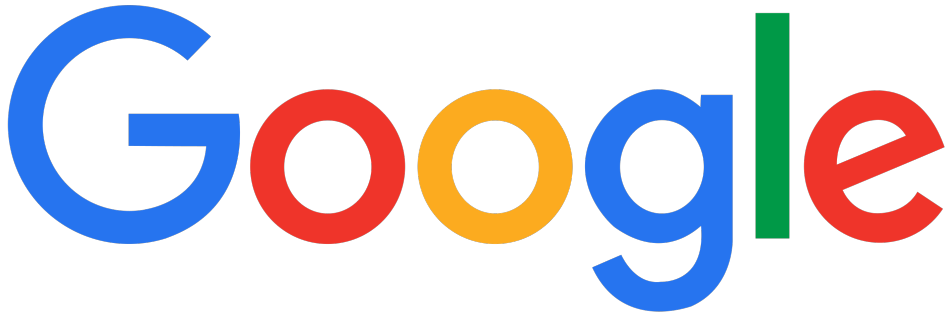
Angular Framework

Visão geral

Prof. Fernando Luis - fernando.oliveira@iffarroupilha.edu.br

Introdução

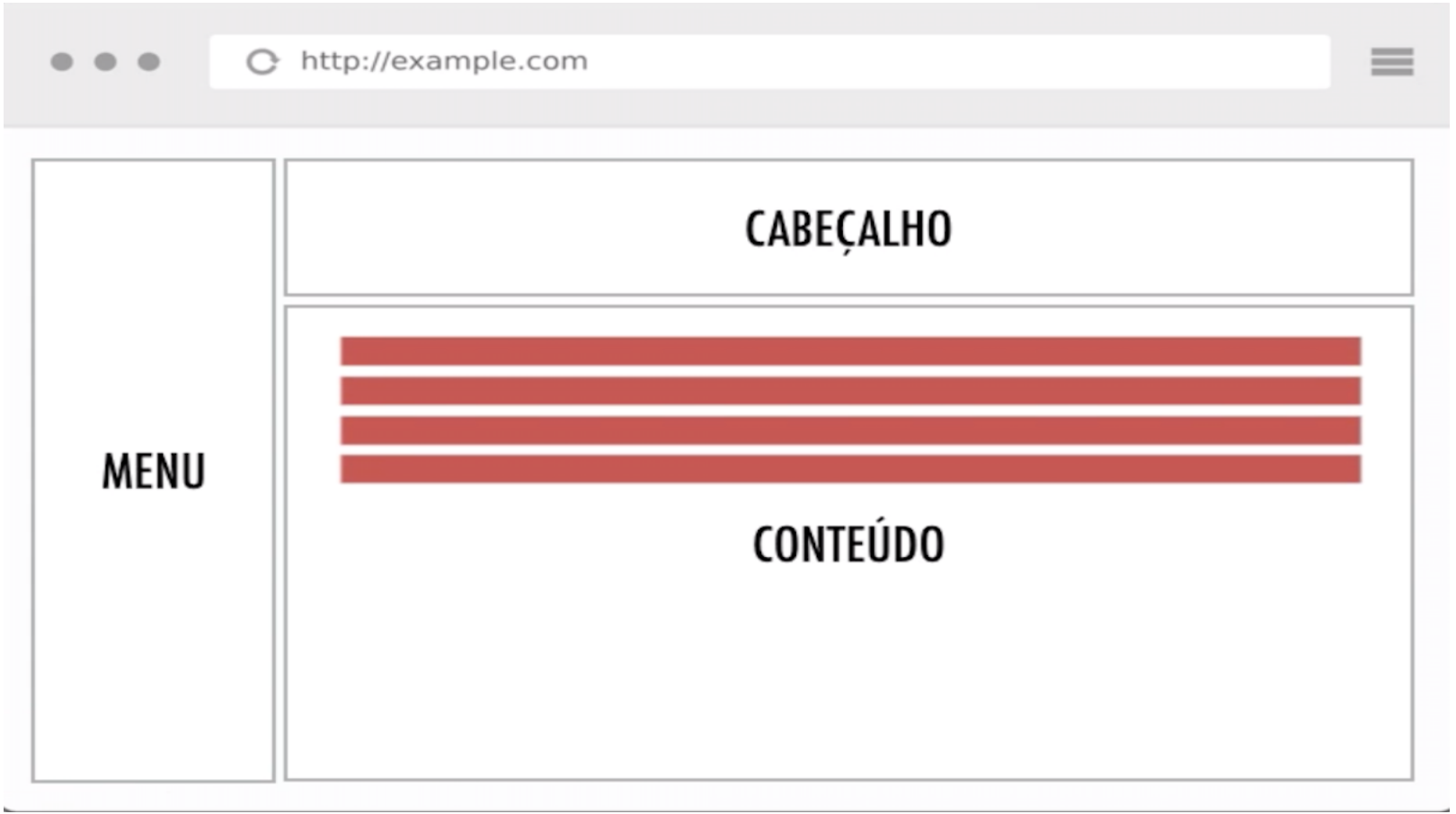
- Angular é um framework JavaScript open-source que auxilia na execução de single-page applications.



- Construído sob a ideologia de que programação declarativa deve ser usada para construção de Interfaces de Usuário e componentes de software, enquanto que a programação imperativa é indicada para escrever as regras de negócio

Introdução - SPA

Introdução - SPA



Introdução

A programação declarativa define "O que ao invés de como"

Ex:

- Os produtos devem ser exibidos nesse local;
- Se esse botão for clicado chame essa função.

Introdução

Com Agular os programadores podem concentrar seus esforços nas regras de negócio, tirando todo proveito das vantagens do ECMAScript 6 e da orientação a objetos.

Introdução

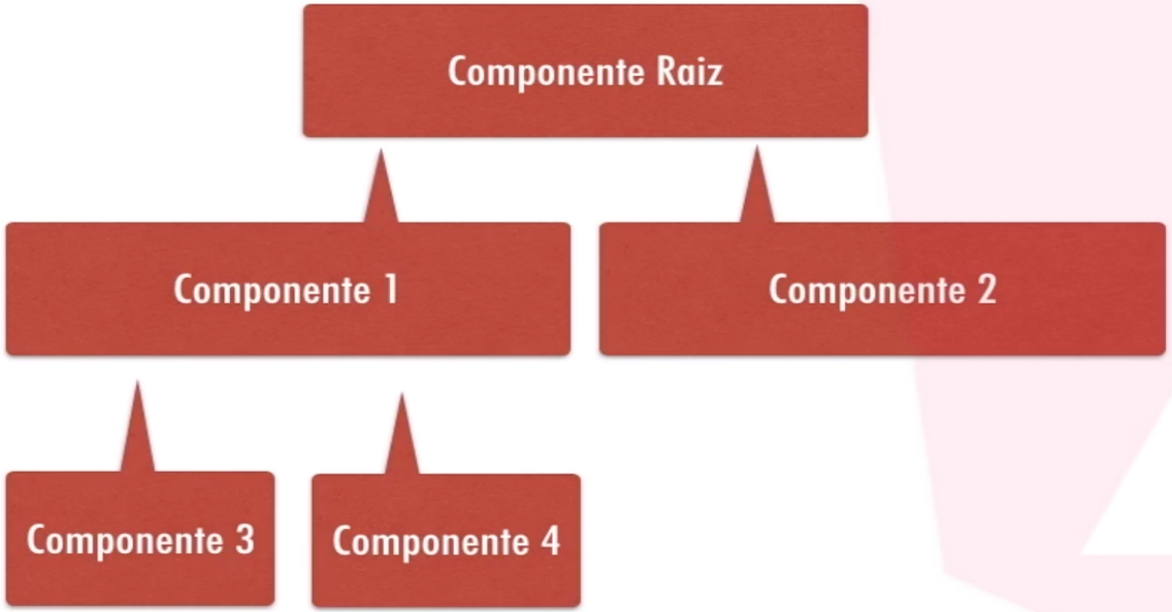
O HTML é ótimo para declarar documentos estáticos, contudo, não dá suporte para declarar visualizações de forma dinâmicas em aplicações web. O Angular permite ampliar o vocabulário HTML, o ambiente resultante é um modelo híbrido, legível e rápido para se desenvolver.

Componente raiz

O angular é orientado a componentes, isso significa que você vai escrever diversos componentes minúsculos que juntos constituirão uma aplicação inteira. Um Component é a combinação de um template HTML com uma classe que controla parte da tela.

O componente principal da aplicação é o *app.component.ts*, no qual definimos o componente raiz (root component), que se tornará uma árvore de componentes aninhados conforme a aplicação for evoluindo.

Componente raiz



Componente raiz

Componente raiz

```
1  import { Component } from '@angular/core'
2
3  @Component({
4       selector: 'app-root',
5       template: 'Inicializando uma aplicação Angular'
6  })
7  export class AppComponent { }
```

O código acima refere-se a uma classe em typescript, acrescida do termo export, o qual determina que a classe poderá ser utilizada em outros pontos do código.

Componente raiz

```
1  import { Component } from '@angular/core'
2
3  @Component({
4        selector: 'app-root',
5        template: 'Inicializando uma aplicação Angular'
6  })
7  export class AppComponent { }
```

O @Component representa um padrão de projeto denominado decorator, sendo utilizado para passagem de parâmetros.
Neste exemplo configura-se o "selector" que define a tag utilizada no documento HTML e o "template" responsável pelo conteúdo a ser renderizado.

Módulo raiz

O modulo raiz é o módulo principal da aplicação, responsável por dizer ao Angular como montar a aplicação e identifica os recursos que estão acessíveis.

Todos elementos criados dentro do projeto (páginas, componentes, filtros entre outros) precisão ser declarados neste módulo, do contrário o Angular lançará um erro de execução ao tentar acessar o recurso.

Módulo raiz

```
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } '@angular/core';
3  import { AppComponent } from './app.component'
4
5  @NgModule({
6       imports: [BrowserModule],
7       declarations: [AppComponent],
8       bootstrap: [AppComponent]
9  })
10 export class AppModule {}
```

O decorator @NgModule é utilizado para definir os elementos que fazem parte da aplicação e recursos utilizados.

A definição do módulo raiz é dado através do arquivo "*main.js*"

Diretivas

Diretivas

Diretivas são como "extensões" da linguagem HTML que permitem a implementação de novos comportamentos de forma declarativa.
Na prática você não vai precisar escrever várias linhas de código para executar funções que deveriam ser simples, como por exemplo, listar uma coleção de itens (produtos, clientes etc) em tela.

Diretivas de interface

Algumas diretivas são destinadas a manipulação de templates de página ou html. Toda vez que você ver uma diretiva marcada com o * (asterisco) antes, significa que essa é uma diretiva de template, exemplos:

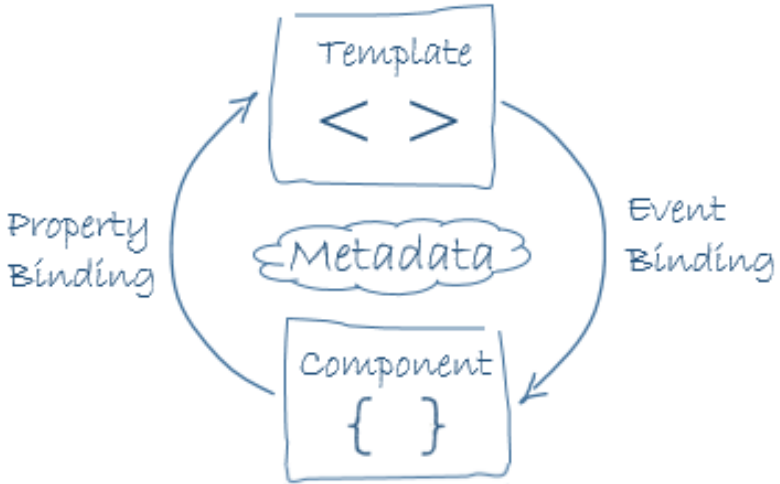
- `*ngFor="let item in items"`
- `*ngIf="model.hasAccess"`
- `*ngShow="model.canShow"`

Data binding

Data binding

O binding representa a conexão entre os componentes, propriedades e eventos da tela para com o controlador (ts).
A direção da ligação pode ser unidirecional, da tela para o controle ou do controle para tela, ou biderecional (Two Way Data Binding) ocorrendo nos dois sentidos.
O binding é configurado por caracteres:

- [ ]
- ( )
- {{ }}
- ([ ])

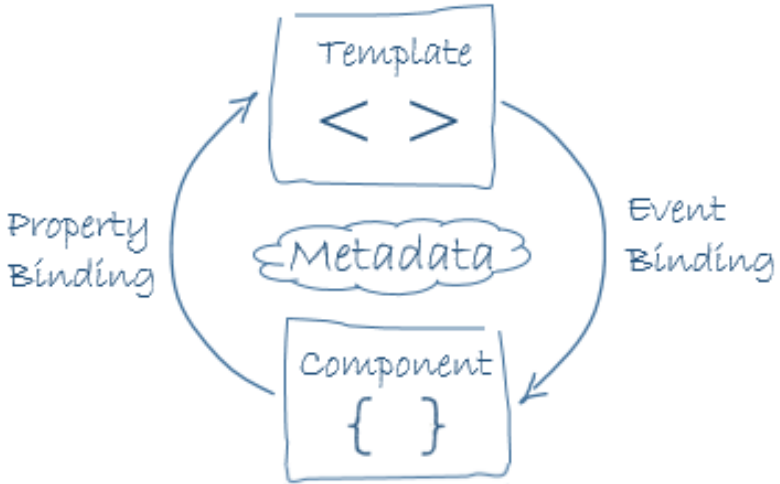


Binding []

O uso do par *[]* atribui o resultado da expressão para: a) uma propriedade do componente (se houver), ou b) uma propriedade do elemento HTML.

Quando atribuído a um atributo, significa que o bind está sendo feito da fonte de dados (ts) para a view. De maneira unidirecional, ou seja, se modificar na view, não refletira no modelo.

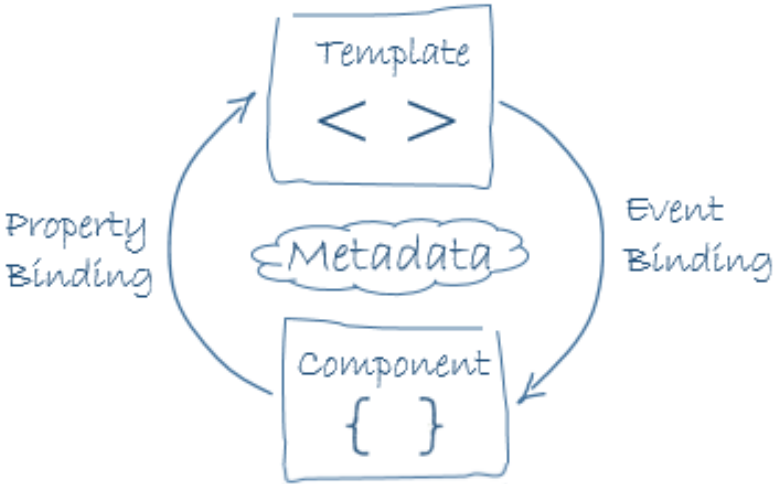
```
<img [src]="model.image" />
```



Binding ()

Quando utilizado, o () indica que o bind está sendo feito da view para a fonte de dados. Sempre de maneira unidirecional. Por exemplo

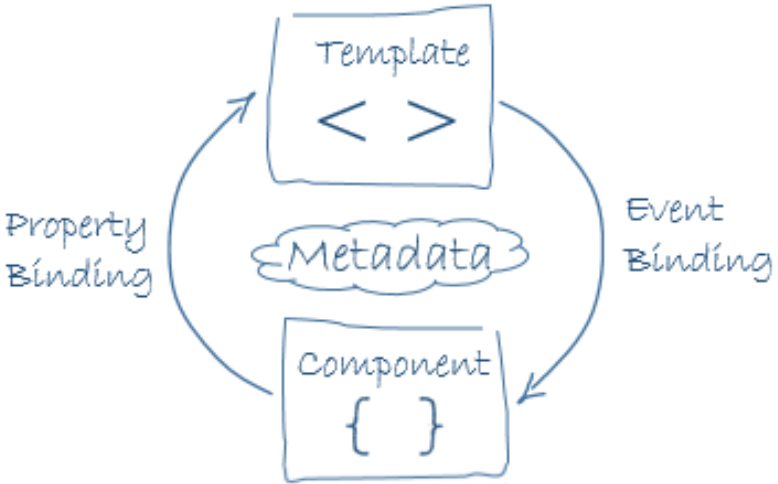
```
<button (click)="doSomething()">Botão</button>
```



Binding {{}}

O bind {{}} é unidirecional da fonte de dados para a view, uma alternativa ao [] usado principalmente para dar bind em textos. Por exemplo:

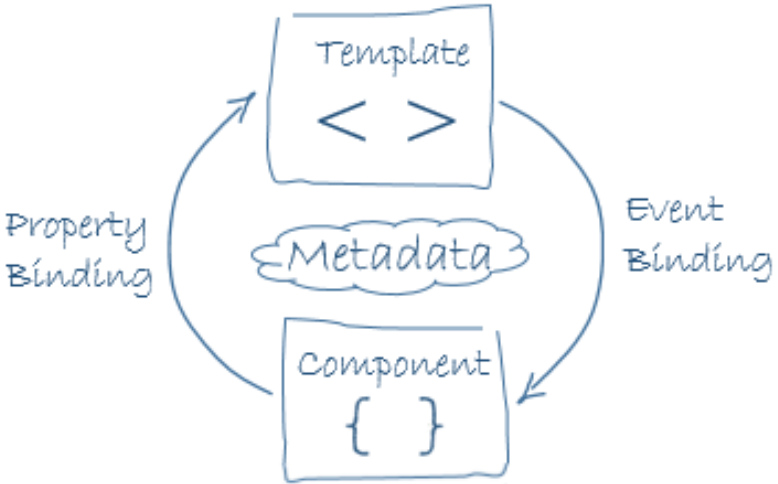
```
<p>{{model.text}}</p>
```



Binding [()]

O bind conbinado [()] é aplicado em conjunto com a diretiva ngModel, com ela temos a ligação bidirecional (two-way data binding)

```
<input type="text" [(ngModel)]="model.name" />
```



Exemplos práticos