" #meuParagrafo>{{valor}}</p>
  </div>
</div>
```

## Resultado:

Tela inicial:



Inserindo o texto "Sucesso" na caixa de texto e em seguida clicando no botão "Clicar":



## Aula 39 – Utilização da diretiva ng content

Crie um component chamado "noticia":

```
ng g c noticia --spec false
```

### noticia.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-noticia',
  templateUrl: './noticia.component.html',
  styleUrls: ['./noticia.component.css']
})
export class NoticiaComponent {

  titulo: string = 'Título da notícia';
  texto: string = 'Elit cupidatat elit dolor incididunt ut ullamco do nisi consectetur quis ullamco. Nulla reprehenderit labore est eiusmod duis id magna sunt nostrud ullamco. Velit nulla ad et Lorem id in cillum. Qui enim esse labore nulla irure est esse.';

}
```

### noticia.component.html

```
<div class="panel panel-default espacio">
  <h2>{{titulo}}</h2>
  <ng-content></ng-content>
</div>
```

### app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {

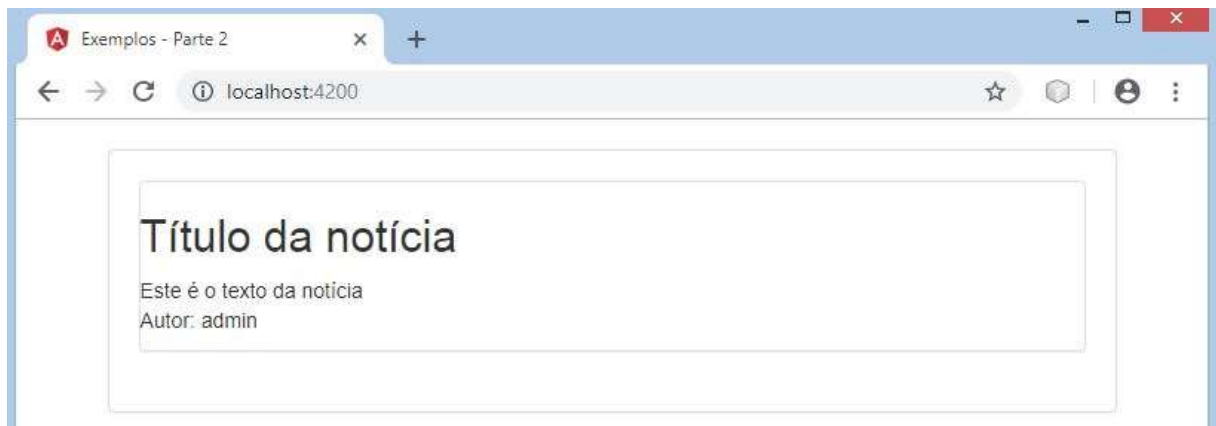
  texto: string = "Este é o texto da notícia";

}
```

### app.component.html

```
<div class="container">
  <div class="panel panel-default espaco">
    <app-noticia>
      {{texto}}
      <p>Autor: admin</p>
    </app-noticia>
  </div>
</div>
```

Resultado:



## Aula 40 – Ciclo de vida de um component

Crie um component chamado "teste":

```
ng g c teste --spec false
```

`ngOnInit()` é um método que pode executar códigos quando o componente for iniciado.

`ngOnInit()` vai representar um determinado estágio de vida do componente e só é executado quando é feita uma instanciação do componente, ou seja, neste caso quando é colocado uma diretiva do `app-teste`.

O `construtor` é executado antes do `ngOnInit()`.

### teste.component.ts

```
selector: 'app-teste',
templateUrl: './teste.component.html',
styleUrls: ['./teste.component.css']
})
export class TesteComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

## Estágios do ciclo de vida de um component

<code>ngOnInit</code>	Executado quando o componente é instanciado.
<code>ngOnChanges</code>	Executado quando uma diretiva <code>@Input</code> altera uma propriedade.
<code>ngDoCheck</code>	Executado sempre que existe alguma alteração no componente.
<code>ngAfterContentInit</code>	Executado depois da utilização de <code>ng-content</code> dentro do template.
<code>ngAfterContentChecked</code>	Executado sempre que existe uma alteração de conteúdo do <code>ng-content</code> .
<code>ngAfterViewInit</code>	Executado depois da iniciação de <code>@ChildView</code> .
<code>ngAfterViewChecked</code>	Executado sempre que acontece uma alteração via <code>@ChildView</code> .
<code>ngOnDestroy</code>	Executado quando o componente vai ser destruído.

## Aula 41 – Ciclo de vida de um component - Exemplos

Exclua o componente "teste".

### app.component.ts

```
import { Component, OnInit, DoCheck } from '@angular/core';

@Component({
  selector: 'app-teste',
  templateUrl: './teste.component.html',
  styleUrls: ['./teste.component.css']
})
export class TesteComponent implements OnInit, DoCheck {

  eventos: string[] = [];

  texto: string = '';

  constructor(){
    this.eventos.push("Constructor foi executado.");
  }

  // Acontece sempre que existe uma instanciação do component
  ngOnInit() {
    this.eventos.push("ngOnInit foi executado.");
  }

  // Acontece quando algo é alterado ou refrescado no nosso component
  ngDoCheck() {
    this.eventos.push("Aconteceram alterações.");
  }

  executar(){
    this.texto = 'alterado';
  }

}
```

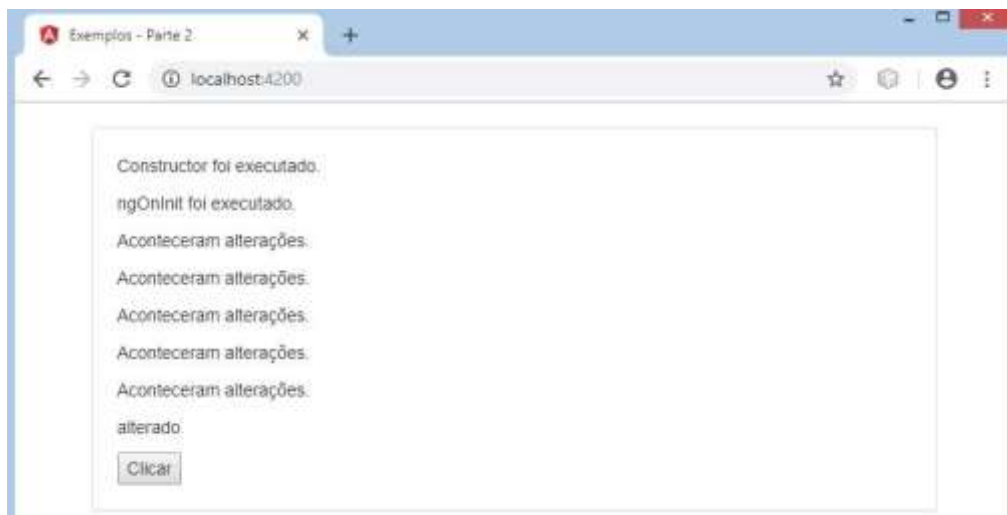
## app.component.html

```
<div class="container">
  <div class="panel panel-default espaco">
    <div *ngFor="let evento of eventos">
      <p>{{evento}}</p>
    </div>
    <p>{{texto}}</p>
    <button (click)="executar()">Clicar</button>
  </div>
</div>
```

## Resultado:



Clicando três vezes no botão:



## Aula 42 – Ciclo de vida de um component - ngOnChanges

Crie um component chamado "teste":

```
ng g c teste --spec false
```

### app.component.ts

```
import { Component, OnInit, DoCheck, OnChanges } from '@angular/core';
```

```
@Component({
  selector: 'app-teste',
  templateUrl: './teste.component.html',
  styleUrls: ['./teste.component.css']
})
export class TesteComponent implements OnInit, DoCheck, OnChanges {

  novo_valor: string = 'Novo valor';

  constructor(){}

  ngOnInit() {

  }

  ngDoCheck(){

  }

  ngOnChanges(c: SimpleChanges){
    console.log('Valor alterado');
    console.log(c);
  }

}
```

### app.component.html

```
<div class="container">
  <div class="panel panel-default espaco">

    <app-teste [valor]="novo_valor"></app-teste>

  </div>
</div>
```



Resultado:



## Aula 43 – Diretivas estruturais – Notas importantes

### app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-teste',
  templateUrl: './teste.component.html',
  styleUrls: ['./teste.component.css']
})
export class TesteComponent {

  nomes: string[] = ['João','Ana','Carlos'];

  ver: boolean = true;

}
```

### app.component.html

```
<div class="container">
  <div class="panel panel-default espaco">
    <ul class="list-group" *ngIf="ver">
      <li class="list-group-item" *ngFor="let nome of nomes">
        {{nome}}
      </li>
    </ul>

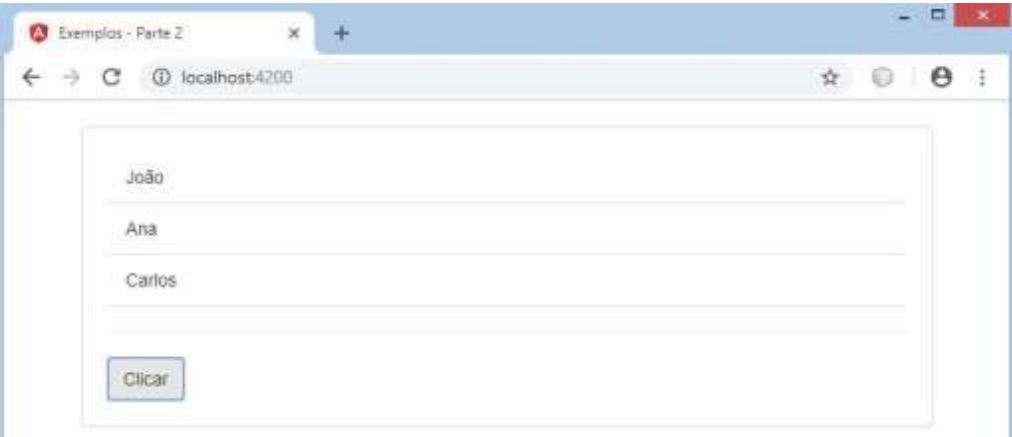
    <hr>

    <button class="btn btn-default" (click)="ver = !ver">Clicar</button>

  </div>
</div>
```

**Resultado:**

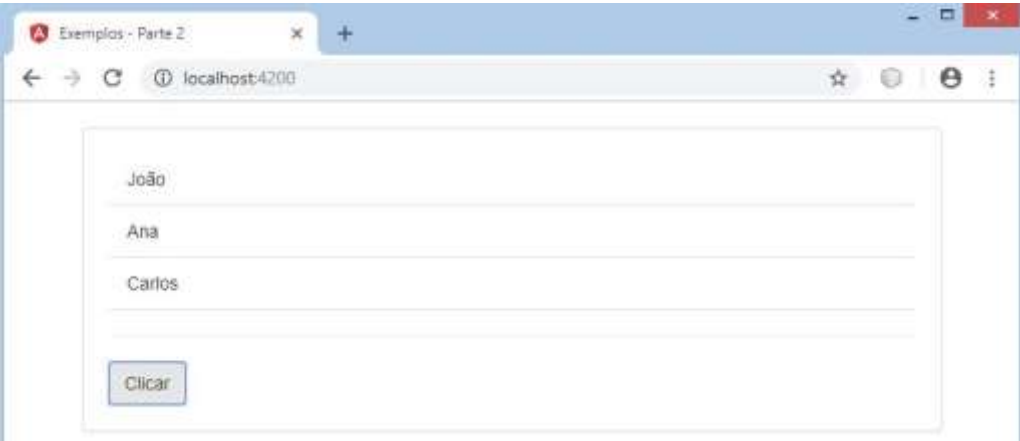
Inicial



Clicando no botão:



Clicando novamente:



## Aula 44 – Revisitando as diretivas de atributo

### app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-teste',
  templateUrl: './teste.component.html',
  styleUrls: ['./teste.component.css']
})
export class TesteComponent {

  nomes: string[] = ['João', 'Ana', 'Carlos'];

  ver: boolean = true;
  vercss: boolean = false;

}
```

### app.component.html

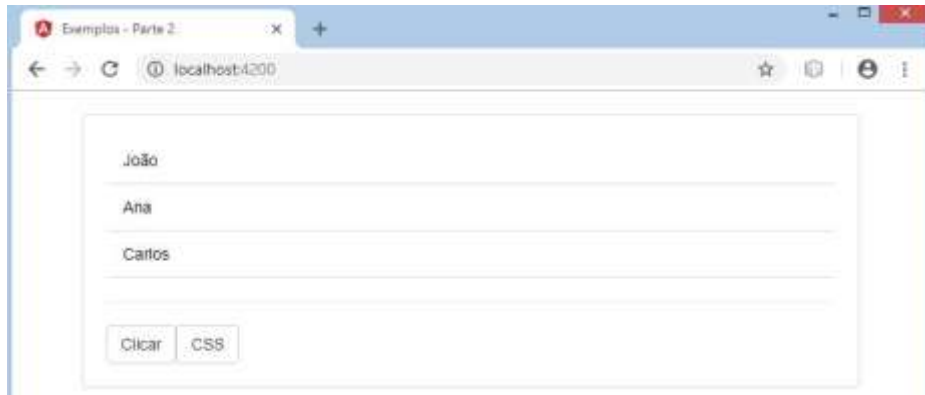
```
<div class="container">
  <div class="panel panel-default espaco">
    <ul class="list-group" *ngIf="ver">
      <li class="list-group-item"
        [ngClass]="{'cor-fundo':vercss}"
        [ngStyle]="{'color':vercss == true ? 'red' : 'black'}"
        *ngFor="let nome of nomes">
        {{nome}}
      </li>
    </ul>

    <hr>

    <button class="btn btn-default" (click)="ver = !ver">Clicar</button>
    <button class="btn btn-default" (click)="vercss = !vercss">CSS</button>
  </div>
</div>
```

## Resultado:

Inicial



Clicando no botão CSS:



Clicando no botão "Clicar"



## Aula 45 – Criando a nossa primeira Directive

Em [App](#) crie uma pasta chamada "[minhaDirective](#)".

Dentro desta pasta insira um arquivo chamado "[minhaDirective.directive.ts](#)"

### **minhaDirective.directive.ts**

```
import { Directive, ElementRef } from "@angular/core";
```

```
@Directive({  
  selector: '[minhaDirective]'  
})
```

```
export class minhaDirective{
```

```
  e:ElementRef;
```

```
  constructor(elemento: ElementRef){  
    this.e=elemento;  
    this.e.nativeElement.style.fontWeight='bold';  
  }  
}
```

Insira essa directive em [app.module.ts](#)

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { MeuComponent } from './meu/meu.component';
import { JogoComponent } from './jogo/jogo.component';
import { Comp1Component } from './comp1/comp1.component';
import { Comp2Component } from './comp2/comp2.component';
import { SociosComponent } from './socios/socios.component';
import { NoticiaComponent } from './noticia/noticia.component';
import { TesteComponent } from './teste/teste.component';
import { minhaDirective } from './minhaDirective/minhaDirective.directive';
```

```
@NgModule({
  declarations: [
    AppComponent,
    MeuComponent,
    JogoComponent,
    Comp1Component,
    Comp2Component,
    SociosComponent,
    NoticiaComponent,
    TesteComponent,
    minhaDirective
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { MeuComponent } from './meu/meu.component';
import { JogoComponent } from './jogo/jogo.component';
import { Comp1Component } from './comp1/comp1.component';
import { Comp2Component } from './comp2/comp2.component';
import { SociosComponent } from './socios/socios.component';
import { NoticiaComponent } from './noticia/noticia.component';
import { TesteComponent } from './teste/teste.component';
import { minhaDirective } from './minhaDirective/minhaDirective.directive';
```

```
@NgModule({
  declarations: [
    AppComponent,
```

```

    MeuComponent,
    JogoComponent,
    Comp1Component,
    Comp2Component,
    SociosComponent,
    NoticiaComponent,
    TesteComponent,
    minhaDirective
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

### app.component.html

```

<div class="container">
  <div class="panel panel-default espaco">
    <ul class="list-group" *ngIf="ver">
      <li class="list-group-item" [ngClass]='{"cor-fundo":vercss}' [ngStyle]='{"color":vercss == true ?
'red' : 'black'}'
        *ngFor="let nome of nomes">
          {{nome}}
        </li>
      </ul>

    <hr>

    <button class="btn btn-default" (click)="ver = !ver">Clicar</button>
    <button class="btn btn-default" (click)="vercss = !vercss">CSS</button>

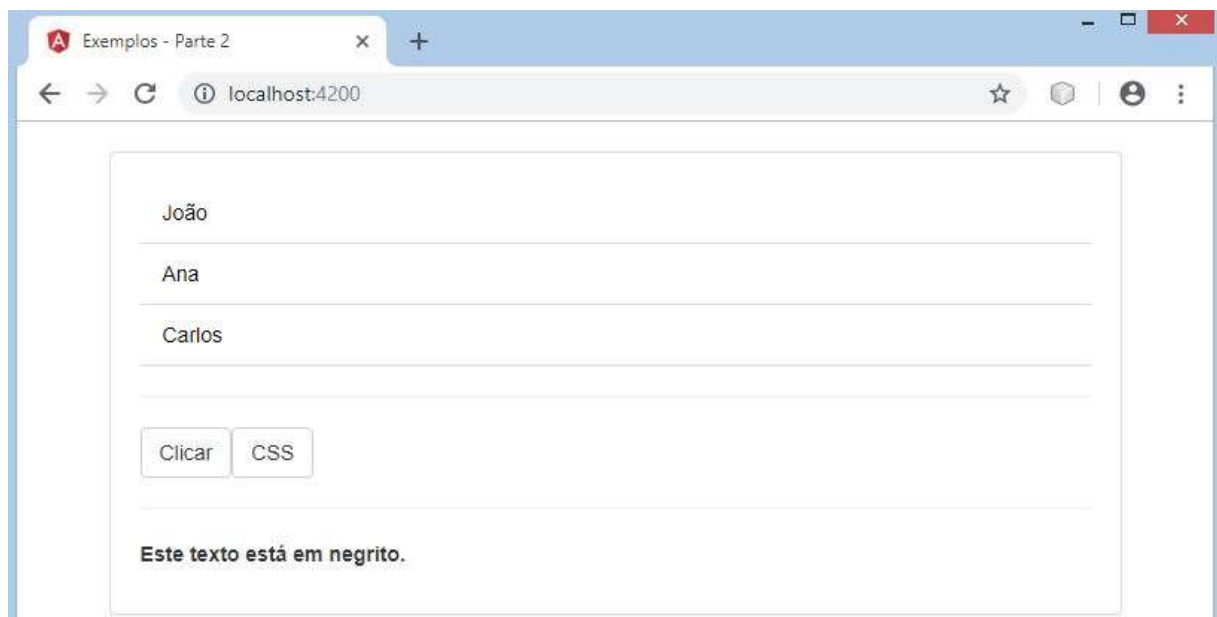
    <hr>
    <p minhaDirective>Este texto está em negrito.</p>

  </div>
</div>

```



## Resultado:



## Aula 46 – Criar Directive com CLI e uso de Renderer2

- Renomear a pasta `minhaDirective` para `Directives`.
- Fazer a alteração do nome da pasta em `app.module.ts`
- Na pasta "`Directives`" crie uma nova diretiva chamada "`novaDirective`":

`ng g d novaDirective --specs false`

### `nova-directive.directive.ts`

```
import { Directive, OnInit, ElementRef, Renderer2 } from '@angular/core';

@Directive({
  selector: '[appNovaDirective]'
})
export class NovaDirectiveDirective implements OnInit {

  constructor(private elemento: ElementRef, private render: Renderer2) {

  }

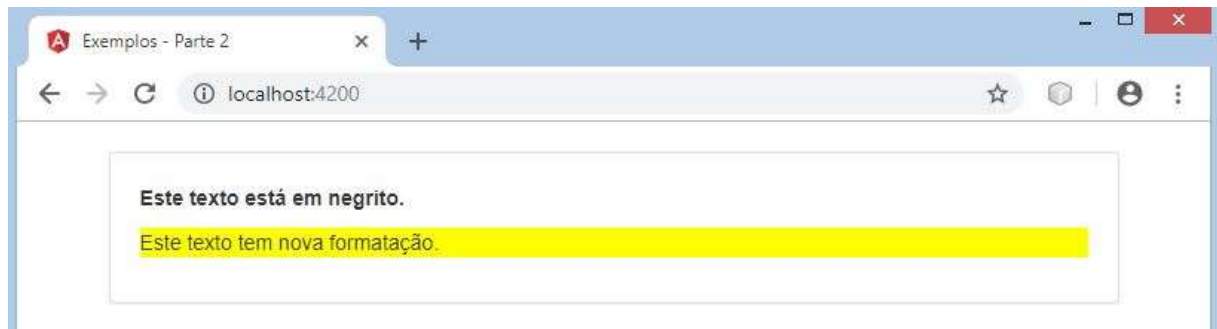
  ngOnInit() {
    this.render.setStyle(this.elemento.nativeElement, 'background', 'yellow');
  }

}
```

### `app.component.html`

```
<div class="container">
  <div class="panel panel-default espaco">
    <p minhaDirective>Este texto está em negrito.</p>
    <p appNovaDirective>Este texto tem nova formatação.</p>
  </div>
</div>
```

## Resultado:



## Aula 47 – Detetar eventos em Directives - HostListener

### nova-directive.directive.ts

```
import { Directive, OnInit, ElementRef, Renderer2 } from '@angular/core';

@Directive({
  selector: '[appNovaDirective]'
})
export class NovaDirectiveDirective implements OnInit {

  constructor(private elemento: ElementRef, private render: Renderer2) {}

  ngOnInit() {
  }

  @HostListener('mouseenter') m1() {
    this.render.setStyle(this.elemento.nativeElement, 'background', 'red');
  }

  @HostListener('mouseleave') m2() {
    this.render.setStyle(this.elemento.nativeElement, 'background', 'transparent');
  }

  @HostListener('click') m3() {
    this.render.setStyle(this.elemento.nativeElement, 'background', 'green');
  }
}
```

### app.component.html

```
<div class="container">
  <div class="panel panel-default espaco">
    <p minhaDirective>Este texto está em negrito.</p>
    <p class='pad' appNovaDirective>Este texto tem nova formatação.</p>
  </div>
</div>
```

### app.component.css

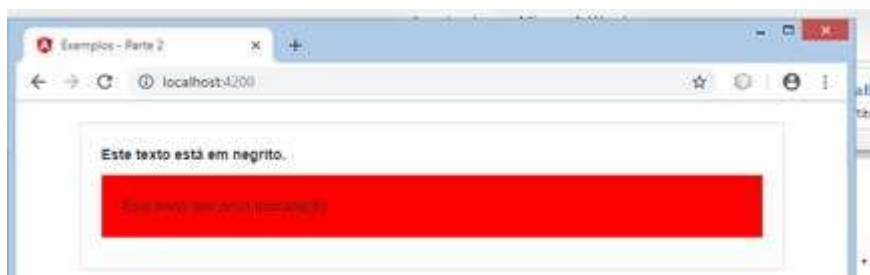
```
.espaco{  
  margin: 20px;  
  padding: 20px;  
}  
  
.cor-fundo{  
  background-color: bisque;  
}  
  
.pad{  
  padding: 20px;  
}
```

### Resultado:

Inicial



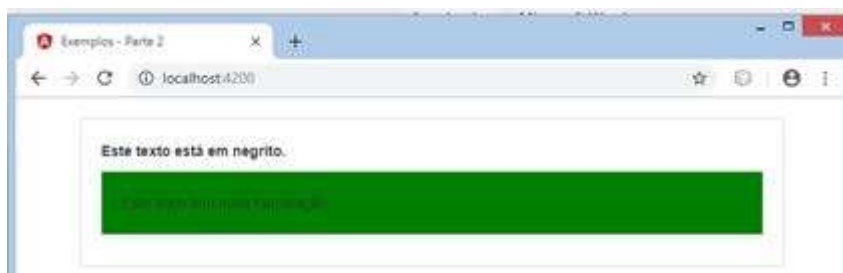
Passando o mouse sobre o segundo parágrafo:



Tirando o mouse do segundo parágrafo



Clicando o mouse



## Aula 48 – HostListener e HostBinding

**nova-directive.directive.ts**

```
import { Directive, OnInit, ElementRef, Renderer2 } from '@angular/core';

@Directive({
  selector: '[appNovaDirective]'
})
export class NovaDirectiveDirective implements OnInit {

  constructor(private elemento: ElementRef, private render: Renderer2) {}

  ngOnInit() {
  }

  constructor(private elemento: ElementRef, private render: Renderer2) {}

  ngOnInit() {
  }

  /* @HostBinding('style.backgroundColor') corFundo: string = 'red';

  @HostListener('mouseenter') m1() {
    this.corFundo = 'yellow';
  }

  @HostListener('mouseleave') m2() {
    this.corFundo = 'red';
  }
  */

  @HostBinding('value') texto: string = '';

  @HostListener('mouseenter') colocarTexto() {
    this.texto = 'Entre no input';
  }

  @HostListener('mouseleave') removerTexto() {
    this.texto = '';
  }
}
```

## app.component.html

```
<div class="container">  
  <div class="panel panel-default espaco">  
    <p minhaDirective>Este texto está em negrito.</p>  
    <p class='pad'>Este texto tem nova formatação.</p>  
    <input type="text" appNovaDirective />  
  </div>  
</div>
```

## Resultado:

Inicial



Passando o mouse sobre a caixa de texto:



Tirando o mouse da caixa de texto:

