



Desenvolvimento de Aplicações WEB
Angular JS

Profa. Joyce Miranda

AngularJS

- ▶ Esse material foi criado com base no curso oferecido pela TimTec, acessível através do link abaixo:



<https://cursos.timtec.com.br/course/desenvolvimento-web-com-angularjs/intro>

AngularJS

- ▶ O que é?
 - ▶ Framework de desenvolvimento de aplicações web mantido pela Google.
 - ▶ Visa o aumento de produtividade e a diminuição do esforço na implementação.
- ▶ Por que estudar?
 - ▶ Framework atual e com crescente aceitação no mercado.



AngularJS

► Pré-Requisitos



► Organização do Curso

Introdução

Conceitos Básicos

Services

Routing

Custom Directives

AngularJS

- ▶ Download/Instalação



<https://angularjs.org/>

AngularJS

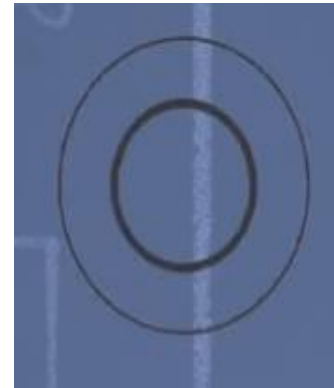
- ▶ **Característica**
 - ▶ Utiliza atributos HTML específicos que são capazes de adicionar comportamento às páginas WEB.
- ▶ **Como funciona?**



**Lendo
HTML**



**Identificando
Atributos**

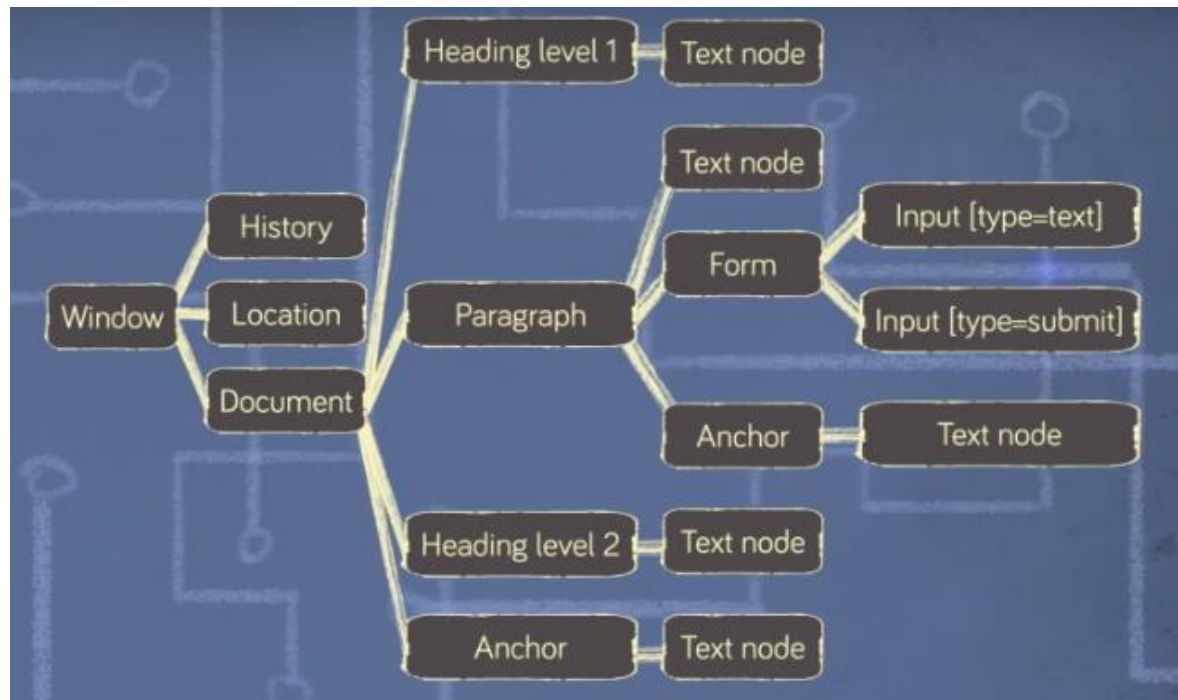


**Aplicando
Comportamento**

AngularJS

► Característica

- Realiza a análise, a compreensão e a manipulação da árvore DOM por meio de atributos específicos



Árvore DOM

AngularJS

▶ Característica

- ▶ Desacoplamento total entre *front-end* e *back-end*.

▶ Como funciona?

▶ **Front-end**

- ▶ Responsável por toda a geração e exibição do HTML.

▶ **Back-end**

- ▶ Envia os dados para o *front-end* através de serviços HTTP na forma de arquivos JSON.

▶ Vantagem

- ▶ Páginas responsivas e rápidas, tendo em vista que todo o processamento do HTML é feito na máquina do cliente

WEB

AngularJS

► Quiz

○ AngularJS é um framework que...

☐ É implementado em Java

☐ Estende o HTML

☐ Utiliza NodeJS

☐ É derivado do JSF

☐ Estende o PHP

WEB

AngularJS

► Quiz

Marque Verdadeiro ou Falso:

	Verdadeiro	Falso
O AngularJS Trabalha com HTML gerado pelo Backend	<input type="radio"/>	<input type="radio"/>
O AngularJS le o html de uma página e procura seus atributos	<input type="radio"/>	<input type="radio"/>
O AngularJS trabalha diretamente com HTML, CSS e Javascript	<input type="radio"/>	<input type="radio"/>

AngularJS

▶ Característica

▶ **SPA – *Single Page Application***

- ▶ Todo código HTML, JS e CSS é carregado em um único *page load*.
- ▶ A página não recarrega em nenhum momento do processo.
- ▶ O controle não é transferido a outras páginas.

▶ Como funciona?

- ▶ Sensação de troca de página com *PUSH STATE*
 - Função JS que serve para mudar a url do *browser* sem dar *refresh*.

▶ Vantagem

- ▶ Experiência de navegação mais fluida e dinâmica.
- ▶ Diminuição da comunicação com o servidor.

► Quiz

Marque Verdadeiro ou Falso:

	Verdadeiro	Falso
SPA significa Single-Page Application	<input type="radio"/>	<input type="radio"/>
Aplicações SPA só possuem 1 arquivo HTML	<input type="radio"/>	<input type="radio"/>
Aplicações SPA utilizam javascript para simular a troca de página	<input type="radio"/>	<input type="radio"/>

AngularJS

► Característica

► MVC – *Model View Controller*



► Quiz

Relacione os conceitos de MVC:

Coluna 1

Coluna 2

1. Model

Modelos de Dados e Funções

2. View

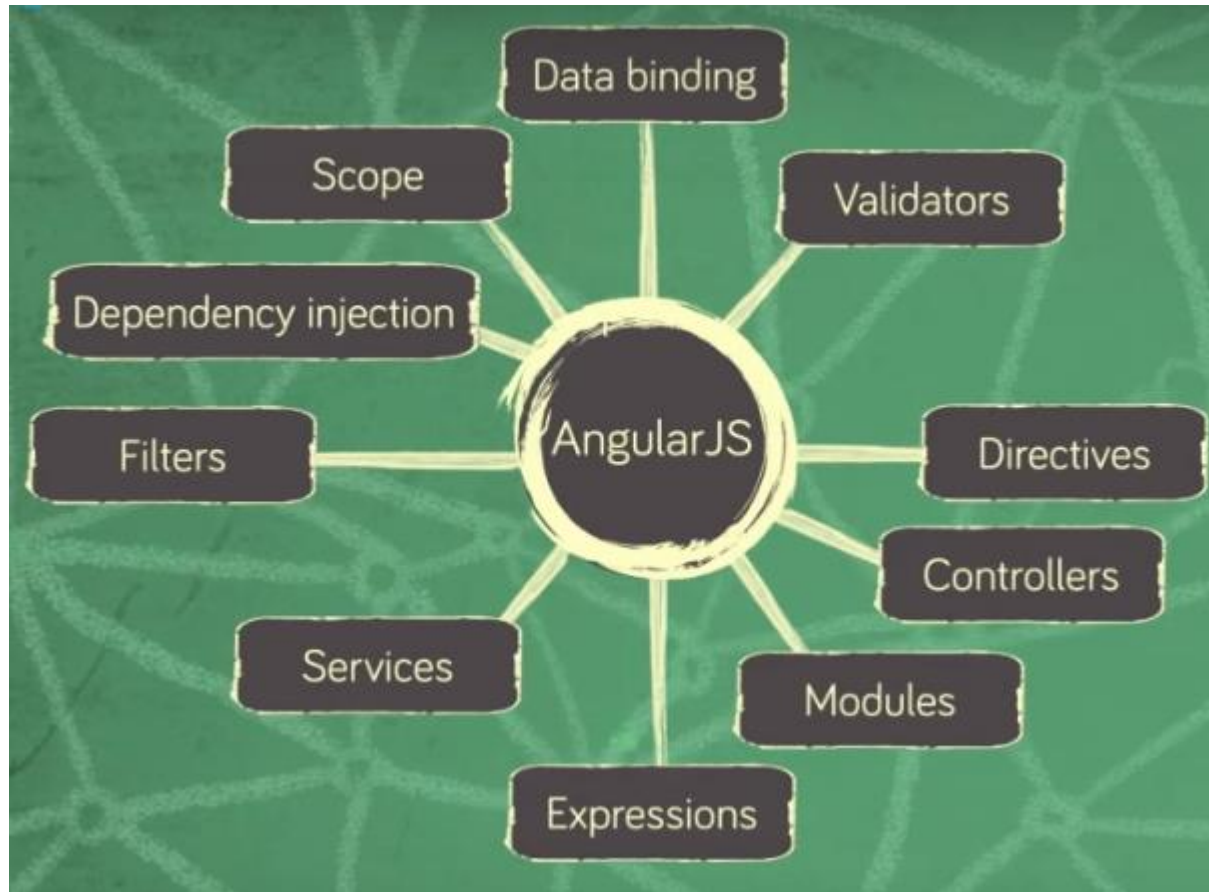
Mediador entre modelos e suas
exibições

3. Controller

Representações para exibição dos
dados

AngularJS

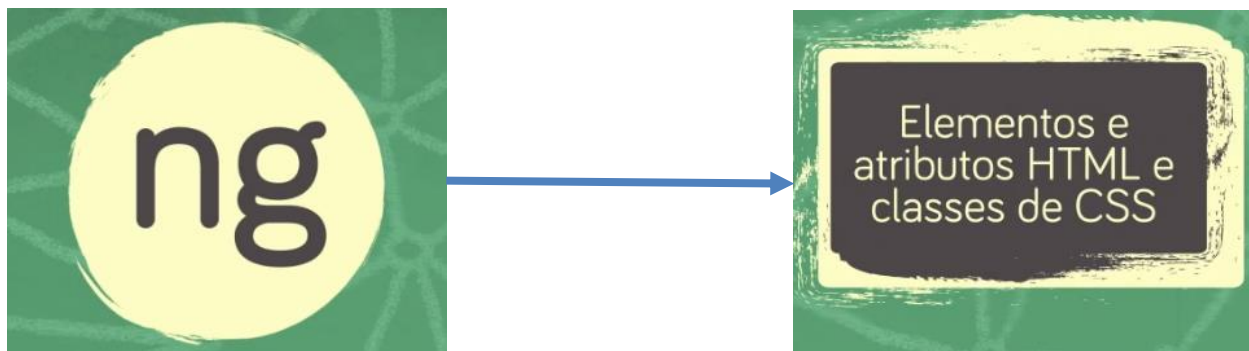
► Características Específicas



AngularJS

► Directives (Diretivas)

- Extensões de atributo HTML com o prefixo **ng-**.
 - São aplicadas em elementos da árvore DOM da página.



- Indicam ao compilador de HTML do AngularJS que ele deve aplicar um comportamento específico a um determinado elemento.

AngularJS

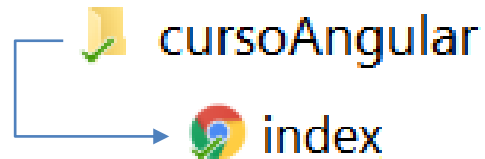
► Directives: **ng-app**

- Define o elemento raiz que irá conter toda a sua aplicação
 - Todos os *controllers* e *models*
- Indica qual aplicação deve ser executada no momento que uma página é acessada.



AngularJS

- ▶ *Aplicando **ng-app***
 - ▶ **Crie a pasta do projeto**
 - ▶ “cursoAngular”
 - ▶ **Crie a página HTML**
 - ▶ “index.html”



```
<html>
  <head>
    <meta charset="utf-8">
    <!-- metas de segurança -->
    <meta http-equiv="X-UA-Compatible" content="IE-edge">
    <meta http-equiv="Content-type" content="text/html">
    <!-- referência bootstrap -->
    <link rel="stylesheet" href="css/bootstrap.css">
  </head>
  <body class="container-fluid">
    <header>
      <h1 style="text-align: center; color: blue">
        Bem vindo ao Curso de Angular!
      </h1>
    </header>
    <!-- referência angular -->
    <script type="text/javascript" src="js/libs/angular.js"></script>
  </body>
</html>
```

Tags de Segurança para a permissão
de alguns comportamentos javascript
pelo navegador

```
<html>
```

index.html

```
<head>
```

```
<meta charset="utf-8">
```

```
<!--metas de segurança -->
```

```
<meta http-equiv="X-UA-Compatible" content="IE-edge">
```

```
<meta http-equiv="Content-type" content="text/html">
```

```
<!--referência bootstrap -->
```

```
<link rel="stylesheet" href="css/bootstrap.css">
```

```
</head>
```

```
<body class="container-fluid">
```

Obtenha o arquivo na página oficial do
CSS: <https://getbootstrap.com/>

```
<header>
```

```
<h1 style="text-align: center; color: blue">
```

```
  Bem vindo ao Curso de Angular!
```

```
</h1>
```

```
</header>
```

```
<!--referência angular -->
```

```
<script type="text/javascript" src="js/libs/angular.js"></script>
```

```
</body>
```

```
</html>
```

```
<html>
```

index.html

```
<head>
```

```
<meta charset="utf-8">
```

```
<!--metas de segurança -->
```

```
<meta http-equiv="X-UA-Compatible" content="IE-edge">
```

```
<meta http-equiv="Content-type" content="text/html">
```

```
<!--referência bootstrap -->
```

```
<link rel="stylesheet" href="css/bootstrap.css">
```

```
</head>
```

```
<body class="container-fluid">
```

```
<header>
```

```
<h1 style="text-align: center; color: blue">
```

Bem vindo ao Curso de Angular!

```
</h1>
```

```
</header>
```

Crie as pastas *js/libs* no seu projeto e salve o arquivo baixado do site oficial do AngularJS: <https://angularjs.org/>

```
<!--referência angular -->
```

```
<script type="text/javascript" src="js/libs/angular.js"></script>
```

```
</body>
```

```
</html>
```

Angular JS

▶ Aplicando **ng-app**

▶ Definir uma **app**

- ▶ Criar o arquivo **“app.js”**



```
app.js
var app = angular.module('aplicacao', []);
```


Módulo define uma aplicação no AngularJS.

[] -> Recebe as dependências da aplicação

Angular JS

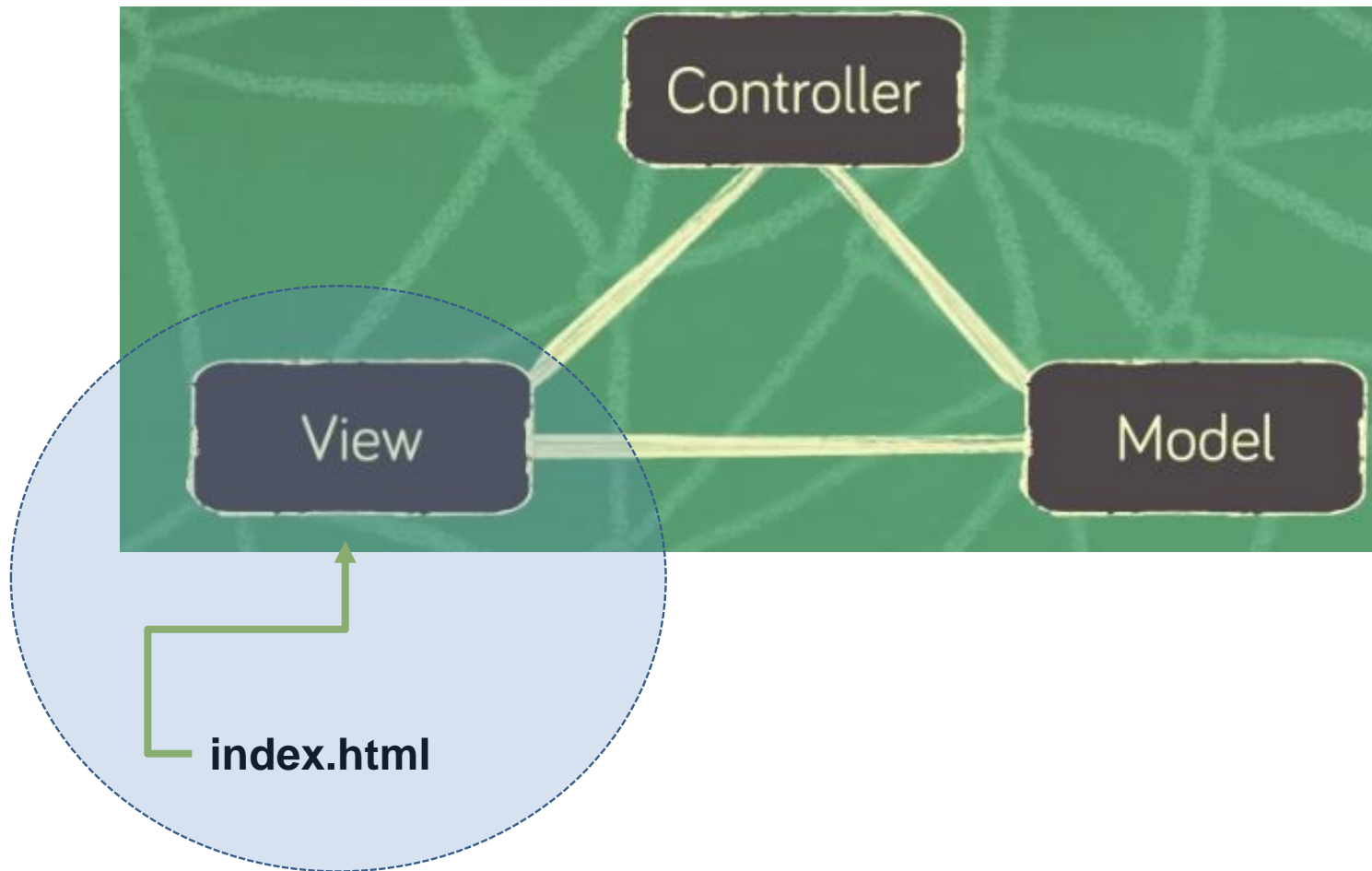
- ▶ Aplicando **ng-app**
 - ▶ Ligar a **app** ao **index.html**

Tudo o que estiver dentro do `<body>` pertencerá a essa aplicação

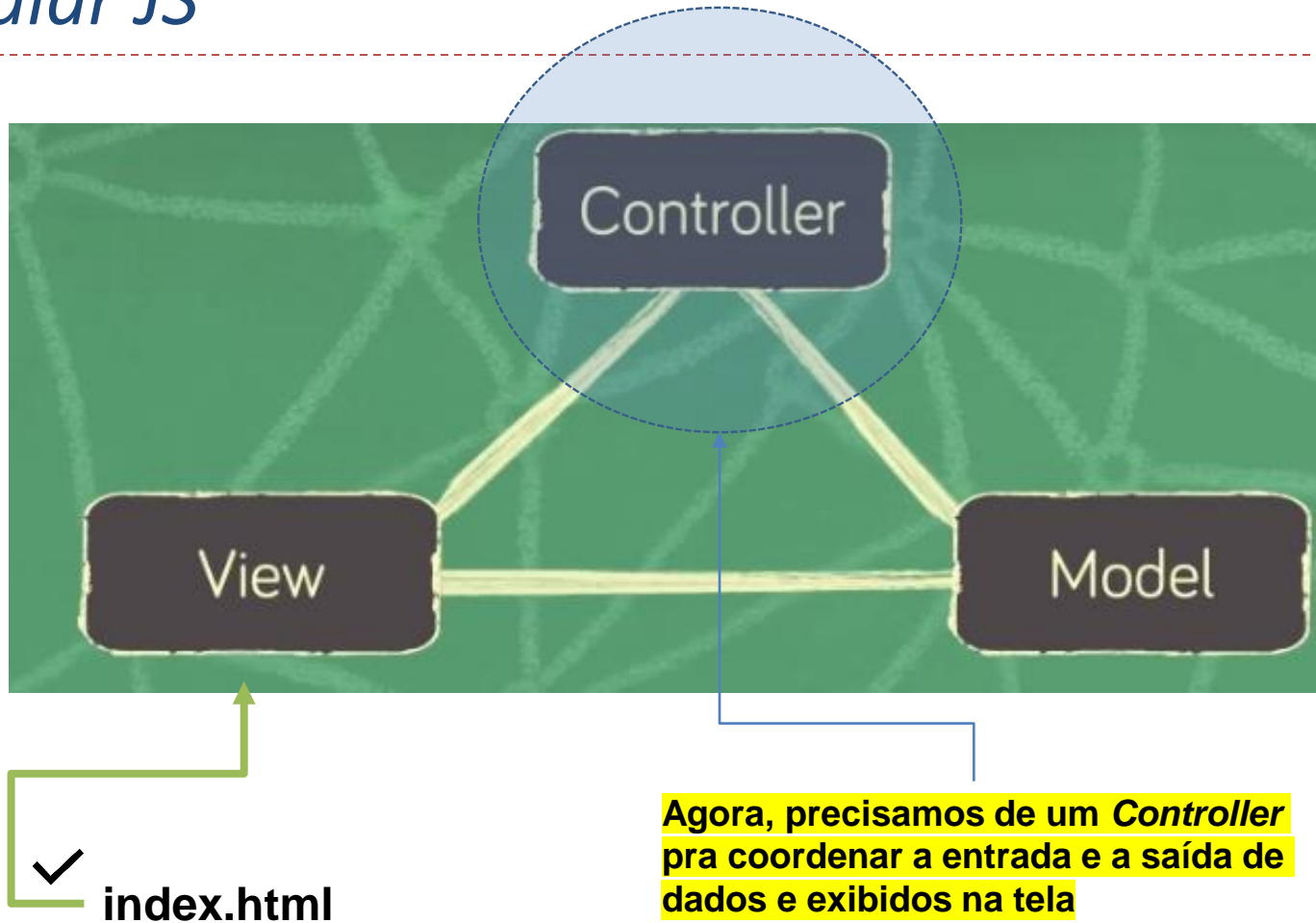


```
<body class="container-fluid" ng-app="aplicacao">
  <header>
    <h1 style="text-align: center; color: blue">
      Bem vindo ao Curso de Angular!
    </h1>
  </header>
  <!--referência angular -->
  <script type="text/javascript" src="js/libs/angular.js"></script>
  <script type="text/javascript" src="js/app.js"></script>
</body>
```


Angular JS



Angular JS



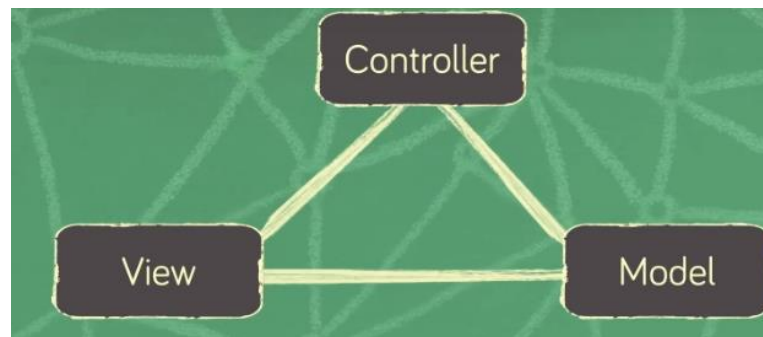
Angular JS

► Directives: **ng-controller**

► Liga um *Controller* a uma página (*View*)

► **Controller**

- Objeto JS que contém modelos e funções que são usados por uma página
- Cada página deverá ter um *Controller* vinculado a ela que realizará toda a lógica de negócios associada.



Angular JS

▶ Aplicando **ng-controller**

▶ Definir um *controller*

- ▶ Criar o arquivo “primeiro-controller.js”

```
angular.module('aplicacao').controller('PrimeiroController', function($scope){  
});
```

▶ **\$scope**

- ❑ Elemento de ligação entre a tela (*View*) e o *Controller*.
- ❑ Permite que a tela faça referência a variáveis e funções definidas no *Controller*.
- ❑ Todas as variáveis e funções definidas no *\$scope* podem ser acessadas pela *View*

Angular JS

► Aplicando *ng-controller*

Fazendo referência do
Controller dentro da View

index.html

```
<body class="container-fluid" ng-app="aplicacao">
  <header>
    <h1 style="text-align: center; color: blue">
      Bem vindo ao Curso de Angular!
    </h1>
  </header>
  <div ng-controller="PrimeiroController" class="bg-light"></div>
  <!--referência angular -->
  <script type="text/javascript" src="js/libs/angular.js"></script>
  <script type="text/javascript" src="js/app.js"></script>
  <script type="text/javascript" src="js/primeiro-controller.js"></script>
</body>
```

Angular JS

▶ Aplicando **ng-controller**

- ▶ Declarando variáveis dentro e fora do *\$scope*

primeiro-controller.js

```
angular.module('aplicacao').controller('PrimeiroController', function($scope){  
    $scope.nome = "Joyce Miranda";  
    var nome = "Fulano de Tal";  
});
```

Angular JS

▶ Aplicando **ng-controller**

▶ Acessando variáveis do \$scope na tela

- ▶ Para acessar variáveis do \$scope utilize **{{ }}**

index.html

```
<div ng-controller="PrimeiroController" class="bg-light">  
  Olá {{nome}}, seja bem-vindo(a)!  
</div>
```

Olá Joyce Miranda, seja bem-vindo(a)!

Saída do Navegador

Angular JS

► Directives: **ng-controller**

► ng-show | ng-hide | ng-if

ng-show: exibe um elemento HTML caso a expressão informada seja VERDADEIRA

ng-hide: esconde um elemento do HTML caso a expressão informada seja VERDADEIRA

ng-if: remove um elemento do DOM caso a expressão informada seja VERDADEIRA

Angular JS

► Directives: ng-show | ng-hide | ng-if

primeiro-controller.js

```
angular.module('aplicacao').controller('PrimeiroController', function($scope){  
  $scope.nome = "Joyce Miranda";  
  var nome = "Fulano de Tal";  
  $scope.iniciado = false;  
});
```

index.html

```
<div ng-controller="PrimeiroController" class="bg-light">  
  <p>Olá {{nome}}, seja bem-vindo(a)!</p>  
  <p ng-show="iniciado">Curso em Andamento</p>  
  <p ng-hide="iniciado">Curso Finalizado</p>  
  <p ng-if="!iniciado">Aguarde o Próximo Curso...</p>  
</div>
```

Olá Joyce Miranda, seja bem-vindo(a)!

Curso Finalizado

Aguarde o Próximo Curso...

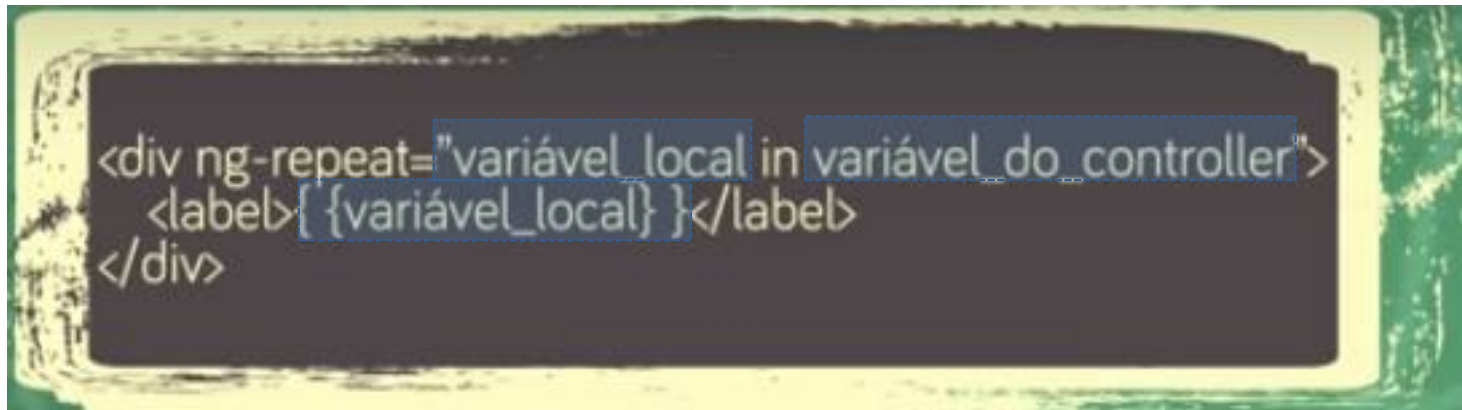
navegador

Angular JS

► Directives

► **ng-repeat**

- Permite a repetição de um trecho HTML para cada elemento de uma coleção



```
<div ng-repeat="variável_local in variável_do_controller">  
  <label>{variável_local}</label>  
</div>
```

Angular JS

► Directives: **ng-repeat**

primeiro-controller.js

```
$scope.alunos = ['Aluno 1', 'Aluno 2', 'Aluno 3'];
```

index.html

```
<li ng-repeat="aluno in alunos">{{aluno}}</li>
```

- Aluno 1
- Aluno 2
- Aluno 3

navegador

Angular JS

► Directives: **ng-repeat**

primeiro-controller.js

```
$scope.alunos = ['Aluno 1', 'Aluno 2', 'Aluno 3'];
```

```
<div class="table-responsive">
  <table class="table table-striped">
    <tr><th>Nome</th></tr>
    <tr ng-repeat="aluno in alunos">
      <td>{{aluno}}</td>
    </tr>
  </table>
</div>
```

index.html (9)

Nome

Aluno 1

Aluno 2

Aluno 3

navegador

Angular JS

► Directives

► **ng-click**

- Permite atribuir comportamento quando um elemento HTML for clicado.



Angular JS

► Directives

► ng-click

index.html (7)

```
<button ng-show="iniciado" class="btn btn-danger" ng-click="finalizar()">Finalizar</button>  
<button ng-hide="iniciado" class="btn btn-success" ng-click="iniciar()">Iniciar</button>
```

primeiro-controller.js

```
$scope.iniciado = true;
```

```
$scope.finalizar = function(){  
  $scope.iniciado = false;  
};
```

```
$scope.iniciar = function(){  
  $scope.iniciado = true;  
};
```

Curso em Andamento

Finalizar



Curso Finalizado

Aguarde o Próximo Curso...

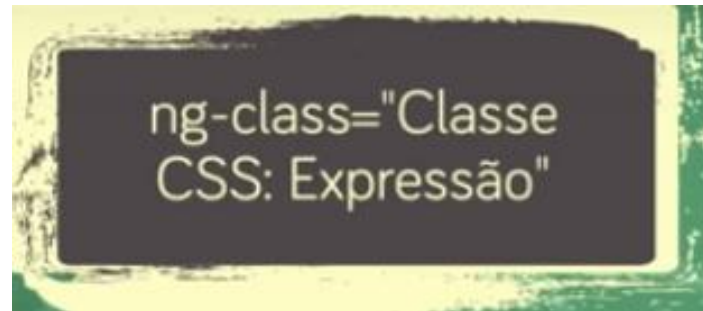
Iniciar

Angular JS

► *Directives*

► **ng-class**

- Aplica uma classe CSS a um elemento caso uma expressão seja verdadeira.



Angular JS

► Directives

► ng-class

index.html (8)

```
<div ng-class="{ 'panel panel-primary': iniciado, 'panel panel-danger': !iniciado }">
  <div class="panel-heading">
    <h3 class="panel-title">Situação do Curso</h3>
  </div>
  <div class="panel-body">
    <p ng-show="iniciado">Curso em Andamento</p>
    <p ng-hide="iniciado">Curso Finalizado</p>
    <p ng-if="!iniciado">Aguarde o Próximo Curso...</p>
  </div>
</div>
```

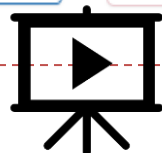
Situação do Curso

Curso em Andamento

Situação do Curso

Curso Finalizado

Aguarde o Próximo Curso...



Angular JS

► Quiz

Marque Verdadeiro ou Falso:

Verdadeiro

Falso

Directives são extensões de atributos htmls com o prefixo ng-

☐☐

Diretivas são elementos de HTML

☐☐

Você precisa desenvolver todas as diretivas que vai utilizar com o AngularJS

☐☐

► Quiz

Relacione as colunas:

Coluna 1		Coluna 2
1. ng-app	<input type="text" value="▼"/>	Variável de ligação entre controller e o html
2. ng-controller	<input type="text" value="▼"/>	Trata o evento de clique do html
3. \$scope	<input type="text" value="▼"/>	Diretiva que relaciona o controllador do view com um bloco de html
4. ng-hide	<input type="text" value="▼"/>	Exibe um bloco de html se a expressão informada avaliar true
5. ng-show	<input type="text" value="▼"/>	Esconde um bloco de html se a expressão informada avaliar true
6. ng-repeat	<input type="text" value="▼"/>	Diretiva que determina a aplicação propriamente dita
7. ng-click	<input type="text" value="▼"/>	Adiciona/remove uma classe css caso a expressão fornecida seja verdadeira ou falsa, respectivamente
8. ng-class	<input type="text" value="▼"/>	Repete um bloco de html para cada elemento do array fornecido

Angular JS

▶ Forms

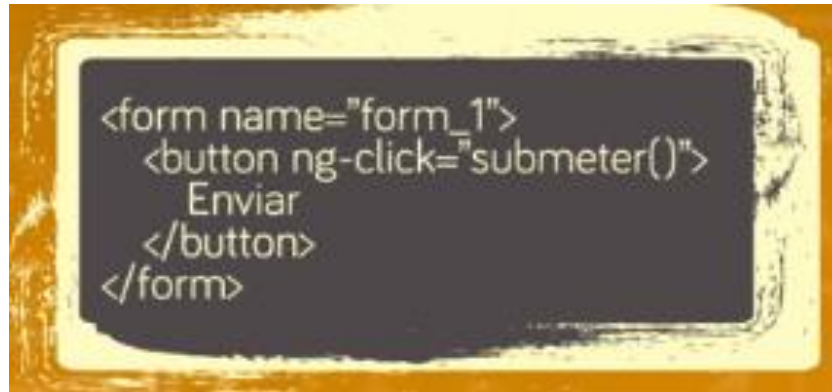
- ▶ São elementos HTML utilizados para a coleta de dados da *View*

▶ *\$scope*

- ▶ Elemento JS de ligação entre o *Controller* e a *View*

- ▶ Os Forms utilizam o ***\$scope*** para recuperar elementos da ***View*** e enviar para o ***Controller***

Angular JS



```
<form name="form_1">  
  <button ng-click="submeter()">  
    Enviar  
  </button>  
</form>
```

- ▶ Quando o nome do *<form>* é definido, o Angular JS vai criar uma propriedade com o nome do *<form>* dentro do *\$scope*, para que esta possa ser acessível no *Controller*.

► Quiz

Sobre Forms HTMLs:

	Verdadeiro	Falso
São elementos html para colectar inputs do usuário.	<input type="radio"/>	<input type="radio"/>
Podem possuir diversos campos de input diferentes.	<input type="radio"/>	<input type="radio"/>
Os valores preenchidos no form pelo usuário podem ser validados.	<input type="radio"/>	<input type="radio"/>

Angular JS

► Directives

► ng-model

- Faz a ligação de um elemento de entrada do **<form>** com uma propriedade do **\$scope**



- Se a propriedade existir no **\$scope** com valor definido, o campo de entrada virá preenchido automaticamente no **<form>**
- Se a propriedade não existir no **\$scope**, quando o valor for preenchido no campo de entrada, a propriedade será automaticamente criada.

Angular JS

► Directives

► ng-model

Nome

index.html (10)

```
<form name="form_1">
  <div class="row">
    <div class="col-md-4">
      <div class="form-group form-group-default">
        <label>Nome</label>
        <input type="text" class="form-control" name="nome_aluno" id="nome_aluno" ng-model="nome_aluno">
      </div>
    </div>
  </div>
  <div class="row">
    <div class="col-md-4">
      <button class="btn btn-primary" ng-click="submeter()">
        Adicionar
      </button>
    </div>
  </div>
</form>
```

Angular JS

► Directives

► ng-model

```
$scope.submeter = function(){  
    $scope.alunos.push($scope.nome_aluno)  
};
```



primeiro_controller.js

Nome

Aluno 1

Aluno 2

Aluno 3

Joyce Miranda

Finalizar

Nome

Joyce Miranda

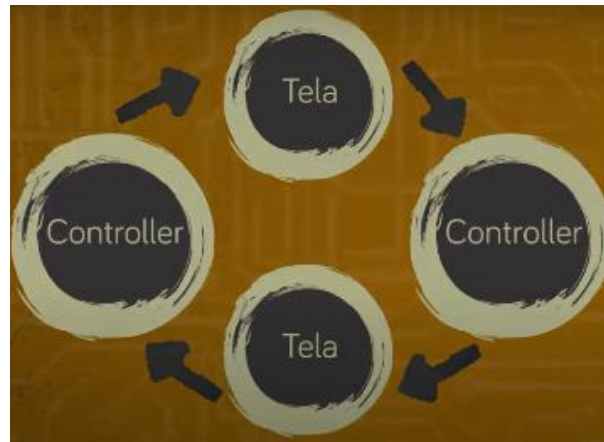
Adicionar

Angular JS

▶ Características

▶ *True Binding*

- ▶ Capacidade de atualização de propriedades em tempo real



- ▶ Se uma propriedade é alterada na tela, o valor é atualizado no **Controller** e vice-versa.

Angular JS

► Características

► *True Binding*

Campos diferentes, mesmo model!

```
<div class="col-md-4">
  <div class="form-group form-group-default">
    <label>Nome</label>
    <input type="text" class="form-control" name="nome_aluno" id="nome_aluno"
  </div>
</div>
<div class="col-md-4">
  <div class="form-group form-group-default">
    <label>Nome 2</label>
    <input type="text" class="form-control" name="nome_aluno_2" id="nome_aluno_2"
  </div>
</div>
```

ng-model="nome_aluno">

ng-model="nome_aluno">



index.html (11)

Nome

Joyce M

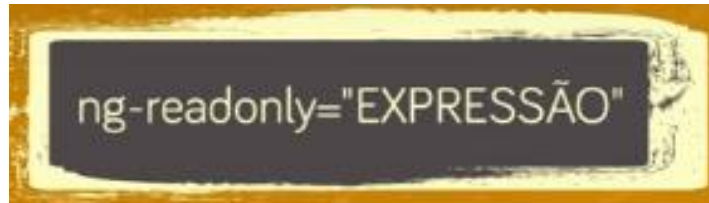
Nome 2

Joyce M

Angular JS

► Directives

► **ng-readonly**



- Torna o campo de um formulário “somente leitura” caso a expressão seja verdadeira.

► **ng-disabled**

- Habilita ou desabilita campos de entrada conforme algumas condições do *Controller*.

Angular JS

► Directives

► ng-readonly / ng-disabled

index.html (12)

```
<input type="text" class="form-control" name="nome_aluno" id="nome_aluno" ng-model="nome_aluno"
ng-readonly="!iniciado">
```

```
<button class="btn btn-primary" ng-click="submeter()" ng-disabled="!iniciado">
  Adicionar
</button>
```



Iniciar

Nome

Adicionar

Finalizar

Nome

Joyce Miranda

Adicionar

Angular JS

▶ *Directives*

▶ **\$valid**

- ▶ Validação de Formulário

▶ **Diretivas Validadoras do Angular**

▶ **ng-required**

- Torna um campo obrigatório caso a expressão seja verdadeira

▶ **ng-maxlength / ng-minlength**

- Define a qtde máxima/mínima de caracteres permitida para um campo

▶ **ng-pattern**

- Define um padrão de *string* permitido para um campos

Angular JS

► Directives

► \$valid

primeiro_controller.js

```
$scope.submeter = function(){  
    if($scope.form_1.$valid){  
        $scope.alunos.push($scope.nome_aluno)  
    }else{  
        alert('Preencha o formulário corretamente!!');  
    }  
};
```

index.html (13)

```
<input type="text" class="form-control" name="nome_aluno" id="nome_aluno" ng-model="nome_aluno"  
ng-readonly="!iniciado"  
ng-minlength="10" ng-maxlength="50" ng-required="true">
```



Nome

Adicionar

► Quiz

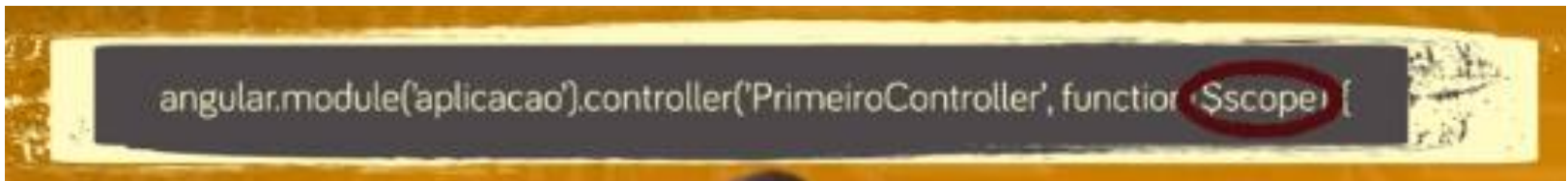
Relacione as colunas:

Coluna 1		Coluna 2
1. ng-model	<input type="text" value="v"/>	Desabilita um elemento do html
2. ng-readonly	<input type="text" value="v"/>	Propriedade booleana do form no Angularjs que indica se ele está válido
3. ng-disable	<input type="text" value="v"/>	Faz com que um campo de input seja somente leitura
4. \$valid	<input type="text" value="v"/>	Relaciona um campo de input com uma propriedade do \$scope do controller associado

Angular JS

► *Filters*

- São funções transformativas que podem ser aplicadas em expressões
- Podem ser usadas tanto nas *Views* quanto nos *Controllers*
- Para usar os *Filters* no *Controller*, devemos adicionar o *Filter* como dependência desse *Controller*



```
angular.module('aplicacao').controller('PrimeiroController', function($scope) {
```

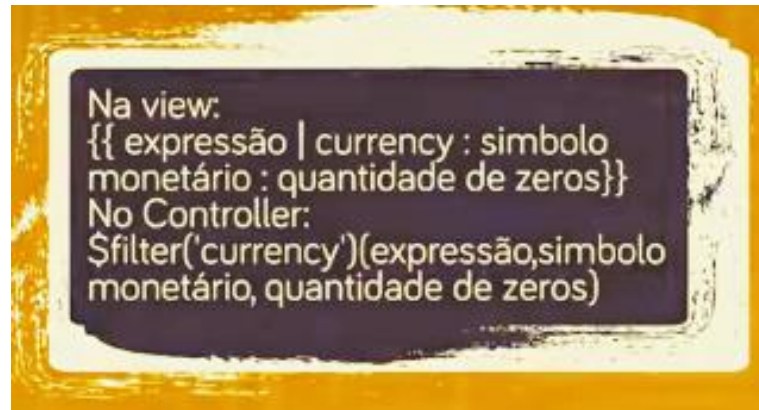
- Os parâmetros da *function* no *Controller* indicam quais recursos esse *Controller* necessita, ou seja, indicam suas dependências.
- Logo, adicionando o *Filter* como parâmetro, ele será acessível dentro desse *Controller*.

Angular JS

► *Filters*

► *currency*

- Formata um número transformando-o em um valor monetário.



Angular JS

► Filters

► *currency*

- Formata um número transformando-o em um valor monetário.

```
<p>Preço: {{valor | currency : "R$ "}}</p>
```

```
<div>  
  <label>Definir Preço</label>  
  <input type="number" id="valor" name="valor" ng-model="valor" />  
</div>
```



Olá Joyce Miranda, seja bem-vindo(a)!

Situação do Curso

Curso em Andamento

Preço: R\$ 250.00

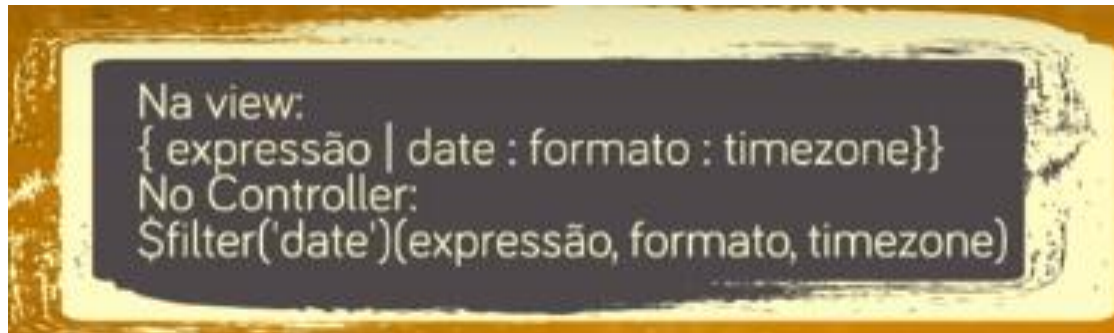
Definir Preço

Angular JS

► Filters

► *date*

- Formata um valor como uma data.



```
$scope.hoje = new Date();
```

```
<p>Última atualização: {{hoje | date: "dd/MM/yyyy"}}</p>
```

Última atualização: 12/06/2020

Angular JS

► *Filters*

► *filter*

- Extrai *subsets* de uma coleção, retornando uma nova coleção.



Angular JS

► Filters

► *filter*

- Extrai *subsets* de uma coleção, retornando uma nova coleção.



```
<div>
  <label>Filtrar por nome: </label>
  <input type="text" name="filtro" ng-model="filtro"/>
</div>

<br/>

<div class="table-responsive">
  <table class="table table-striped">
    <tr><th>Nome</th></tr>
    <tr ng-repeat="aluno in alunos | filter: filtro">
      <td>{{aluno}}</td>
    </tr>
  </table>
</div>
```

Filtrar por nome:	<input type="text"/>
Nome	
Aluno 1	
Aluno 2	
Aluno 3	

Filtrar por nome:	<input type="text" value="1"/>
Nome	
Aluno 1	

► Alterando o Form



Nome	Idade
Joãozinho	9
Ricardinho	11
Felipinho	11
Zildinha	14
Marianinha	10
Luluzinha	12
Fulana de Tal	25

Finalizar

Nome

Fulana de Tal

Idade

25

Adicionar

Angular JS

► Alterando o Form

```
<div class="row">
  <div class="col-md-4">
    <div class="form-group form-group-default">
      <label>Nome</label>
      <input type="text" class="form-control" name="nome_aluno" id="nome_aluno" ng-model="nome_aluno"
        ng-readonly="!iniciado"
        ng-minlength="10" ng-maxlength="50" ng-required="true">
    </div>
  </div>
  <div class="col-md-4">
    <div class="form-group form-group-default">
      <label>Idade</label>
      <input type="number" class="form-control" name="idade_aluno" id="idade_aluno" ng-model="idade_aluno"
        ng-readonly="!iniciado" ng-required="true">
    </div>
  </div>
</div>
```

Angular JS

► Alterando o Form

```
$scope.alunos = [{ 'nome': 'Joãozinho', 'idade': 9},  
                  { 'nome': 'Ricardinho', 'idade': 11},  
                  { 'nome': 'Felipinho', 'idade': 11},  
                  { 'nome': 'Zildinha', 'idade': 14},  
                  { 'nome': 'Marianinha', 'idade': 10},  
                  { 'nome': 'Luluzinha', 'idade': 12}];
```

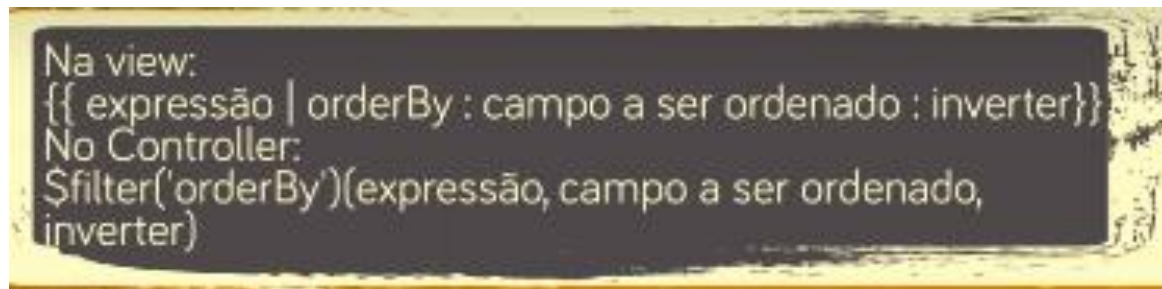
```
$scope.submeter = function() {  
    if($scope.form_1.$valid) {  
        novo_aluno = {};  
        novo_aluno['nome'] = $scope.nome_aluno;  
        novo_aluno['idade'] = parseInt($scope.idade_aluno);  
        $scope.alunos.push(novo_aluno);  
    } else {  
        alert("Preencha o formulário corretamente");  
    }  
};
```


Angular JS

► *Filters*

► *order by*

- Ordena uma coleção de dados/objetos



Na view:
{{ expressão | orderBy : campo a ser ordenado : inverter }}

No Controller:
\$filter('orderBy')(expressão, campo a ser ordenado,
inverter)

Angular JS

► *Filters*

► *order by*

- Ordena uma coleção de dados/objetos

```
<tr ng-repeat="aluno in alunos | filter: filtro | orderBy: 'idade' : false ">  
  <td>{{aluno.nome}}</td>  
  <td>{{aluno.idade}}</td>  
</tr>
```

Angular JS

► Filters



► *order by*

- Ordenação com click na coluna..

Nome	Idade
Zildinha	14
Luluzinha	12
Felipinho	11
Ricardinho	11
Marianinha	10
Joãozinho	9

Angular JS

▶ *\$filter*

- ▶ *Para utilizar o filter precisamos adicioná-lo como parâmetro da função Controller*
- ▶ *O Angular utiliza de injeção de dependência para adicionar esse serviço ao nosso Controller*

▶ *Injeção de Dependência*

- ▶ *É um design pattern de software*
 - ▶ *Trata a forma como os componentes de software obtêm suas dependências*
- ▶ *Componentes dependem de outros elementos e de alguma forma devem estar disponíveis para eles qdo eles forem executados.*

▶ *Injeção de Dependência no Angular*

- ▶ *O Angular possui um subsistema q identifica as dependências e as provê aos componentes*
 - ▶ *Services*
 - *Qdo o Angular identifica uma dependência do tipo Service, ele verifica se uma instancia desse Service já existe.. Passando essa como referência. Se não existir ele cria uma*
 - *São Singletons: Instâncias únicas dentro do sistema*
 - ▶ *Controllers*
 - *Qdo uma View requisita um Controller, uma nova instância do Controller será criada*
 - *Diferentes requisições geram diferentes instâncias do Controller*

Angular JS

► Filters

► *order by*

- Ordenação com click na coluna..

```
angular.module('aplicacao').controller('PrimeiroController', function($scope, $filter){
```

```
$scope.ordenarPorNome = function(){  
    $scope.ordenadoPorNome = !$scope.ordenadoPorNome;  
    $scope.alunos = $filter('orderBy')($scope.alunos, 'nome', $scope.ordenadoPorNome);  
}  
  
$scope.ordenarPorIdade = function(){  
    $scope.ordenadoPorIdade = !$scope.ordenadoPorIdade;  
    $scope.alunos = $filter('orderBy')($scope.alunos, 'idade', $scope.ordenadoPorIdade);  
}
```

Angular JS

► Filters

► *order by*

- Ordena uma coleção de dados/objetos

```
<div class="table-responsive">
  <table class="table table-striped">
    <tr>
      <th ng-click="ordenarPorNome()">Nome</th>
      <th ng-click="ordenarPorIdade()">Idade</th>
    </tr>
    <tr ng-repeat="aluno in alunos | filter: filtro ">
      <td>{{aluno.nome}}</td>
      <td>{{aluno.idade}}</td>
    </tr>
  </table>
</div>
```

► Quiz

Sobre os Filters do AngularJS, marque Verdadeiro ou Falso:

	Verdadeiro	Falso
Filters são funções transformativas	<input type="radio"/>	<input type="radio"/>
Filters podem ser utilizados tanto nas views quanto nos controllers.	<input type="radio"/>	<input type="radio"/>
O filter currency transforma o valor fornecido em um formato monetário	<input type="radio"/>	<input type="radio"/>
O filter date transforma o valor fornecido em um objeto do tipo date	<input type="radio"/>	<input type="radio"/>
O filter filter filtra padrões de expressões regulares	<input type="radio"/>	<input type="radio"/>
O filter orderBy ordena uma coleção	<input type="radio"/>	<input type="radio"/>

Angular JS

► *Services*

- Elementos JS especializados utilizados para resolver problemas específicos da aplicação.
- Permite criar elementos reutilizáveis na aplicação, podendo ser reaproveitados em diferentes *Controllers*.



Angular JS

► Services

► \$http

- Serviço do Angular para comunicação HTTP
- Utilizado para fazer requisições ao servidor *back.end*, seja para enviar ou para solicitar dados.



Angular JS

► Services

► \$http

► Chamada dos métodos

```
$http.get('/url').then(function(response) {  
  função executada caso a resposta seja de sucesso  
}, function(response) {  
  função executada caso a resposta seja de erro.  
});
```

Requisição

```
$http.get  
$http.head  
$http.post  
$http.put  
$http.delete
```

Resposta

```
response.status; - contém o status da resposta.  
response.data; - os dados retornados.
```

Angular JS

► *Services*

- Comunicação de aplicações web SPA com o *back.end* é feita utilizando recursos REST.
- REST
 - Chamadas HTTP que tem como retorno dados no formato JSON
- Para comunicação com serviços REST existe o *ngResource*



- Proporciona métodos de mais alto nível que o **\$http**

Angular JS

► Services

► ngResource

- Não vem por padrão no Angular
- Precisa ser baixado e incluído no projeto

<https://docs.angularjs.org/api/ngResource>

"/code.angularjs.org/X.Y.Z/angular-resource.js"

index.html

```
<script type="text/javascript" src="js/libs/ngResource.js"></script>
```

```
var app = angular.module('aplicacao', ['ngResource']);
```

app.js

Angular JS

▶ Acessando Back.End

- ▶ Vamos utilizar uma aplicação web existente que já oferece seus recursos.

- ▶ <http://api.openweathermap.org/data/2.5/weather?q=Rio%20de%20Janeiro,BR&units=metric&APPID=305bd017d8cd1bd5528226603b96b93b>

```
{ "coord": { "lon": -43.21, "lat": -22.9 }, "weather": [ { "id": 803, "main": "Clouds", "description": "broken clouds", "icon": "04n" } ], "base": "stations", "main": { "temp": 21.55, "feels_like": 22.68, "temp_min": 21, "temp_max": 22, "pressure": 1026, "humidity": 73 }, "visibility": 10000, "wind": { "speed": 1.5, "deg": 150 }, "clouds": { "all": 75 }, "dt": 1592362789, "sys": { "type": 1, "id": 8429, "country": "BR", "sunrise": 1592386321, "sunset": 1592424945, "timezone": -10800, "id": 3451190, "name": "Rio de Janeiro", "cod": 200 }
```

Angular JS

▶ Acessando Back.End

- ▶ Vamos criar um *Controller* para acessar o recurso

previsao-tempo-controller.js

```
angular.module('aplicacao').controller('PrevisaoTempoController', function($scope, $resource){  
});
```

index.html

```
<script type="text/javascript" src="js/previsao-tempo-controller.js"></script>
```

Angular JS

► Acessando Back.End

- Vamos criar um *Controller* para acessar o recurso

previsao-tempo-controller.js

```
angular.module('aplicacao').controller('PrevisaoTempoController', function($scope, $resource){  
  
    var weatherChanel =  
        $resource(  
            'http://api.openweathermap.org/data/2.5/weather?q=Rio de Janeiro,BR&units=metric&APPID=305bd017d8cd1bd5528226603b96b93b');  
  
    var getWeather = function(){  
        weatherChanel.get().$promise.then(function(response){  
            $scope.tempo = response;  
            console.log($scope.tempo);  
        }, function(promise) {  
            alert('Erro ao acessar recurso!');  
        });  
    }  
  
    getWeather();  
});
```

Angular JS

▶ Acessando Back.End

- ▶ Vamos criar um *Controller* para acessar o recurso

```
<div class="well" ng-controller="PrevisaoTempoController">  
  <h3>Demonstrando ng-resource</h3>  
  <h3>Previsão do Tempo</h3>  
  <h5>Local: {{tempo.name}}</h5>  
  <h5>Temp.: {{tempo.main.temp}}</h5>  
  <h5>Temp. Min: {{tempo.main.temp_min}}</h5>  
  <h5>Temp. Max: {{tempo.main.temp_max}}</h5>  
</div>
```

Demonstrando ng-resource

index.html

Previsão do Tempo

Local: Rio de Janeiro

Temp.: 21.09 C

Temp. Min: 20 C

Temp. Max: 22 C

► Quiz

Selecione sobre o \$http e o \$resource as alternativas Verdadeiras:

- ☐ Ambos são serviços que vêm por padrão com o AngularJS
- ☐ Ambos realizam comunicação http com o backend.
- ☐ O service \$http realiza chamadas à recursos rest.
- ☐ O service \$resource é um recurso para chamadas http genéricas.

Angular JS

▶ Criando nossas próprias *Services*

- ▶ *Vamos criar um serviço **AlunosCollectionService** que vai ser responsável por executar métodos relacionados à coleção de alunos.*

```
angular.module('aplicacao').service(AlunosCollectionService', function($filter){  
  
});
```

alunos-collective-service.js

```
<script type="text/javascript" src="js/alunos-collection-service.js"></script>
```

index.html

```
angular.module('aplicacao').service('AlunosCollectionService', function($filter){
```

```
    var alunos = [{ 'nome': 'Joãozinho', 'idade': 9},  
                    { 'nome': 'Ricardinho', 'idade': 11},  
                    { 'nome': 'Felipinho', 'idade': 11},  
                    { 'nome': 'Zildinha', 'idade': 14},  
                    { 'nome': 'Marianinha', 'idade': 10},  
                    { 'nome': 'Luluzinha', 'idade': 12}];
```

```
    this.getAlunos = function(){  
        return alunos;  
    }
```

```
    this.adicionarAluno = function(aluno){  
        alunos.push(aluno);  
    }
```

```
    var ordenadoPorNome = false;  
    var ordenadoPorIdade = false;
```

```
    this.ordenarPorNome = function(){  
        ordenadoPorNome = !ordenadoPorNome;  
        alunos = $filter('orderBy')(alunos, 'nome', ordenadoPorNome);  
    }
```

```
    this.ordenarPorIdade = function(){  
        ordenadoPorIdade = !ordenadoPorIdade;  
        alunos = $filter('orderBy')(alunos, 'idade', ordenadoPorIdade);  
    }
```

```
});
```

alunos-collective-service.js

Angular JS

► Criando nossas próprias *Services*

► Criando um novo Controller com função específica para Cadastrar Alunos

```
angular.module('aplicacao').controller('NovaInscricaoController', function($scope, AlunosCollectionService) {  
  
    $scope.submeter = function() {  
        if($scope.form_1.$valid) {  
            novo_aluno = {};  
            novo_aluno['nome'] = $scope.nome_aluno;  
            novo_aluno['idade'] = parseInt($scope.idade_aluno);  
            AlunosCollectionService.adicionarAluno(novo_aluno);  
        } else {  
            alert("Preencha o formulário corretamente");  
        }  
    };  
});
```

nova-inscrição-controller.js

```
<script type="text/javascript" src="js/nova-inscricao-controller.js"></script>
```

index.html

Angular JS

► Criando nossas próprias *Services*

index.html

- Adaptando index.html | Associando o **NovaInscricaoController** | Removendo referências a variáveis do \$scope não mais utilizadas

```
<div class="well" ng-controller="NovaInscricaoController">
  <form name="form_1">
    <div class="row">
      <div class="col-md-4">
        <div class="form-group form-group-default">
          <label>Nome</label>
          <input type="text" class="form-control" name="nome_aluno" id="nome_aluno" ng-model="nome_aluno"
            ng-minlength="10" ng-maxlength="50" ng-required="true">
        </div>
      </div>
      <div class="col-md-4">
        <div class="form-group form-group-default">
          <label>Idade</label>
          <input type="number" class="form-control" name="idade_aluno" id="idade_aluno" ng-model="
            idade_aluno" ng-required="true">
        </div>
      </div>
    </div>
    <div class="row">
      <div class="col-md-4">
        <button class="btn btn-primary" ng-click="submeter()" >
          Adicionar
        </button>
      </div>
    </div>
  </form>
</div>
```

Angular JS

► Criando nossas próprias *Services*

- *Criando um novo Controller com função específica para Listar Alunos*

```
angular.module('aplicacao').controller('ListagemAlunosController', function($scope, $filter, AlunosCollectionService){  
  
});
```

listagem-alunos-controller.js

```
<div ng-controller="ListagemAlunosController" class="bg-light">
```

index.html

primeiro-controller.js => listagem-aluno-controller.js

```
angular.module('aplicacao').controller('ListagemAlunosController', function($scope, $filter, AlunosCollectionService){
  $scope.nome = "Joyce Miranda";
  var nome = "Fulano de Tal";

  $scope.alunos = AlunosCollectionService.getAlunos();

  $scope.iniciado = true;

  $scope.hoje = new Date();

  $scope.finalizar = function(){
    $scope.iniciado = false;
  };

  $scope.iniciar = function(){
    $scope.iniciado = true;
  };

  $scope.ordenarPorNome = function(){
    AlunosCollectionService.ordenarPorNome();
    $scope.alunos = AlunosCollectionService.getAlunos();
  }

  $scope.ordenarPorIdade = function(){
    AlunosCollectionService.ordenarPorIdade();
    $scope.alunos = AlunosCollectionService.getAlunos();
  }

});
```

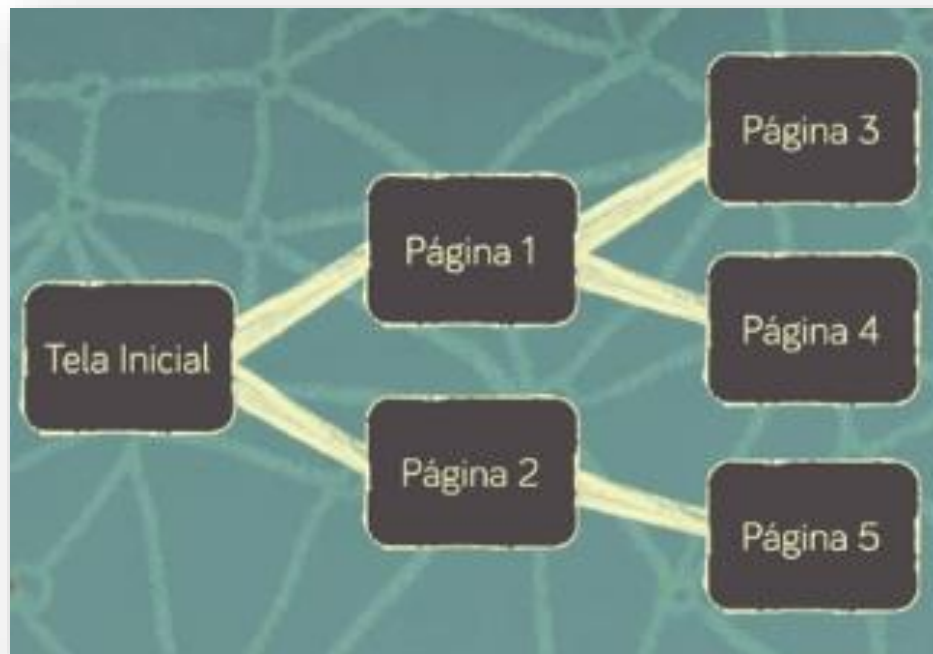
***** Fazendo referências aos métodos do AlunosCollectionService**

***** Remove a função de submissão**

Angular JS

► *Routing*

- Mecanismo de navegabilidade
 - Transição de uma página X para uma página Y



Angular JS

▶ *Routing*

▶ Navegação tradicional

- ▶ Ao acessar uma URL o navegador vai carregar o conteúdo da página, carregando toda a aplicação novamente.

▶ SPA no Angular

- ▶ A navegação funciona por substituição de *templates*
- ▶ O primeiro *load* carregará a maioria das informações
- ▶ Cada transição de página carregará o trecho HTML de um conteúdo específico
- ▶ Cabe definir qual trecho HTML será correspondente a cada uma das páginas da aplicação

Angular JS

▶ *Routing*

- ▶ É necessário definir qual trecho HTML será correspondente a cada uma das páginas da aplicação

▶ Conceito

▶ **Página da Aplicação**

- É um estado de exibição.
 - Página Inicial => Estado Inicial

▶ **Navegação**

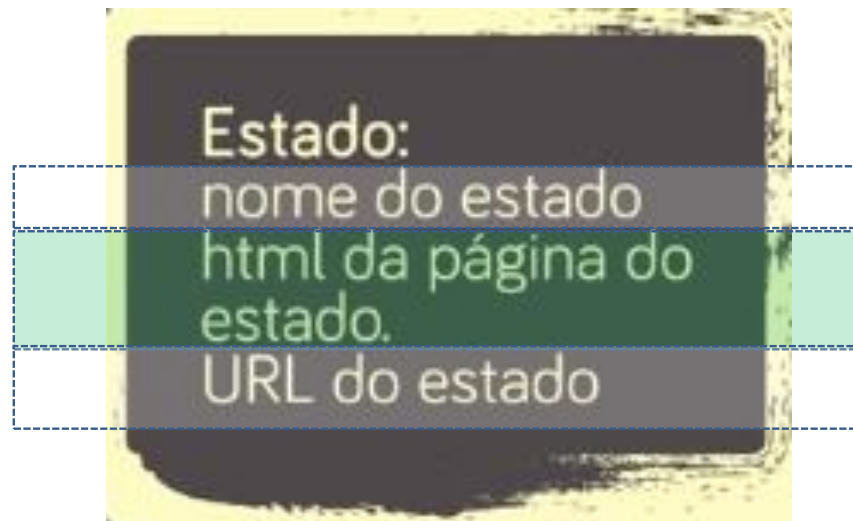
- Transição de um Estado X para um Estado Y



Angular JS

► *Routing*

- Gerenciamento de transição entre estados
- Verifica qual o estado desejado e identifica qual página deverá ser exibida para esse estado.



Identificação de um Estado

Angular JS

► Routing

- Para definirmos a navegabilidade da aplicação é necessário:
 - Definir um conjunto de **rotas** (estados)
 - Criar os arquivos HTML correspondentes a cada estado

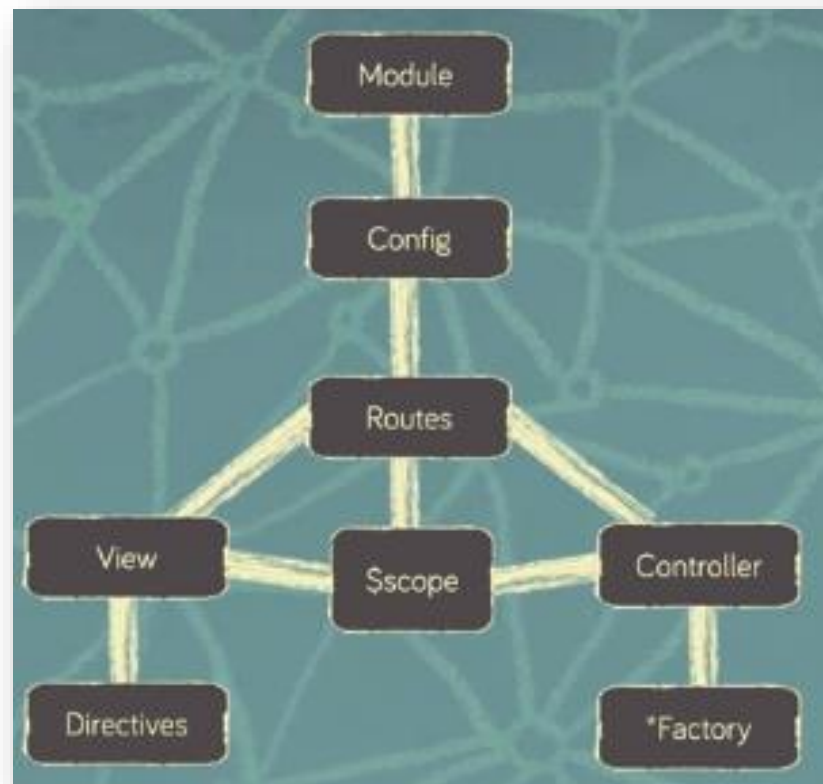
Estados da Aplicação



Angular JS

► *Routing*

- Diagrama da arquitetura de uma aplicação Angular JS



Angular JS

► *Routing*

► *ngRoute*

- Não vem por padrão no Angular
- Precisa ser baixado e incluído no projeto

<https://docs.angularjs.org/api/ngRoute>

"//code.angularjs.org/X.Y.Z/angular-route.js"

index.html

```
<script type="text/javascript" src="js/libs/ngRoute.js"></script>
```

```
var app = angular.module('aplicacao', ['ngResource', 'ngRoute']);
```

app.js

Angular JS

► Routing

► Registrando o roteamento

```
var app = angular.module('aplicacao', ['ngResource', 'ngRoute']);

app.config(function($routeProvider){
    $routeProvider
        .when('/aplicacao-inicial',{
            templateUrl: 'views/inicial.html'
        })
        .when('/aplicacao-listagem',{
            templateUrl: 'views/listagem-alunos.html'
        })
        .when('/aplicacao-cadastro',{
            templateUrl: 'views/cadastro-aluno.html'
        })
        .when('/aplicacao-previsao',{
            templateUrl: 'views/previsao.html'
        })
        .otherwise('/aplicacao-inicial')
});
```

app.js

Angular JS

▶ *Routing*

- ▶ Criando arquivos *.html dentro da pasta views



cadastro-aluno



inicial



listagem-alunos



previsao

Angular JS

► *Routing*

► Editando conteúdo do inicial.html

```
<div class="well">
  <h3>Página Inicial</h3>|
  <h2>Seja bem-vindo(a)!</h2>
  <h1>Para mais informações acesse outras páginas através do menu.</h1>
</div>
```

inicial.html

Angular JS

► Routing

- Editando conteúdo das outras páginas ***.html**
 - Copiando trechos (<divs>) da página **index.html** que correspondem a *Controllers*

```
<div class="well" ng-controller="PrevisaoTempoController">
  <h3>Demonstrando ng-resource</h3>
  <h3>Previsão do Tempo</h3>
  <h5>Local: {{tempo.name}}</h5>
  <h5>Temp.: {{tempo.main.temp}} C</h5>
  <h5>Temp. Min: {{tempo.main.temp_min}} C</h5>
  <h5>Temp. Max: {{tempo.main.temp_max}} C</h5>
</div>
```

previsao.html

Angular JS

► Routing

► Editando conteúdo das outras páginas *.html

- Copiando trechos (<divs>) da página index.html que correspondem a *Controllers*

```
<div class="well" ng-controller="NovaInscricaoController">
  <form name="form_1">
    <div class="row">
      <div class="col-md-4">
        <div class="form-group form-group-default">
          <label>Nome</label>
          <input type="text" class="form-control" name="nome_aluno" id="nome_aluno" ng-model="nome_aluno"
            ng-minlength="10" ng-maxlength="50" ng-required="true">
        </div>
      </div>
      <div class="col-md-4">
        <div class="form-group form-group-default">
          <label>Idade</label>
          <input type="number" class="form-control" name="idade_aluno" id="idade_aluno" ng-model="idade_aluno"
            ng-required="true">
        </div>
      </div>
    </div>
    <div class="row">
      <div class="col-md-4">
        <button class="btn btn-primary" ng-click="submeter()" >
          Adicionar
        </button>
      </div>
    </div>
  </form>
</div>
```

```

<div ng-controller="ListagemAlunosController" class="bg-light">
  <p>Olá {{nome}}, seja bem-vindo(a)!</p>

  <div ng-class="{ 'panel panel-primary': iniciado, 'panel panel-danger': !iniciado }">
    <div class="panel-heading">
      <h3 class="panel-title">Situação do Curso</h3>
    </div>
    <div class="panel-body">
      <p ng-show="iniciado">Curso em Andamento</p>
      <p ng-hide="iniciado">Curso Finalizado</p>
      <p ng-if="!iniciado">Aguarde o Próximo Curso...</p>
      <p>Preço: {{valor | currency: "R$ "}}</p>
      <p>Última atualização: {{hoje | date: "dd/MM/yyyy"}}</p>
    </div>
  </div>

  <div>
    <label>Definir Preço</label>
    <input type="number" id="valor" name="valor" ng-model="valor" />
  </div>

  <br/>

  <div>
    <label>Filtrar por nome: </label>
    <input type="text" name="filtro" ng-model="filtro" />
  </div>

  <br/>

  <div class="table-responsive">
    <table class="table table-striped">
      <tr>
        <th ng-click="ordenarPorNome()">Nome</th>
        <th ng-click="ordenarPorIdade()">Idade</th>
      </tr>
      <tr ng-repeat="aluno in alunos | filter: filtro">
        <td>{{aluno.nome}}</td>
        <td>{{aluno.idade}}</td>
      </tr>
    </table>
  </div>

  <br/>

  <button ng-show="iniciado" class="btn btn-danger" ng-click="finalizar()">Finalizar</button>
  <button ng-hide="iniciado" class="btn btn-success" ng-click="iniciar()">Iniciar</button>
</div>

```

Listagem-alunos.html

Angular JS

► Routing

► Editando index.html

- Incluindo um menu navegacional e ng-view

► ng-view

- Componente que terá o conteúdo atualizado conforme a transição de estado.

```
<nav class="navbar navbar-default" ng-controller="MenuController">
  <div class="collapse navbar-collapse" id="menu">
    <ul class="nav navbar-nav">
      <li><a ng-click="goInicio()">Início</a></li>
      <li><a ng-click="goListagem()">Alunos</a></li>
      <li><a ng-click="goCadastro()">Cadastro</a></li>
      <li><a ng-click="goPrevisao()">Previsão do Tempo</a></li>
    </ul>
  </div>
</nav>
```

index.html

```
<div ng-view></div>
```

Angular JS

```
<script type="text/javascript" src="js/menu-controller.js"></script>
```

► Routing

- Criando menu-controller.js com as funções referenciadas no menu

```
angular.module('aplicacao').controller('MenuController', function($scope, $location){  
  
    $scope.goInicio = function() {  
        $location.path('/aplicacao-inicial')  
    };  
  
    $scope.goListagem = function() {  
        $location.path('/aplicacao-listagem')  
    };  
  
    $scope.goCadastro = function() {  
        $location.path('/aplicacao-cadastro')  
    };  
  
    $scope.goPrevisao = function() {  
        $location.path('/aplicacao-previsao')  
    };  
  
});
```

menu-controller.js

► Quiz

Sobre navegação em aplicações SPA, marque Verdadeiro ou Falso:

	Verdadeiro	Falso
Não existe navegação em aplicações SPA.	<input type="radio"/>	<input type="radio"/>
Em aplicações SPA, quando se clica em um link ou botão, o navegador faz uma requisição http e carrega a página correspondente	<input type="radio"/>	<input type="radio"/>
Navegar em uma aplicação significa transitar de uma página x para outra página y.	<input type="radio"/>	<input type="radio"/>
Routing é um mecanismo para realizar a transição de estados.	<input type="radio"/>	<input type="radio"/>
A navegação é feita definindo estados e informando quando deseja-se fazer uma transição.	<input type="radio"/>	<input type="radio"/>

Angular JS

► Custom Directives

- Criando diretivas próprias para utilizar no seu projeto



- Útil quando temos elementos de tela q se repetem em diferentes momentos do sistema
- Assim, podemos criar uma única diretiva que gera um componente que pode ser reaproveitado

Angular JS

▶ *Custom Directives*

▶ Criando Diretivas

- ▶ Utilize um prefixo próprio para o nome da diretiva criada, de forma a diferenciá-la das diretivas existentes do Angular JS, evitando conflitos e colisões.



Angular JS

▶ Custom Directives

▶ Criando Diretivas

- ▶ Crie o arquivo JS específico para a diretiva

```
angular.module('MODULO').directive('MeuComponenteDirective', function () {  
  return {  
    restrict: 'RESTRICÇÃO',  
    templateUrl: URL  
  };  
});
```

▶ Nome da Diretiva

- ❑ O Angular JS irá separar o nome em “Camel Case” por um nome “Lower Case”, utilizando hífen para separar as palavras
 - ❑ Ex: NomeDiretiva => nome-diretiva
- ❑ Quando o Angular JS parsear a **View** e encontrar o nome-diretiva, ele irá procurar uma **directive** correspondente a NomeDiretiva para sua execução.



Angular JS

► Custom Directives

► Criando Diretivas

- Crie o arquivo JS específico para a diretiva

```
angular.module('MODULO').directive('MeuComponenteDirective', function () {  
  return {  
    restrict: 'RESTRICÇÃO',  
    templateUrl: URL  
  };  
});
```

- Quando vc cria um diretiva, vc precisa definir em qual tipo de elemento ela poderá ser aplicada

- ☐ Restrict : “Restrição”
 - ☐ Valor Padrão: EA

E - elemento html (default):
<nome-diretiva></nome-diretiva>

A - atributo html (default): <div
nome-diretiva="exp"></div>

C - Class: <div
class="nome-diretiva: exp,"></div>

WEB

Angular JS

► Custom Directives

► Estudo de Caso

- Encapsular a Tela de Cadastro de Aluno para ser utilizada tanto no módulo de Cadastro qto no módulo de Listagem

Novo Cadastro

Nome

Idade

Adicionar



Angular JS

► Custom Directives

► Estudo de Caso

- Step: Criando Arquivo “cadastro-aluno-directive.js” para a nova Diretiva

cadastro-aluno-directive.js

```
angular.module('aplicacao').directive('cadastroAluno', function(){  
    return{  
        restrict: 'E',  
        templateUrl: 'views/cadastro-aluno-template.html'  
    };  
});
```

- Step: Referenciando “cadastro-aluno-directive.js” na View

index.html

```
<script type="text/javascript" src="js/cadastro-aluno-directive.js"></script>
```



Angular JS

▶ Custom Directives

▶ Estudo de Caso

- ▶ Step: Criando o *Template* (**cadastro-aluno-template.html**) para a diretiva criada
- ▶ Step: Definindo o conteúdo para o *Template*
 - Copiar conteúdo do arquivo “cadastro-aluno.html” para o arquivo “cadastro-aluno-template.html”
 - Substituir o conteúdo do arquivo “cadastro-aluno.html” pelo conteúdo abaixo



Angular JS

► Custom Directives

► Estudo de Caso

views/cadastro-aluno-template.html

```
<div class="well" ng-controller="NovaInscricaoController">
  <form name="form_1">
    <div class="row">
      <div class="col-md-4">
        <div class="form-group form-group-default">
          <label>Nome</label>
          <input type="text" class="form-control" name="nome_aluno" id="nome_aluno" ng-model="nome_aluno"
            ng-minlength="10" ng-maxlength="50" ng-required="true">
        </div>
      </div>
      <div class="col-md-4">
        <div class="form-group form-group-default">
          <label>Idade</label>
          <input type="number" class="form-control" name="idade_aluno" id="idade_aluno" ng-model="idade_aluno"
            ng-required="true">
        </div>
      </div>
    </div>
    <div class="row">
      <div class="col-md-4">
        <button class="btn btn-primary" ng-click="submeter()" >
          Adicionar
        </button>
      </div>
    </div>
  </form>
</div>
```

Angular JS

▶ *Custom Directives*

- ▶ Estudo de Caso

- ▶ Step: Acrescentar diretiva na Tela de Cadastro

```
<cadastro-aluno></cadastro-aluno>
```

views/cadastro-aluno.html

Angular JS

► Custom Directives

► Estudo de Caso

► Step: Acrescentar diretiva

```
<button class="btn btn-primary" ng-click="exibirCadastro()">
  <span ng-hide="cadastrar">Cadastrar</span>
  <span ng-show="cadastrar">Ocultar</span>
</button>

<br/><br/>

<div class="panel panel-primary" ng-if="cadastrar">
  <div class="panel-heading">
    <h3 class="panel-title">Novo Cadastro</h3>
  </div>
  <div class="panel-body">
    <cadastro-aluno></cadastro-aluno>
  </div>
</div>
```

views/listagem-alunos.html



Angular JS

▶ Custom Directives

- ▶ Estudo de Caso
 - ▶ Step: Alterar Controller

```
$scope.cadastrar = false;
```

```
$scope.exibirCadastro = function(){  
    $scope.cadastrar = !$scope.cadastrar;  
}
```

listagem-alunos-controller.js



► Quiz

Quais das alternativas a seguir são vantagens de criar as suas próprias diretivas?

- ☐ Criar componentes html reutilizáveis.
- ☐ Simplificar a geração do DOM pelo Browser.
- ☐ Facilitar a manutenção de código.
- ☐ Criar componentes Javascript reutilizáveis.

► Quiz

Sobre o atributo restrict de uma diretiva, marque Verdadeiro ou Falso:

	Verdadeiro	Falso
Caso seu valor seja E, a diretiva será uma diretiva correspondente a um elemento HTML.	<input type="radio"/>	<input type="radio"/>
Caso seu valor seja A, o HTML da diretiva é automaticamente inserido no DOM.	<input type="radio"/>	<input type="radio"/>
Caso seu valor seja C, o AngularJS trata a diretiva como uma classe Javascript.	<input type="radio"/>	<input type="radio"/>
Uma diretiva deve ser exclusivamente 1 dos 3 tipos.	<input type="radio"/>	<input type="radio"/>



Angular

- ▶ Configurando ambiente de trabalho

<https://angular.io/guide/setup-local>

- ▶ Pré-Requisitos

- ▶ JavaScript

- ▶ HTML

- ▶ CSS

- ▶ Node.js

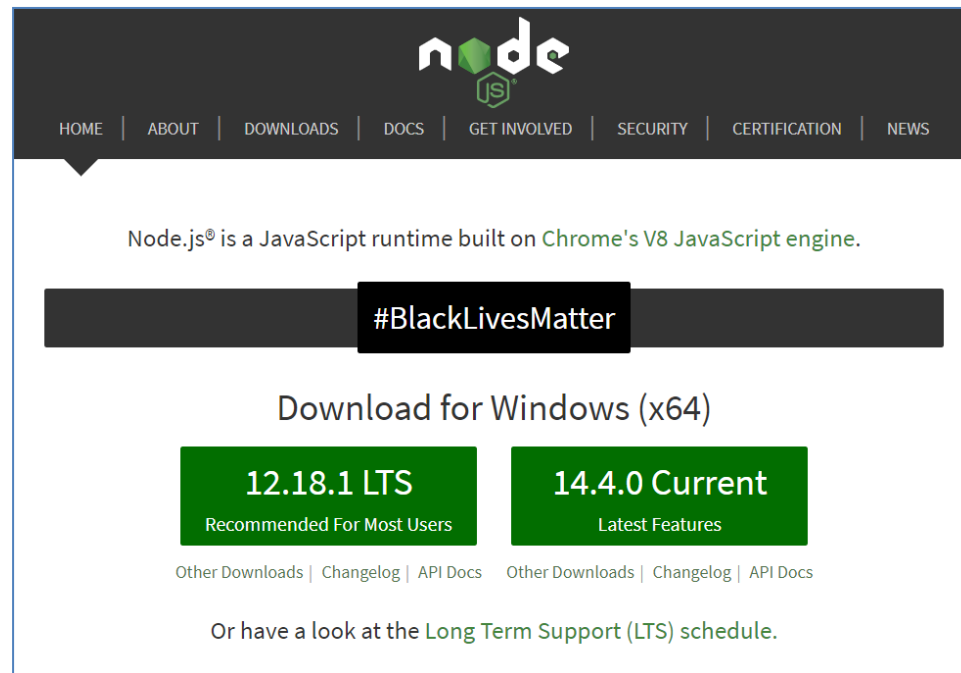
- ▶ Make sure your development environment includes **Node.js**[®] and an **npm** package manager.

<https://nodejs.org/en/>

Angular JS

► Configurando ambiente de trabalho

<https://nodejs.org/en/>



```
C:\Users\mdsjo>node -v  
v12.18.1
```

Angular JS

► Configurando ambiente de trabalho

<https://angular.io/guide/setup-local>

► Install the Angular CLI

- You use the Angular CLI to create projects, generate application and library code, and perform a variety of ongoing development tasks such as testing, bundling, and deployment.

```
npm install -g @angular/cli
```


Angular JS

- ▶ Configurando ambiente de trabalho

<https://angular.io/guide/setup-local>

- ▶ Create a workspace and initial application

- ▶ The **ng new** command prompts you for information about features to include in the initial app.

```
ng new my-app
```

Angular JS

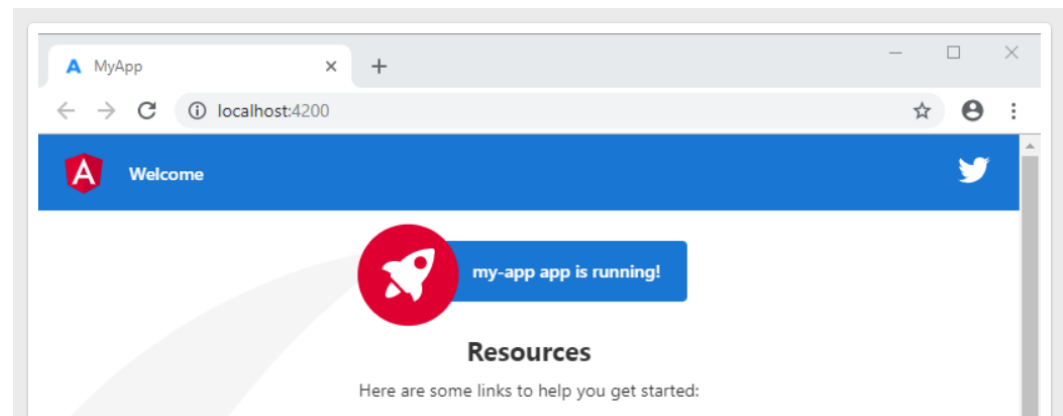
► Configurando ambiente de trabalho

<https://angular.io/guide/setup-local>

► Run the application

- The Angular CLI includes a server, so that you can easily build and serve your app locally.

```
cd my-app  
ng serve --open
```



Angular JS

▶ Estrutura do Projeto

- ▶ Open the project in your favorite editor or IDE and navigate to the `src/app` folder to make some changes to the starter app.

1. `app.component.ts`— the component class code, written in TypeScript.
2. `app.component.html`— the component template, written in HTML.
3. `app.component.css`— the component's private CSS styles.