

# Projeto de Interfaces WEB

## Módulos do Angular Aula 03

# Diretivas Personalizadas

- Volte ao estado “clean” do seu projeto:

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

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

```
export class AppComponent {  
  nome = "Jefferson de Carvalho";
```

```
  salvar(nomeInput:string){  
    this.nome = nomeInput;  
  }  
}
```

```
<div class="container">
```

```
  <div class="alert alert-primary" role="alert">  
    Seu nome é {{nome}}  
  </div>
```

```
  <div class="form-group">  
    <label>Nome</label>  
    <input type="text" class="form-control"  
#nomeInput>  
  </div>
```

```
  <button type="button" class="btn btn-primary"  
(click)="salvar(nomeInput.value)">Salvar</button>  
</div>
```

# Introdução

- Um módulo em Angular, é uma forma de agrupar componentes, semelhante aos pacotes Java.

APPMODULE

COMPONENTE 1

COMPONENTE 2

DIRETIVA 1

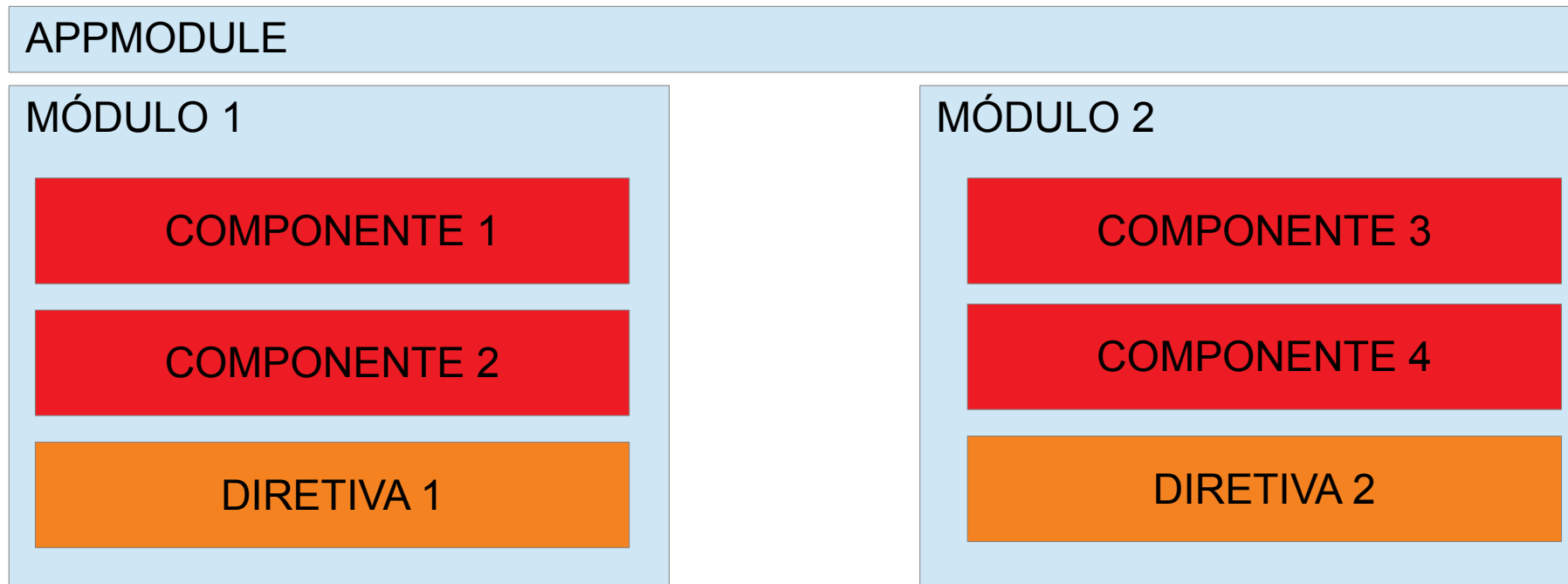
COMPONENTE 3

COMPONENTE 4

DIRETIVA 2

# Introdução

- Podemos organizar nossa aplicação:

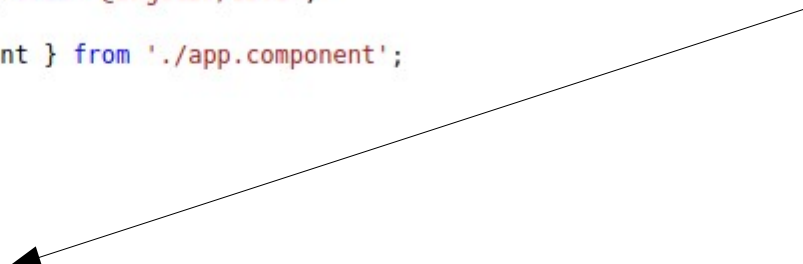


# Criando um Módulo

- É a partir do AppModule que toda a aplicação inicia.

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

Novos módulos são importados aqui.

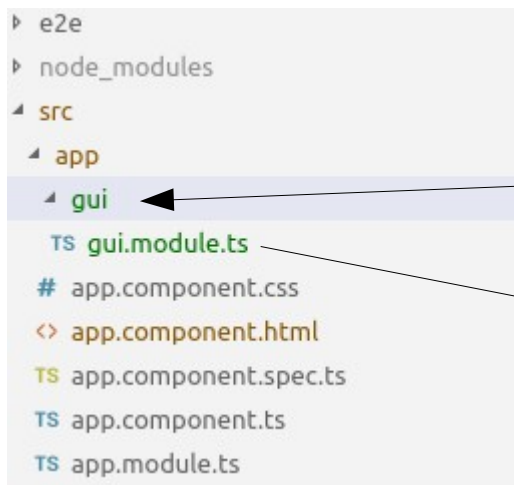


# Criando um Módulo

- Objetivo: vamos criar módulos com componentes simples, apenas para testar a organização da aplicação.
- Execute o comando:
  - `ng g m gui`
- “gui” é apenas um nome qualquer para um módulo onde iremos colocar componentes de interface gráfica (graphical user interface-gui)

# Criando um Módulo

- Veja que foi criado uma pasta “gui”, dentro do seu projeto.



Novo módulo criado.

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';

@NgModule({
  declarations: [],
  imports: [
    CommonModule
  ]
})
export class GuiModule { }
```

# Criar um Componente no Módulo

- Execute o comando:
  - `ng g c gui/header`
- Mais uma vez, “header” é apenas um nome para o componente que queremos salvar dentro da pasta gui.

```
└─ gui
  └─ header
     # header.component.css
     <> header.component.html
     TS header.component.spec.ts
     TS header.component.ts
```



# Criar um Componente no Módulo

- Coloque o seguinte código dentro de header.html:

```
<div class="jumbotron">
  <h1 class="display-4">Hello, {{nome}}</h1>
  <p class="lead">This is a simple hero unit, a simple jumbotron-style component
for calling extra attention to featured content or information.</p>
  <hr class="my-4">
  <p>It uses utility classes for typography and spacing to space content out within
the larger container.</p>
  <a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a>
</div>
```

# Criar um Componente no Módulo

- Coloque o seguinte código dentro de header.ts:

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

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

```
  @Input() nome:string;
```

```
  constructor(){  
    this.nome = "Jefferson";  
  }  
}
```

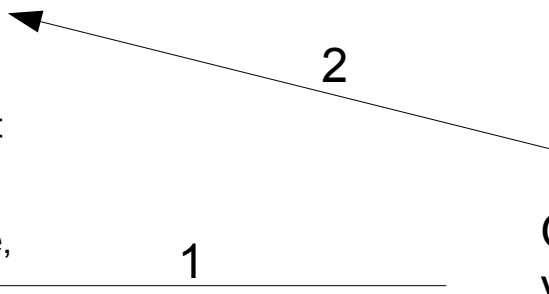
# Criar um Componente no Módulo

- Importe, em AppModule, o módulo criado.

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

```
import { AppComponent } from './app.component';  
import { GuiModule } from './gui/gui.module';
```

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



O import é feito automaticamente, assim que você adiciona o GuiModule em imports.

# Criar um Componente no Módulo

- Em app.componente.html, chame o componente header.

```
<div class="container">  
  <app-header nome="Fulano de Tal"></app-header>  
</div>
```

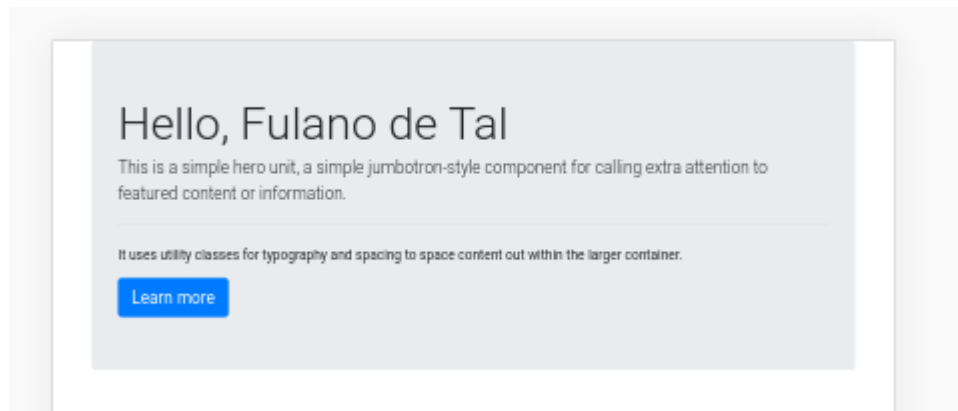
- Note que está dando, erro. Qual é o problema?
- Apesar de AppModule ter importado o GuiModule, o próprio GuiModule não **exporta** o componente app-header.
- Sendo assim, o componente app-header só pode ser usado dentro do módulo em que foi criado.

# Criar um Componente no Módulo

- Solução: exportar o componente header, em GuiModule.

```
import { NgModule } from '@angular/core';  
import { CommonModule } from '@angular/common';  
import { HeaderComponent } from '../header/header.component';
```

```
@NgModule({  
  declarations: [HeaderComponent],  
  exports: [HeaderComponent],  
  imports: [  
    CommonModule  
  ]  
})  
export class GuiModule { }
```

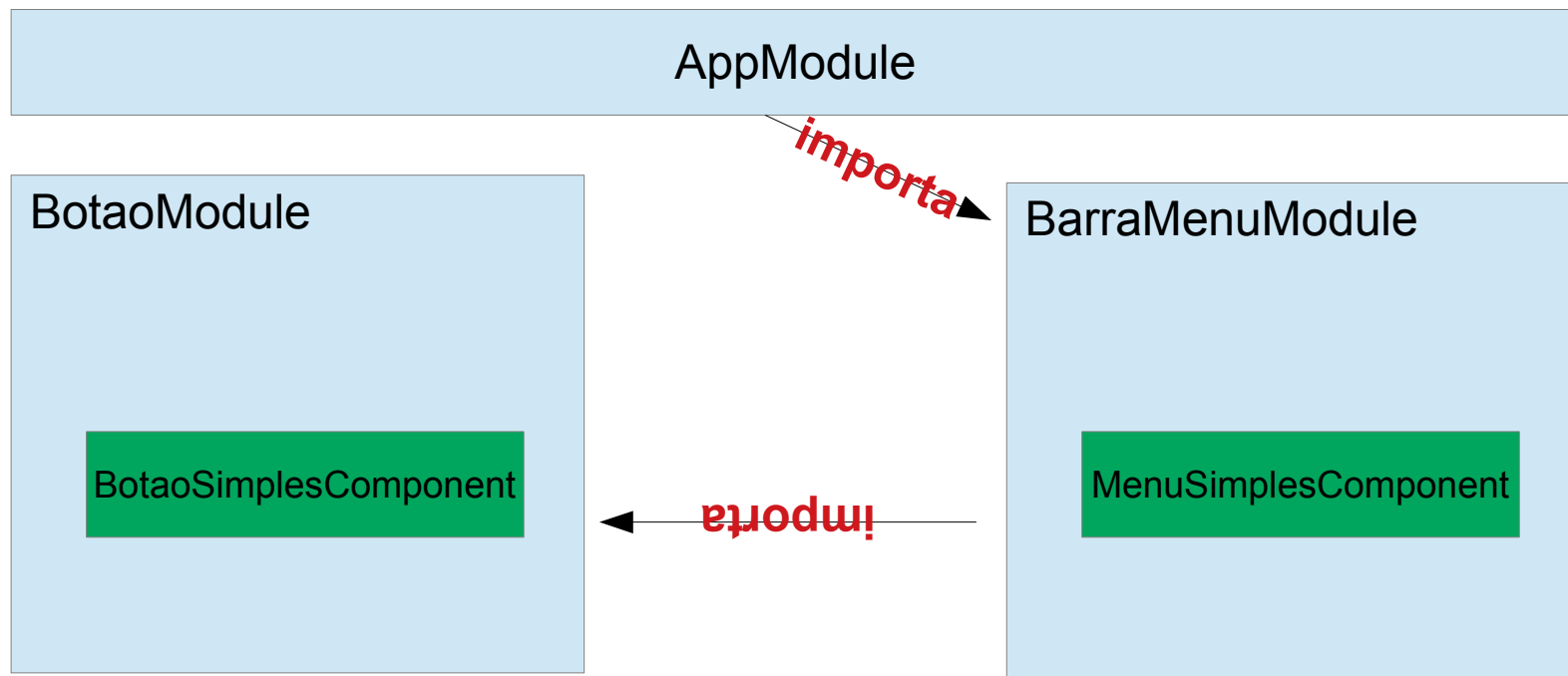


# Criar um Componente no Módulo

- Conclusão
  - Criar um novo módulo (gui)
  - Criar um novo componente (header) dentro do módulo
  - Importar o novo módulo no “módulo-pai” que, no nosso caso, é o AppModule
  - Exportar o novo componente para que o mesmo possa ser usado em app.component.html

# Exportação de Módulos Aninhados

- Crie a seguinte aplicação:



# Exportação de Módulos Aninhados

- Comandos:
  - `ng g m botao`
  - `ng g c botao/botao-simples --spec=false`
  - `ng g m barra-menu`
  - `ng g c barra-menu/menu-simples --spec=false`



# Exportação de Módulos Aninhados

- botao-simples

## .ts

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

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

  @Input() label: string;
}
```

## .html

```
<button class="btn btn-primary">{{label}}</button>
```

## .module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { BotaoSimplesComponent } from './botao-simples/
botao-simples.component';

@NgModule({
  declarations: [BotaoSimplesComponent],
  exports: [BotaoSimplesComponent],
  imports: [
    CommonModule
  ]
})
export class BotaoModule { }
```

# Exportação de Módulos Aninhados

- menu-simples

## .ts

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

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

## .html

```
<app-botao-simples label="SALVAR"></app-botao-simples>
&nbsp;
<app-botao-simples label="CANCELAR"></app-botao-simples>
```

## .module.ts

```
import { BotaoModule } from '../botao/botao.module';
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { MenuSimplesComponent } from './menu-simples/
menu-simples.component';

@NgModule({
  declarations: [MenuSimplesComponent],
  exports: [MenuSimplesComponent],
  imports: [
    CommonModule,
    BotaoModule
  ]
})
export class BarraMenuModule { }
```

# Exportação de Módulos Aninhados

- app-component

.html

```
<div class="container">  
  <app-menu-simples></app-menu-simples>  
</div>
```

.module.ts

```
import { BarraMenuModule } from './barra-menu/barra-menu.module';  
import { BrowserModule } from '@angular/platform-browser';  
import { NgModule } from '@angular/core';  
  
import { AppComponent } from './app.component';  
  
@NgModule({  
  declarations: [  
    AppComponent  
  ],  
  imports: [  
    BrowserModule,  
    BarraMenuModule  
  ],  
  providers: [],  
  bootstrap: [AppComponent]  
})  
export class AppModule { }
```

# Exportação de Módulos Aninhados

- Mas, e se você quiser usar um botao-simples em app.component?

```
<div class="container">  
  <app-menu-simples></app-menu-simples>  
  <hr>  
  <app-botao-simples [label]="LIMPAR"></app-botao-simples>  
</div>
```

- App.component não tem acesso a esse seletor de botao!
- Como resolver?

# Exportação de Módulos Aninhados

- Solução:
  - Ora, se barra-menu.module importa o módulo botao.module, basta que ele, além de exportar o componente menu-simples, exporte também o **componente botao-simples**.

**barra-menu.module.ts**

```
@NgModule({  
  declarations: [MenuSimplesComponent],  
  exports: [MenuSimplesComponent, BotaoSimplesComponent],  
  imports: [  
    CommonModule,  
    BotaoModule  
  ]  
})  
export class BarraMenuModule { }
```

