

AngularJS

Michel Alves

Fernando Baroni

AngularJS



Superheroic JavaScript MVW Framework

Agenda

- **Parte 1 - Apresentadores**
- **Parte 2 - Introdução ao AngularJS**
- **Parte 3 - Diretivas**
- **Parte 4 - Controllers**
- **Parte 5 - Diretiva Customizada**
- **Parte 6 - Desafio**

Apresentadores

Michel Alves

- **Araçoiabense**
- **Vive em Sorocaba**
- **Formado na FATEC Sorocaba**



GFT



2016

<http://baroni.tech/>

6

Fernando Rodrigues Baroni

- **Itapetiningano**
- **Vive em Sorocaba**
- **Formado na UNIP**
- **<http://baroni.tech/>**



IBM



2016

<http://baroni.tech/>

GFT



2016

<http://baroni.tech/>

Stefanini





Features

Super Powers

<input type="checkbox"/> Open Backpacks	<input type="checkbox"/> [1] Level Spy
<input type="checkbox"/> Drop Empty Vials	<input type="checkbox"/> Reconnect <input type="button" value="Au"/>
<input type="checkbox"/> Eat Food	<input type="checkbox"/> Server Save Log
<input type="checkbox"/> From Ground	<input type="checkbox"/> Anti AFK Kick
<input type="checkbox"/> Fishing	<input type="checkbox"/> Stack Items
<input type="checkbox"/> Need Worm	<input type="checkbox"/> Ammo Counter
<input type="text" value="10"/> min cap	<input type="checkbox"/> Screenshot Adv

Basic Features

Many different tools to help with various problems

<input type="checkbox"/> Estimated Creature Level
<input type="checkbox"/> Skill Advanced Messages
<input type="checkbox"/> Loot HUD
<input type="checkbox"/> Exp Statistics
<input type="checkbox"/> Exp Gain Informations
<input type="checkbox"/> Auto Show Statistics
<input type="button" value="Reset All Statistics"/>

HUD and Informations

Some HUD to make the game a little more fun

New protector	
Kind	Moved <input type="button" value="v"/> 0
Name	<input type="text"/>
<input type="checkbox"/> Pause All Bot	
<input type="checkbox"/> Pause Healers	
<input type="checkbox"/> Pause Basic Tools	
<input type="checkbox"/> Pause Healer Tools	
<input type="checkbox"/> Pause Automation Tools	
<input type="checkbox"/> Pause War Tools	
<input type="checkbox"/> Pause Advanced Tools	
<input type="checkbox"/> Cavebot Label	

Protector

You will need some help to sleep in peace

Introdução ao AngularJS

Introdução, material e
ambiente

AngularJS



Superheroic JavaScript MVW Framework

O que é AngularJS?

- **É uma Framework para Javascript**
- **Implementa o padrão MVC**
- **Foco em Single Page Applications (SPA)**
- **É fácil e divertido**
- **Estende o HTML**

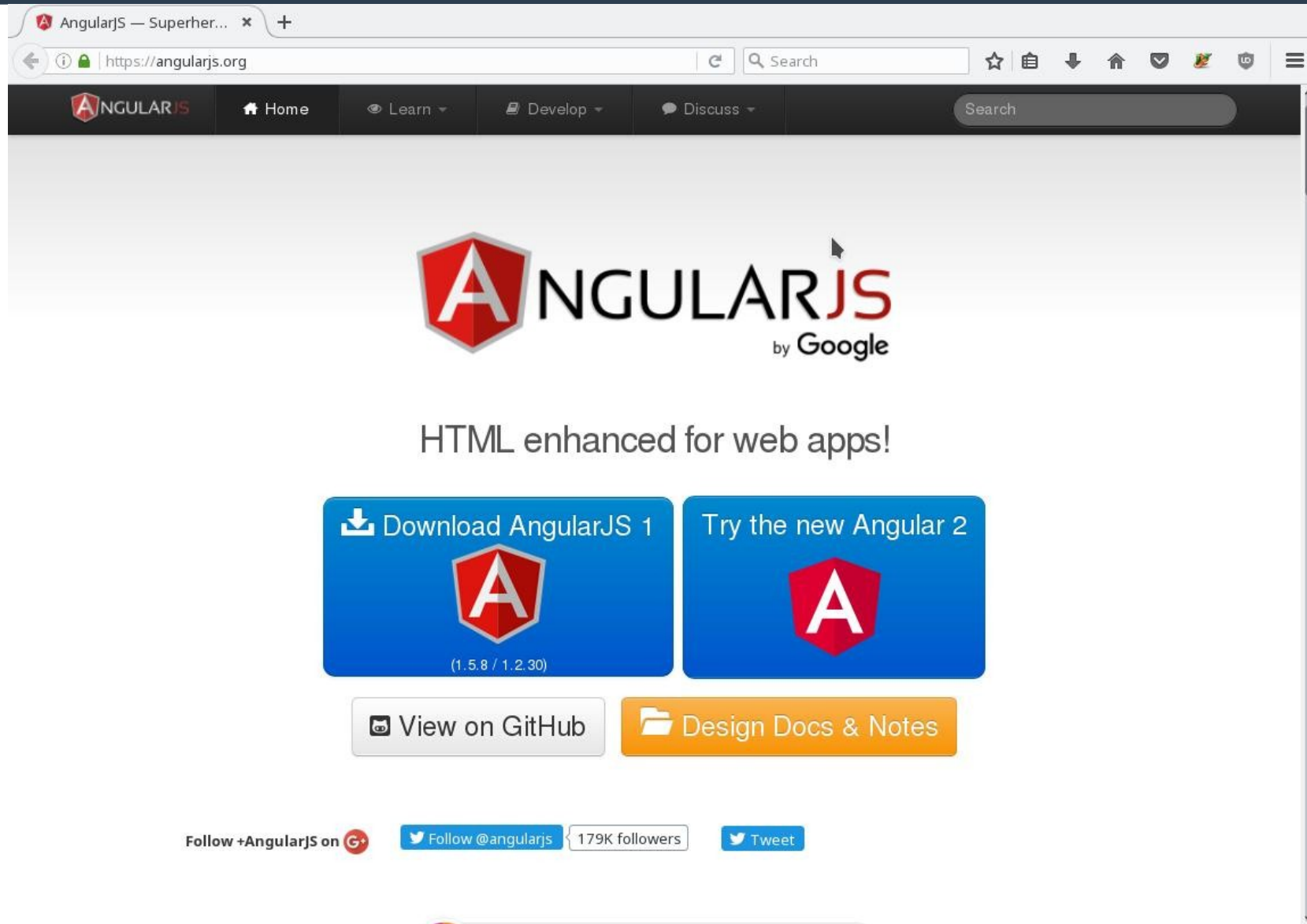
Para que aprender AngularJS?

- **Alta demanda no mercado de trabalho**
- **Simplifica a vida dos programadores**
- **Websites mais poderosos com menor esforço**
- **Tem uma comunidade imensa, é fácil pedir e prover ajuda**

Materiais necessários para o curso

- **Servidor HTTP**
- **Editor de textos**
- **Navegador (preferencialmente Chrome)**
- **AngularJS**
- **Conhecimento básico de logica de programação e Javascript**

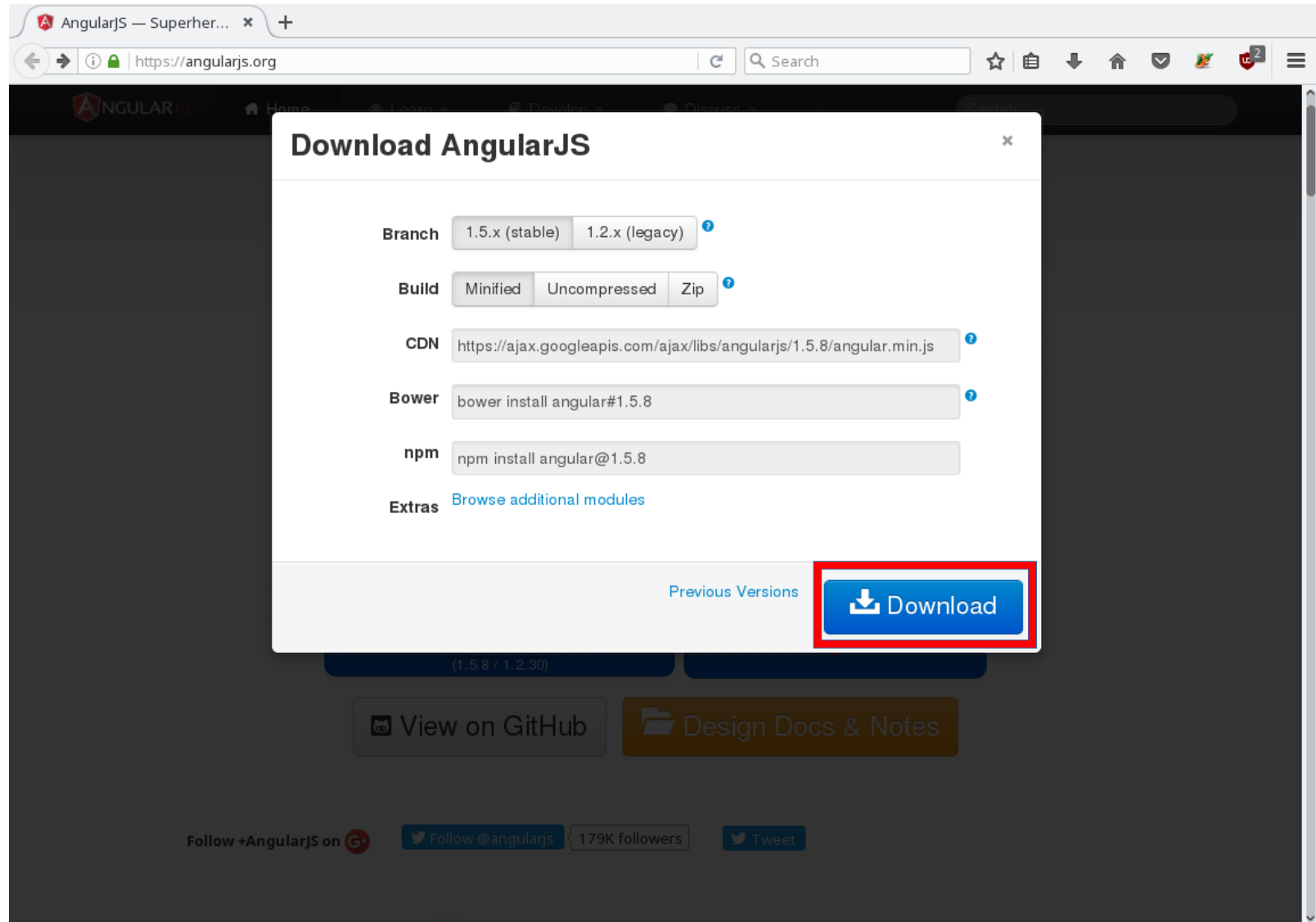
http://angularjs.org/



Download AngularJS 1



Fazer download

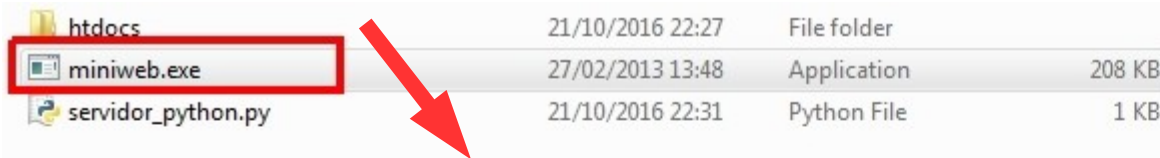


Ou simplesmente

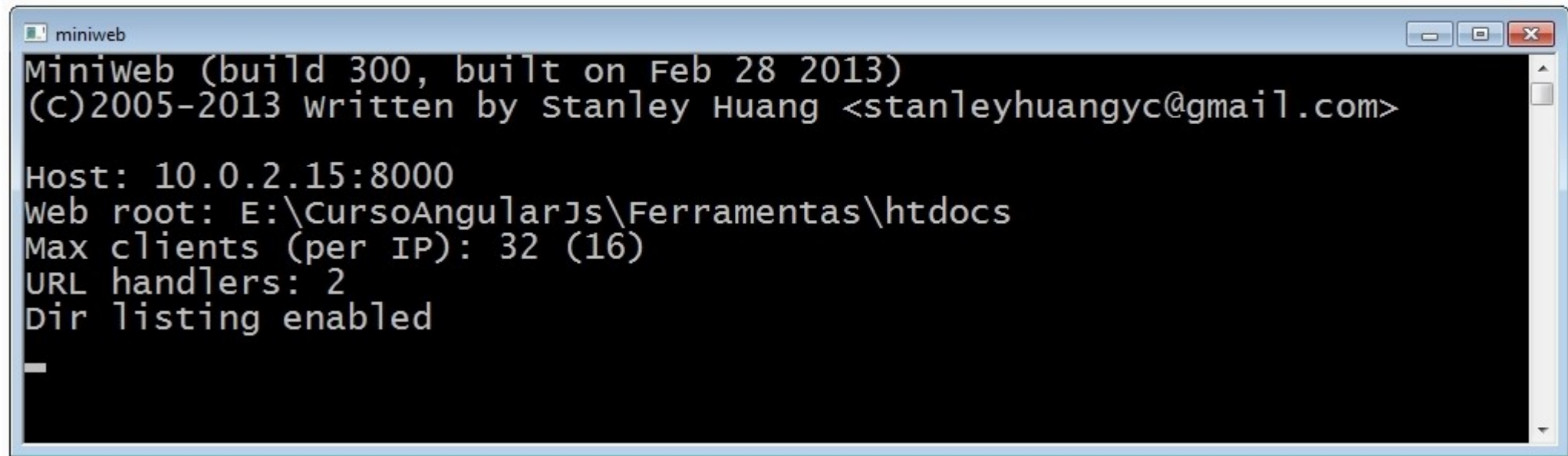
<http://baroni.tech/curso-angular>

Iniciar o servidor HTTP

Para Windows:



htdocs	21/10/2016 22:27	File folder	
miniweb.exe	27/02/2013 13:48	Application	208 KB
servidor_python.py	21/10/2016 22:31	Python File	1 KB



```
Miniweb (build 300, built on Feb 28 2013)
(c)2005-2013 written by Stanley Huang <stanleyhuangyc@gmail.com>

Host: 10.0.2.15:8000
Web root: E:\CursoAngularJs\Ferramentas\htdocs
Max clients (per IP): 32 (16)
URL handlers: 2
Dir listing enabled
```

Para Linux/MacOS:

```
fernando at fernando in ~/CursoAngularJs/Ferramentas using
python2.7 servidor_python.py
serving at 0.0.0.0 : 8000
```

Criando a aplicação AngularJS

Temos nosso template HTML index.html:

```
1 <!DOCTYPE html>
2 <html lang="pt">
3   <head>
4     <meta charset="utf-8"/>
5     <title>AngularJS</title>
6   </head>
7   <body>
8     Olá, mundo!
9   </body>
10 </html>
```

Como transformar um HTML básico em uma aplicação AngularJS?

Etapa 1

Adicionamos o script AngularJS

```
1 <!DOCTYPE html>
2 <html lang="pt">
3   <head>
4     <meta charset="utf-8"/>
5     <title>AngularJS</title>
6     <script src="angular.min.js"></script>
7   </head>
8   <body>
9     Olá, mundo!
10  </body>
11 </html>
```

Etapa 2

Adicionamos a diretiva ngApp

```
1 <!DOCTYPE html>
2 <html lang="pt" ng-app>
3   <head>
4     <meta charset="utf-8"/>
5     <title>AngularJS</title>
6     <script src="angular.min.js"></script>
7   </head>
8   <body>
9     Olá, mundo!
10  </body>
11 </html>
```

Essa diretiva marca o início de uma aplicação AngularJS

Etapa 3

Testamos o AngularJS

```
1 <!DOCTYPE html>
2 <html lang="pt" ng-app>
3   <head>
4     <meta charset="utf-8"/>
5     <title>AngularJS</title>
6     <script src="angular.min.js"></script>
7   </head>
8   <body>
9     Olá, mundo! 5 + 8 = {{5 + 8}}
10  </body>
11 </html>
```

Usamos uma simples expressão matemática para provar que o AngularJS esta funcionando

Não trabalhar com protocol file://

Protocolo file://

ERRADO



Protocolo http://

CORRETO



AngularJS no protocolo file:// faz aparecer bugs obscuros.

Etapa 4

Rodamos nossa aplicação



Olá, mundo! $5 + 8 = 13$

Sucesso!

Diretivas

Definição de Diretiva e as mais comuns

O que são Diretivas?

Diretivas são marcas que determinam o comportamentos. Comportamentos como estilos, EventListeners e até mesmo transformações para outros elementos.

Tags	<code><my-dir></my-dir></code>
Atributos	<code></code>
Classes CSS	<code></code>
Comentários HTML	<code><!-- directive: my-dir exp --></code>

Normalização nos nomes das Diretivas

O padrão de nomes no Angular é o camelCase, porém, no HTML, as diretivas seguem o formato separado-por-traços.

Exemplo: ngApp → ng-app.

O \$compilador do AngularJS também suporta outros formatos:

- Usando traços: `` (mais comum)
- Usando dois-pontos: ``
- Usando underscore: ``
- Prefixado com “data-”: ``
- Prefixado com “x-”: ``

Vamos conhecer algumas das diretivas mais comuns, e logo em seguida aplicá-las em alguns exercícios.

Expressões e ngInit

ngInit inicializa variáveis;

Expressões permitem executar comandos no HTML

```
1      <div ng-init="  
2          mensagem='Olá Mundo!';  
3          resposta_do_universo=42;  
4          pessoa={nome: 'Fernando', sobreNome: 'Rodrigues Baroni', idade: 10}  
5      ">  
6          Expressões matemáticas: {{10*(1+2)}} <br />  
7          Two-way binding: {{mensagem}} <br />  
8          Single-time binding: {{::resposta_do_universo}} <br />  
9          Objetos: {{pessoa.sobreNome}}  
10     </div>
```

É possível fazer uma expressão ser executada apenas uma vez, adicionando dois pontos duas vezes, no início da expressão. Exemplo: {{::nome}}

Filters

Filtros permitem transformar expressões

```
1  {{ "heLl0" | uppercase }} <BR/>
2  {{ "w0rLD" | lowercase }} <BR/>
3  {{ 123456789 | number }} <BR/>
4  {{ [1,8,2,4,7,5] | limitTo: 3 }} <BR/>
5  {{ ['John', 'Maria', 'Jose'] | filter:'J' }}
```

Podem formatar números, datas, filtrar coleções e até mesmo I18N

ngModel

**Vincula um componente de entrada,
para uma variável no escopo**

```
1      <input type="text" placeholder="Nome" ng-model="nome"/>
2
3      <input type="datetime-local" placeholder="Data de nascimento"
4            ng-model="nascimento"/>
5
6      <label><input type="checkbox" ng-model="temFilhos"/>
7            Tem filhos?</label>
```

ngBind

Vincula um elemento qualquer a uma variável do escopo

```
1      Seu nome é <span ng-bind="nome"></span>.  
2  
3      Seu nome é {{nome}}.
```

É parecido a uma `{{expressão}}` e *quase* sempre dá o mesmo resultado.

ngHide, ngShow e ngIf

Mostra ou esconde elementos baseados em uma expressão booleana

```
1 <p ng-hide="expr"></p>
2
3 <span ng-show="expr"></span>
4
5 <div ng-if="expr"></div>
```

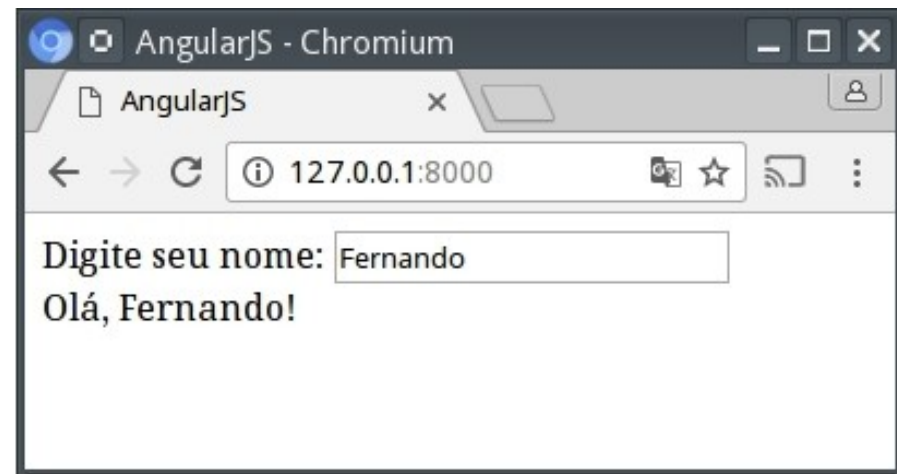
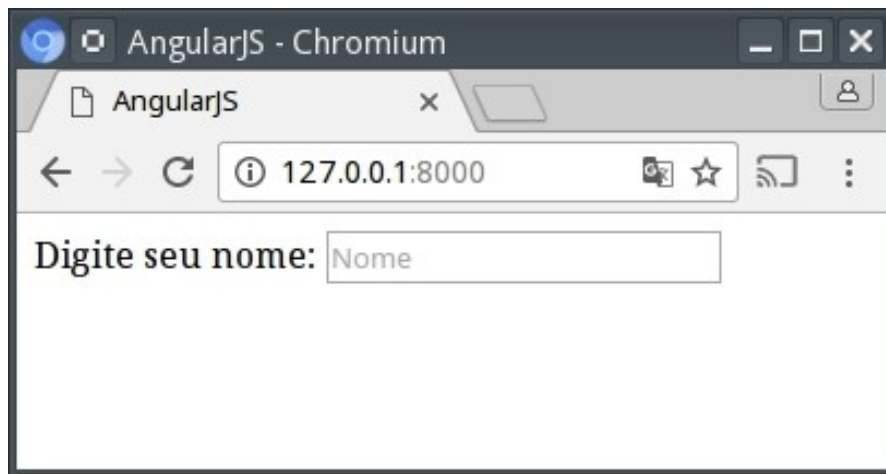
Atenção: ngIf não é igual ao ngShow e ngHide!

ngIf adiciona ou remove o elemento do DOM.

ngHide e ngShow mostra ou esconde o elemento usando CSS.

Desafio: Hello, {{nome}}!

Usando as diretivas aprendidas, fazer uma aplicação Hello World usando models e condicionais para mostrar a mensagem “Olá Mundo” apenas quando o nome for atribuído



Desafio: Hello, {{nome}}!

```
1 <!DOCTYPE html>
2 <html lang="pt" ng-app>
3   <head>
4     <meta charset="utf-8"/>
5     <title>AngularJS</title>
6     <script src="angular.min.js"></script>
7   </head>
8   <body>
9     Digite seu nome:
10     <input type="text" placeholder="Nome" ng-model="nome"/>
11
12     <div ng-show="nome">
13       Olá, {{nome}}!
14     </div>
15   </body>
16 </html>
```

ngClass

Gerencia classes CSS de um elemento, ativando ou desativando a classe condicionalmente.

```
1 <p ng-class="  
2 {  
3   'css_sucesso': com_sucesso  
4   'css_falha': !com_sucesso  
5 }  
6 ">  
7   ng-Class  
8 </p>
```

ngRepeat com lista

Replica um template para cada elemento de uma coleção

```
1 <div ng-init="
2   listaPessoas = ['Fernando', 'João', 'Maria', 'Ricardo', 'Marcelo']
3   ">
4   <p ng-repeat="nome in listaPessoas">
5     A pessoa {{$index + 1}} se chama {{nome}}
6   </p>
7 </div>
```

ngRepeat com dicionário ou objeto

Replica um template para cada elemento de uma coleção

```
1      <div ng-init="
2          objetoFernando = {
3              nome: 'Fernando',
4              sobreNome: 'Rodrigues Baroni',
5              idade: 23}
6      ">
7      <p ng-repeat="(chave, valor) in objetoFernando">
8          A chave '{{chave}}' tem valor '{{valor}}'
9      </p>
10     </div>
```

Controllers

**Como criar um controller,
\$scope e Dependency
Injection.**

Adicionar um novo arquivo para trabalho

Primeiro, vamos adicionar um novo script em nossa aplicação, para organizar melhor nosso código.

A nossa Controller, vai ficar nesse novo arquivo, desta forma, mantemos a View e a lógica da nossa aplicação separadas.

Etapa 1

Adicionamos um script em nossa aplicação

```
1 <!DOCTYPE html>
2 <html lang="pt" ng-app>
3   <head>
4     <meta charset="utf-8"/>
5     <title>AngularJS</title>
6     <script src="angular.min.js"></script>
7     <script src="app.js"></script>
8   </head>
9   <body>
10     Olá, mundo!
11   </body>
12 </html>
```

Etapa 2

Atribuámos um nome para a aplicação AngularJS

```
1 <!DOCTYPE html>
2 <html lang="pt" ng-app="cursoNg123">
3   <head>
4     <meta charset="utf-8"/>
5     <title>AngularJS</title>
6     <script src="angular.min.js"></script>
7     <script src="app.js"></script>
8   </head>
9   <body>
10     Olá, mundo!
11   </body>
12 </html>
```

Etapa 3

Iniciamos o nosso modulo AngularJS

```
1 (function() {  
2  
3     angular.module('cursoNg123', []);  
4  
5 })();
```

Atenção: o nome do modulo no arquivo app.js deve bater com o nome da diretiva ng-app na view (html). O nome não pode começar com “ng”

ng-app="cursoNg123"> → angular.module('cursoNg123', []);

Criando Controller

Agora que temos um arquivo Javascript para trabalhar, também temos o nosso modulo AngularJS, podemos finalmente criar o nosso Controller.

```
1 angular.module(...).controller(  
2     'NomeDoController',  
3     ['dependencia', 'dependencia2',  
4         function(dependencia, dependencia2) {  
5             //Codigo do Controller  
6         }  
7     ]);
```

Injeção de Dependências

O AngularJS tem um sistema de Injeção de Dependências:

```
1 angular.module(...).controller(  
2     'NomeDoController',  
3     ['dependencia', 'dependencia2',  
4         function(dependencia, dependencia2) {  
5             // Código do Controller  
6         }  
7     ]);
```

Talvez você se pergunte, por que nos repetimos as dependências, uma vez na lista de dependências e outra nos argumentos da função do Controller?

Falando em dependências: \$scope

A dependência mais comum é o \$scope.

O \$scope, é um objeto de contexto, que permite que a view (o HTML) leia dados da Controller. Quando queremos expor alguma variável do Controller, ou até mesmo uma função, nos exportamos esses para o \$scope, assim a view pode usar esses objetos exportados.

Olá Mundo, Controller!

Vamos praticar o uso de Controllers com o clássico: 'Hello World':

```
1 (function() {  
2  
3     var app = angular.module('cursoNg123', []);  
4  
5     app.controller('MeuControlador', ['$scope', function($scope) {  
6         $scope.mensagem = "Olá, Controller!";  
7     }]);  
8 })();
```

Injetando a Controller na view

```
1 <!DOCTYPE html>
2 <html lang="pt" ng-app="cursoNg123">
3   <head>
4     <meta charset="utf-8"/>
5     <title>AngularJS</title>
6     <script src="angular.min.js"></script>
7     <script src="app.js"></script>
8   </head>
9   <body>
10    <div ng-controller="MeuControlador">
11      {{mensagem}}
12    </div>
13  </body>
14 </html>
```

É necessário injetar a Controller para usarmos a mesma na View, fazemos isso com a seguinte diretiva: ngController

\$scope na view

Quando injetamos o Controller, temos acesso ao \$scope que o Controller nos expôs!

```
1 <!DOCTYPE html>
2 <html lang="pt" ng-app="cursoNg123">
3   <head>
4     <meta charset="utf-8"/>
5     <title>AngularJS</title>
6     <script src="angular.min.js"></script>
7     <script src="app.js"></script>
8   </head>
9   <body>
10    <div ng-controller="MeuControlador">
11      {{mensagem}}
12    </div>
13  </body>
14 </html>
```

Exportando funções

Podemos também exportar funções para o \$scope

```
8     $scope.somar = function(a, b) {  
9         return a + b;  
10    };
```

E chamar as funções exportadas, usando expressões Angular

```
10    <div ng-controller="MeuControlador">  
11        {{mensagem}} <br/>  
12        A soma de 123 + 456 = {{ somar(123, 456) }}  
13    </div>
```

Diretiva Customizada

Como criar uma diretiva

Como criar uma diretiva basica

Criar diretivas é muito semelhante a criar Controllers:

```
1  angular.module(...).directive('minhaDiretiva', function() {  
2      return {  
3          restrict: 'A',  
4          template: "" +  
5              "<p>" +  
6              "    <span>{{pessoa.nome + ' ' + pessoa.sobrenome}}</span>" +  
7              "</p>"  
8      };  
9  });
```

Diretivas são criadas, fazendo uma função que retorna um objeto.

Atributo: Template e TemplateURL

Template

Define o HTML apresentado no navegador.

Atributo: Template e TemplateURL

TemplateURL

Define uma URL que contem o template HTML para diretiva.

Atributo: Restrict

Usado para definir como nossa diretiva vai ser usada no HTML, sendo uma <TAG> ou um Atributo. Pode ser uma combinação das seguintes letras:

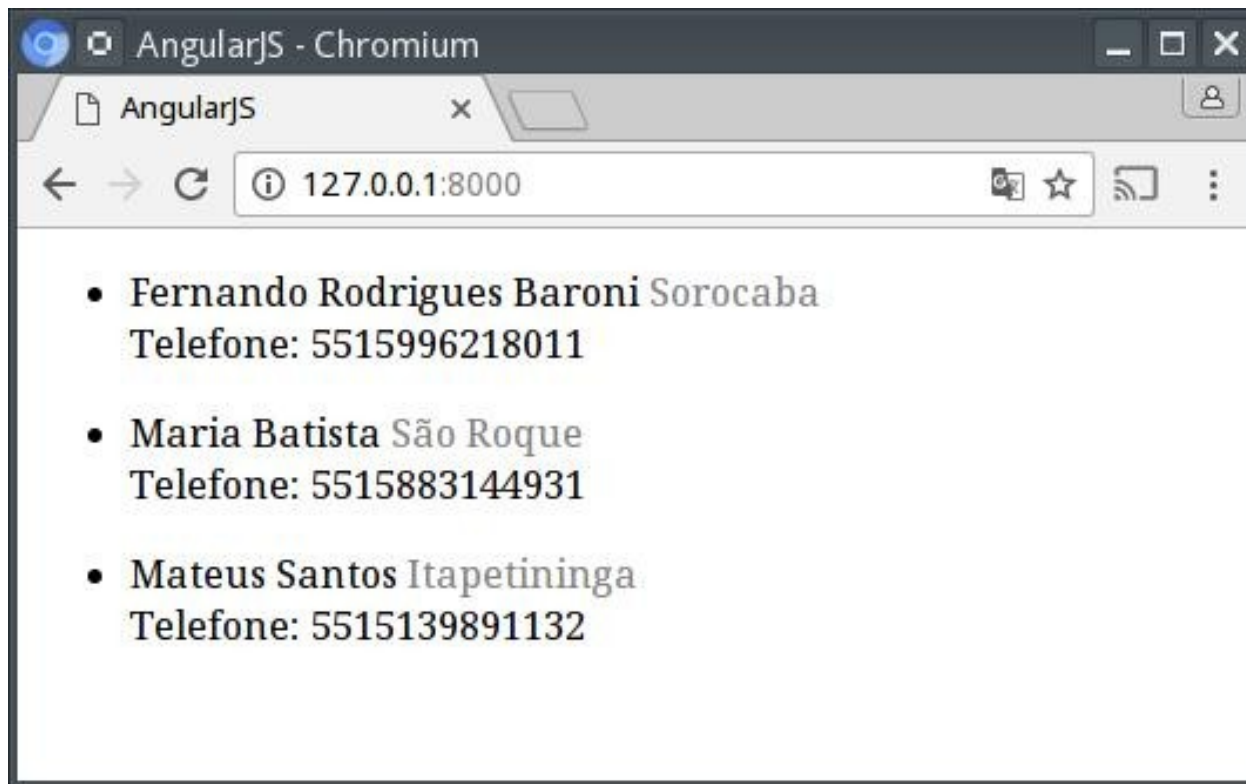
E	Tags	<code><my-dir></my-dir></code>
A	Atributos	<code></code>
C	Classes CSS	<code></code>
M	Comentários	<code><!-- directive: my-dir exp --></code>

Exemplos: E → como tag EA → como tag ou atributos

Desafio

Agenda de Telefone

Faça uma pequena aplicação para exibir uma agenda telefônica.



Requisito 1: Controller

Crie um Controller que inicializa uma lista de objetos, cada objeto vai ter os seguintes atributos: nome, sobreNome, telefone e cidade.

Exemplo:

```
1  {  
2      nome: 'Fernando',  
3      sobrenome: 'Rodrigues Baroni',  
4      telefone: 5515996218011,  
5      cidade: 'Sorocaba'  
6  }
```

Requisito 2: Diretiva

Na view, use alguma diretiva para desenhar todos os contatos registrados no escopo, utilizando a diretiva criada

```
1 app.directive('meuContato', function() {  
2     return {  
3         restrict: 'A',  
4         template: "" +  
5             "<p>" +  
6             "    Mostrar dados do contato" +  
7             "</p>"  
8     };  
9 });
```


GFT

Quer se tornar um GFT Hero?

Benefícios

- Carreira progressiva
- Horário flexível
- Vale Refeição/Vale Alimentação
- Plano de Saúde
- Curso de Inglês gratuito
- Projetos internacionais
- Viagens internacionais
- Café de graça
- Frutas de graça toda segunda-feira e quarta-feira
- Fliperama, Playstation e Xbox 360

Shaping the future of digital business

GFT Sorocaba
Michel Alves
Desenvolvedor Senior
Av. São Francisco, 98 Jardim Sta. Rosália
Sorocaba – SP

Michel.Alves@gft.com

Fim

Obrigado!