

# **Projeto e Desenvolvimento de Software II**

**Prof. Carlos Eduardo de Carvalho Dantas**

**carloseduardodantas@iftm.edu.br**

# **Parte I – Técnicas e Ferramentas para Desenvolvimento de Sistemas**

# AGENDA

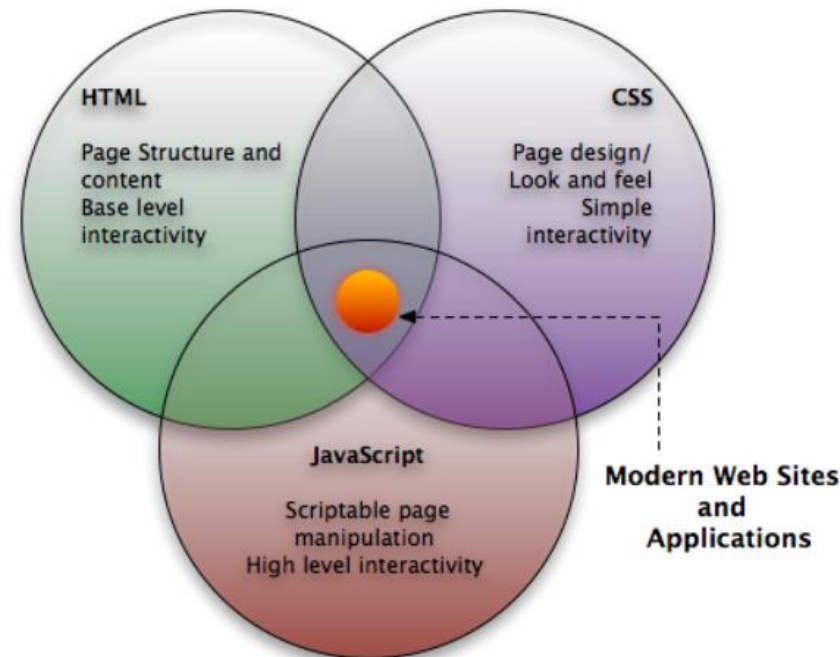
---

## 1. Desenvolvimento Front-End

- Histórico
- Aplicações RIA
- Frameworks front-end
- Modelo MVW
- Web Responsiva
- Escalabilidade e websockets

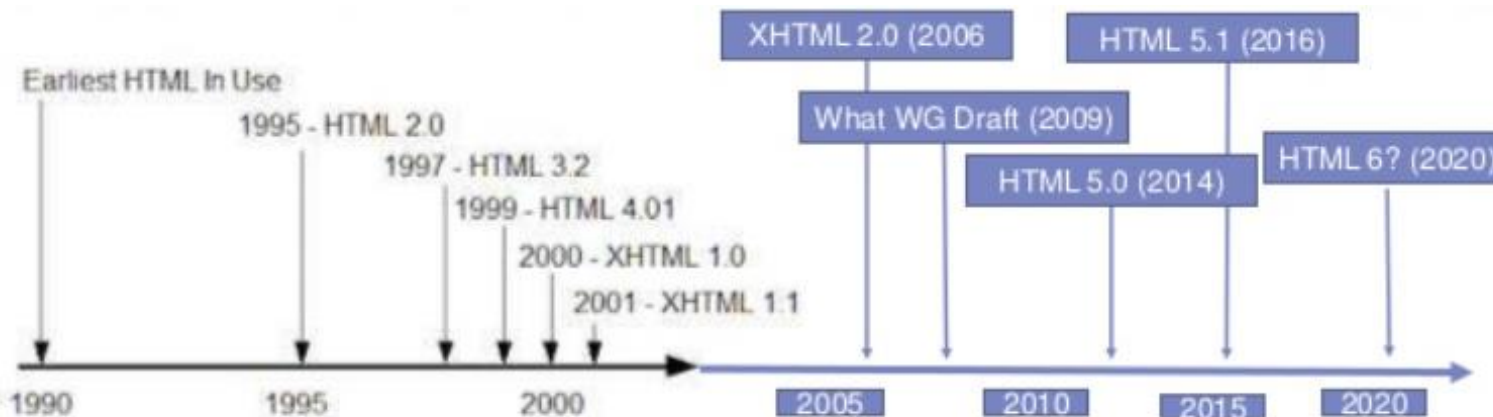
## - Definição

- Sistemas de documentos *Hipermidia* que são interligados e executados na *internet*.
- Documentos *Web* são uma combinação de *HTML*, *JavaScript* e *CSS*.



## - HTML (*HyperText Markup Language*)

- É controlado desde 1995 pelo *W3C*, e desde 2009 trabalha em conjunto com *WHATWG* na confecção do *HTML 5*. Desde a primeira versão, cada atualização adicionou diversas tags, desde a adição de *hyperlinks*, formulários, dentre outros.
- Com o *HTML 5*, ferramentas como *Silverlight* e *Flash* perderam parte de sua utilidade, pois esta versão suporta a maior parte dos recursos que antes só eram possíveis com *plug-ins*. *Exemplos: exibição de vídeos, gráficos 2d ou 3d, etc..*



## - CSS(Cascading Style Sheets)

- Foi criado para controlar a aparência dos documentos HTML, formatando a informação.
- O CSS 3 **multiplicou as possibilidades** de se utilizar estilos na criação de páginas. Exemplos: propriedades como *box-shadow* e funções como *linear-gradient* permitiram a criação de formas, cores e efeitos das páginas diretamente no código, sem precisar de criar imagens para tal.

```
<html>
  <head>
    <style>
      .tnt {
        border-radius: 50%;
        border: 5px solid #000;
        height: 50px;
        line-height: 50px;
        text-align: center;
        width: 50px;
      }
    </style>
  </head>
  <body>
    <h3 class=' tnt' >TNT</h3>
  </body>
</html>
```



## - JavaScript

- Começou como uma “*simples*” linguagem *client-side* para operações simples como validação de formulários.
- O uso primário de *JavaScript* é escrever funções que são embarcadas ou incluídas em páginas *HTML* e que interagem com o *Modelo de objeto nos documentos* (DOM) da página.
- Recursos como *Ajax*, tipagem dinâmica, avaliação em tempo de execução, protótipos, objetos *Json*, dentre outros transformaram *JavaScript* na linguagem mais popular da *web*.

## - DOM (Domain Object Model)

- É uma árvore de elementos da página *HTML* criada, ou um espelho em memória de toda a página criada no navegador quando esta é carregada. Possui funções e propriedades que, ao serem modificadas, disparam alterações instantâneas no que é exibido ao usuário.
- Sendo assim, é possível manipular o *DOM* utilizando código *JavaScript*, ou seja, estes scripts possuem capacidade de modificar dinamicamente o *DOM*.

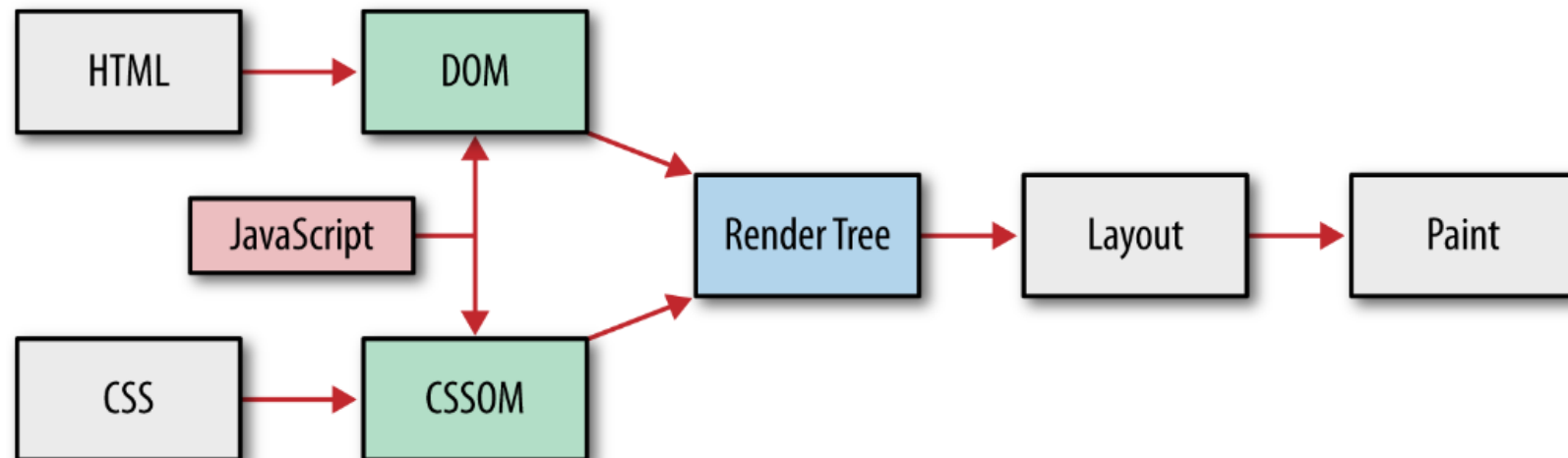
```
<!-- página html -->  
<button class="botao-grava">Novo</button>  
<p class="contatos">Contatos cadastrados: 0 </p>
```

```
var contatos = document.querySelector('.contatos');  
var total = 0;  
var botao = document.querySelector('.botao-grava');  
botao.addEventListener('click', function(event) {  
    total++;  
    contatos.textContent = 'Contatos cadastrados: ' + total;  
});
```



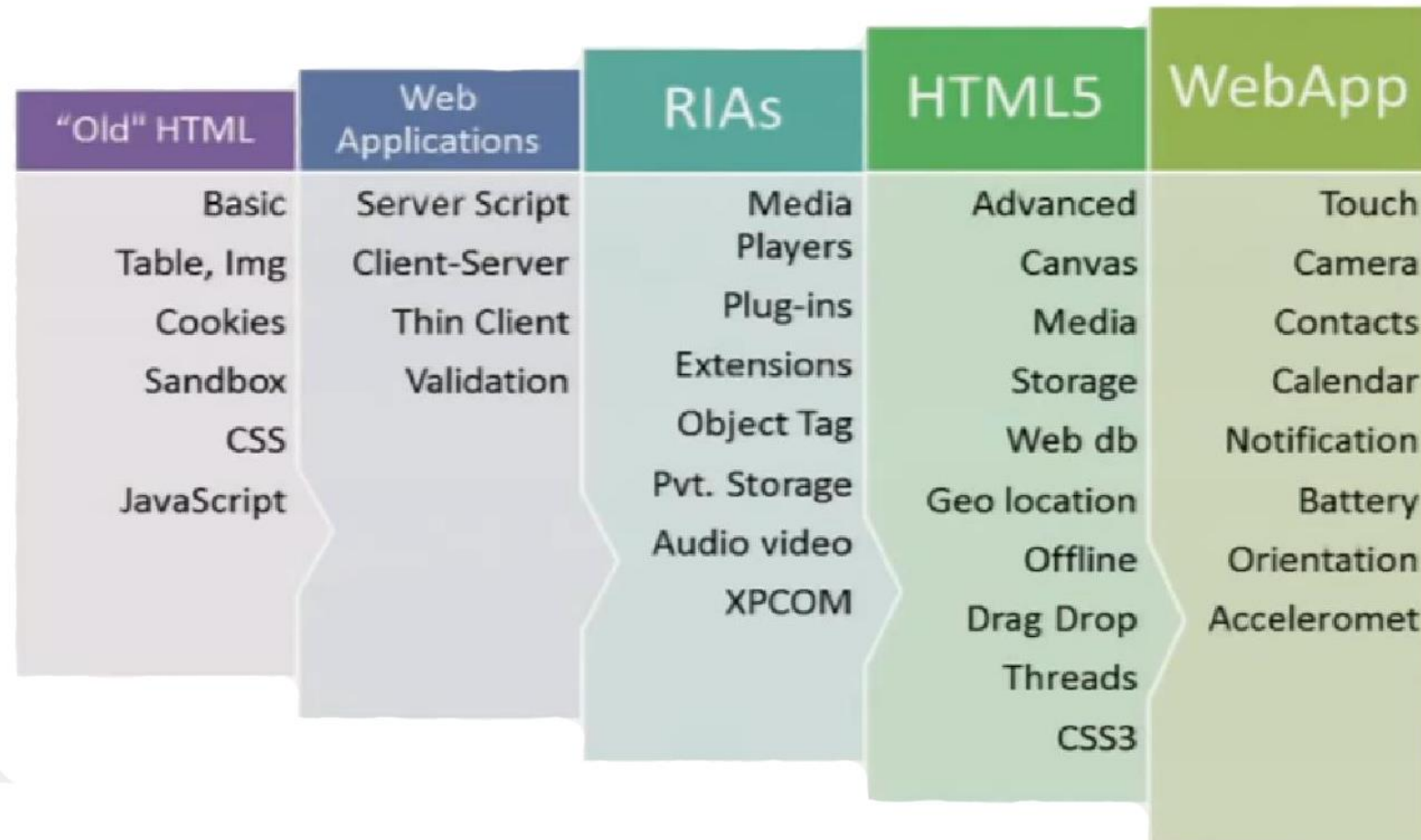
## - Pipeline de Processamento do Browser

- O *parsing* do documento *HTML* constrói o *DOM*. As regras e recursos das folhas de estilos constroem o *CSSOM*. A combinação de ambos formam a árvore de renderização, onde o navegador possui informação suficiente para construir o layout e mostrar algo na tela.
- A execução de *Scripts* podem editar alguma informação do *DOM*. *Scripts* também podem perguntar por estilos de quaisquer objetos, acessando a *CSSOM*.



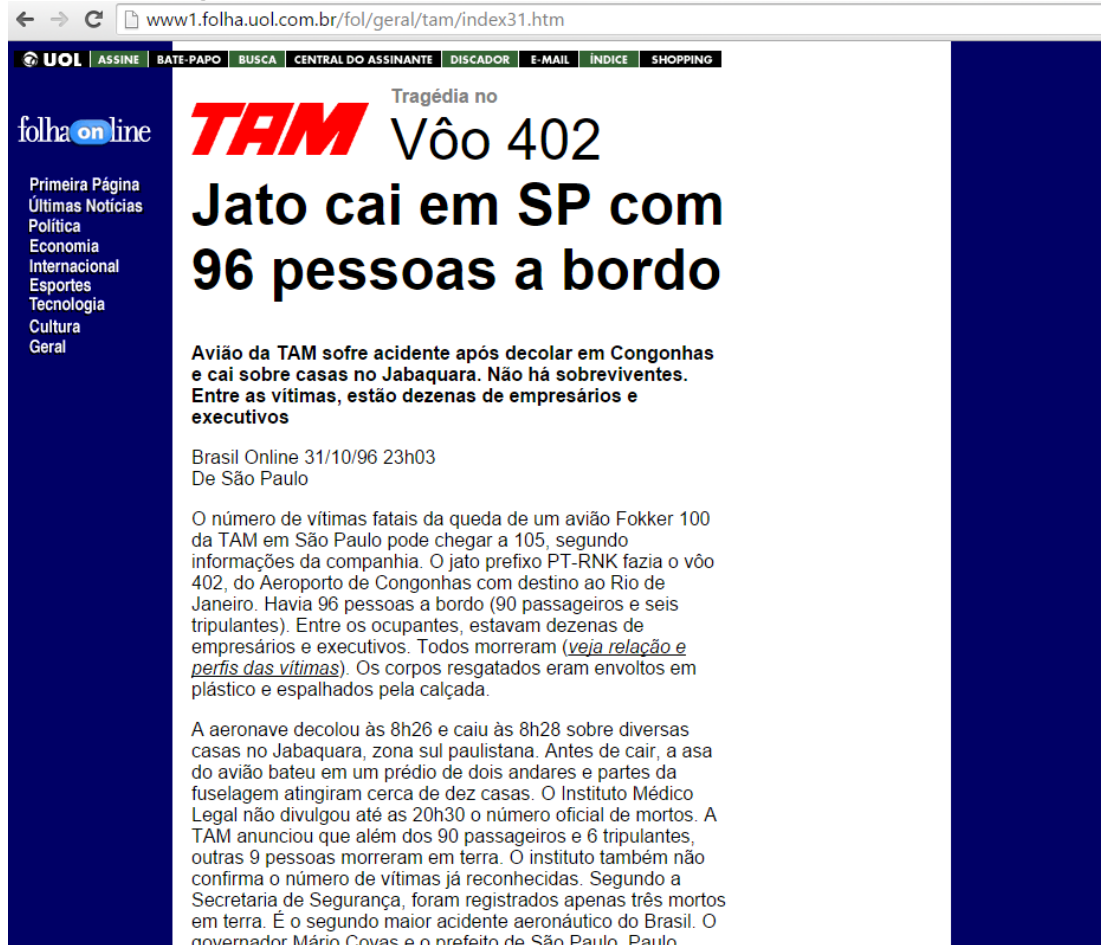
# Evolução de tecnologias Web

## - Evolução



# Evolução de tecnologias Web

## - Evolução – Old HTML



www1.folha.uol.com.br/fol/geral/tam/index31.htm

UOL ASSINE BATE-PAPO BUSCA CENTRAL DO ASSINANTE DISCADOR E-MAIL ÍNDICE SHOPPING

folha online

Primeira Página  
Últimas Notícias  
Política  
Economia  
Internacional  
Esportes  
Tecnologia  
Cultura  
Geral

Tragédia no  
**TAM** Vôo 402

### Jato cai em SP com 96 pessoas a bordo

Avião da TAM sofre acidente após decolar em Congonhas e cai sobre casas no Jabaquara. Não há sobreviventes. Entre as vítimas, estão dezenas de empresários e executivos

Brasil Online 31/10/96 23h03  
De São Paulo

O número de vítimas fatais da queda de um avião Fokker 100 da TAM em São Paulo pode chegar a 105, segundo informações da companhia. O jato prefixo PT-RNK fazia o voo 402, do Aeroporto de Congonhas com destino ao Rio de Janeiro. Havia 96 pessoas a bordo (90 passageiros e seis tripulantes). Entre os ocupantes, estavam dezenas de empresários e executivos. Todos morreram ([veja relação e perfis das vítimas](#)). Os corpos resgatados eram envoltos em plástico e espalhados pela calçada.

A aeronave decolou às 8h26 e caiu às 8h28 sobre diversas casas no Jabaquara, zona sul paulistana. Antes de cair, a asa do avião bateu em um prédio de dois andares e partes da fuselagem atingiram cerca de dez casas. O Instituto Médico Legal não divulgou até as 20h30 o número oficial de mortos. A TAM anunciou que além dos 90 passageiros e 6 tripulantes, outras 9 pessoas morreram em terra. O instituto também não confirma o número de vítimas já reconhecidas. Segundo a Secretaria de Segurança, foram registrados apenas três mortos em terra. É o segundo maior acidente aeronáutico do Brasil. O governador Mário Covas e o prefeito de São Paulo Paulo



novos inclusões **cadê?** eventos veja ?

Consulta  Busca [Informações](#)



[Ciência e Tecnologia](#)  
[Institutos, Centros de Pesquisa](#)

[Cultura](#)  
[Museus, Música, Personalidades](#)

[Esportes](#)  
[Automobilismo, Futebol](#)

[Governo](#)  
[Estados, Prefeituras](#)

[Compras](#)  
[CDs, Flores, Livros](#)

[Educação](#)  
[Cursos, Escolas, Universidades](#)

[Finanças](#)  
[Bancos, Bolsas, Seguros](#)

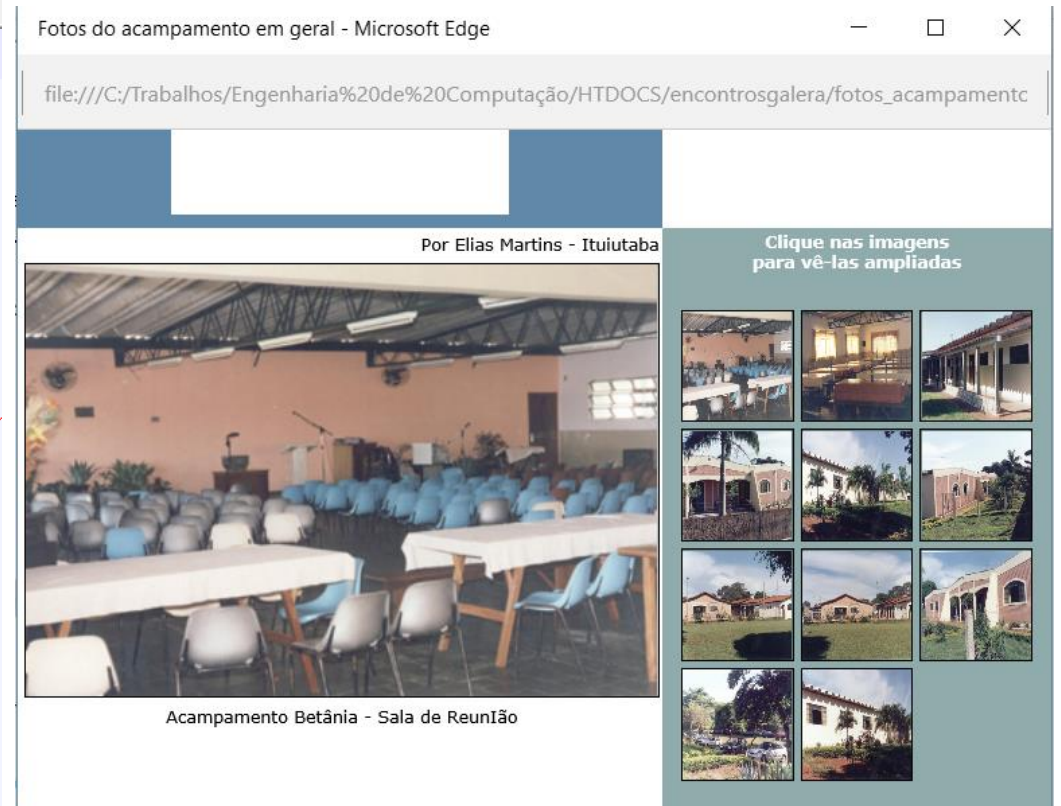
[Indústria e Comércio](#)  
[Automóveis, Telecomunicações](#)

## - Evolução – Old HTML

```
fotos_acampamento.HTM
<SCRIPT>
  id_album = "fotos_acampamento";
  thumbs = 11; //numero de fotos no total
  fotoabre =6; // foto padrao que abre
  imgs_roothpath = "fotospeq"
  imgs_rootpath2 = "fotos_acampamento"
  reverso = 1;
  dropdown_todosalbuns = "selecao.htm"; //caixa de s
  banner = "logomarca";
  abreurl = 0;
  naodaravolta = 0; // coloca Anterior e Proximo

  y = x = 1 ;
  legenda=new Array();
  credito=new Array();

  credito[x++] = "Por Elias Martins - Ituiutaba";
```



## - Evolução – Web Application PHP

```
INDEX.PHP
<?
include "conecta.php";
include "funcoes.php";

$ip = getenv ("REMOTE_ADDR");
$visitas = numero_visitas();
$visitante = "visitante".$visitas;
$inicio_sessao = mysql_query("select now() as hora_atual");
$inicio_sessao = mysql_result($inicio_sessao,0,'hora_atual');
@mysql_query("insert into visitas (visitante,link_atual,ip,hora_in
$sql_ultimo = mysql_query("select encontro,id,date_format(data_ini
$quantos = mysql_num_rows($sql_ultimo);

?>
<html>
<head>
<title> Encontros da Galera Online!!!!!!
<tr>
<td>
<table border=1 cellspacing=5 cellpadding=0 align='center' width='130' class=
<tr bgcolor='A7DAFA'>
<td colspan='2'>
<a href='fotos.php?id=? echo $id ?>&visitante=? echo $visitante?>
</td>
```

## - Evolução – Web Application Java SERVLET + JSP

```
<!-- Codigo adicionado para solucionar controle de grupos -->
<%@page import="java.security.*" %>
<%@page import="javax.security.auth.*" %>
<%
    if (!request.isUserInRole("SGP_Administradores") && !"carlosec".equals(re
        <logic:redirect forward="error403"/> <%
    }
%>

<!-- Codigo adicionado para solucionar controle de grupos -->

<html><!-- InstanceBegin template="/Templates/pgTemplateForm.dwt.jsp" codeOut
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<c:if test="false">
    <link href="/css/estilos.css" rel="stylesheet" type="text/css">
</c:if>
<link href="<%=request.getContextPath()%>/css/estilos.css" rel="styleshe
<style type="text/css">
```



## - Evolução – Web Application ASP

```
<td width="70%">
  <input type="text" class='text' name="strPesquisa" value="<%=Request("strPesquisa")%>" :
</td>
</table>

<BR><BR>

<%
strPesquisa = Request("strPesquisa")

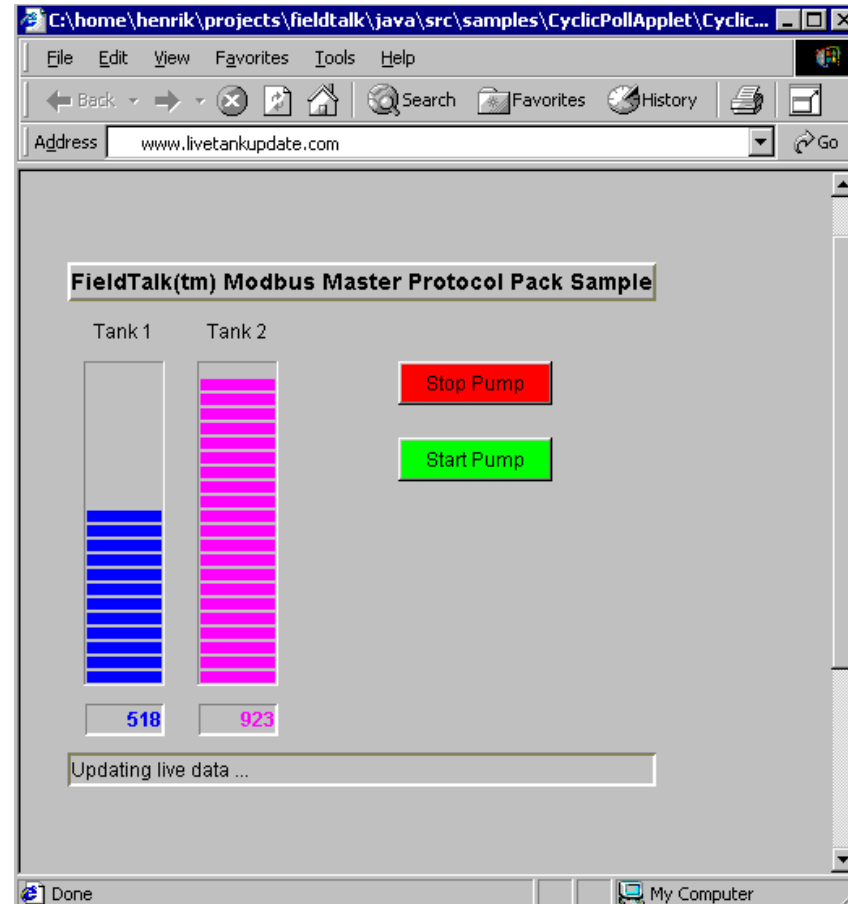
set objConn = Server.CreateObject("ADODB.Connection")
set objRecset = Server.CreateObject("ADODB.Recordset")

objConn.Open strConn

strQuery = "S_sp_Busca_Grupo_Descricao '%" & strPesquisa & "%'"
objRecset.Open strQuery, objConn
```

# Evolução de tecnologias Web

## - Evolução – RIA Applet Java





# Evolução de tecnologias Web

## - Evolução – RIA FLASH



## - Evolução – RIA JSF

HP acaoComercial.jspx

```
</h:selectOneMenu>
```

```
<h:outputText value="Validade:" id="outval" styleClass="outputText" title=
<pxt:panelGrid columnsWidth="90px;20px;*" cellspacing="0"
    cellpadding="0" id="pnlval">
    <rich:calendar inputClass="calendarTam" id="calval"
        datePattern="dd/MM/yyyy"
        disabled="#{acaoComercialBean.disabledOnDefault}"
        enableManualInput="true" inputSize="8"
        value="#{acaoComercialBean.domain.validadeNaoNula.dataInicio}">
        <a4j:support event="onchanged" id="evtcalval" />
    </rich:calendar>
    <h:outputText value="à" styleClass="outputText" id="outdat"
        style="margin: 5px;" />
    <pxt:cell align="left">
```

## - Evolução – RIA ASP.NET

```
<TR>
```

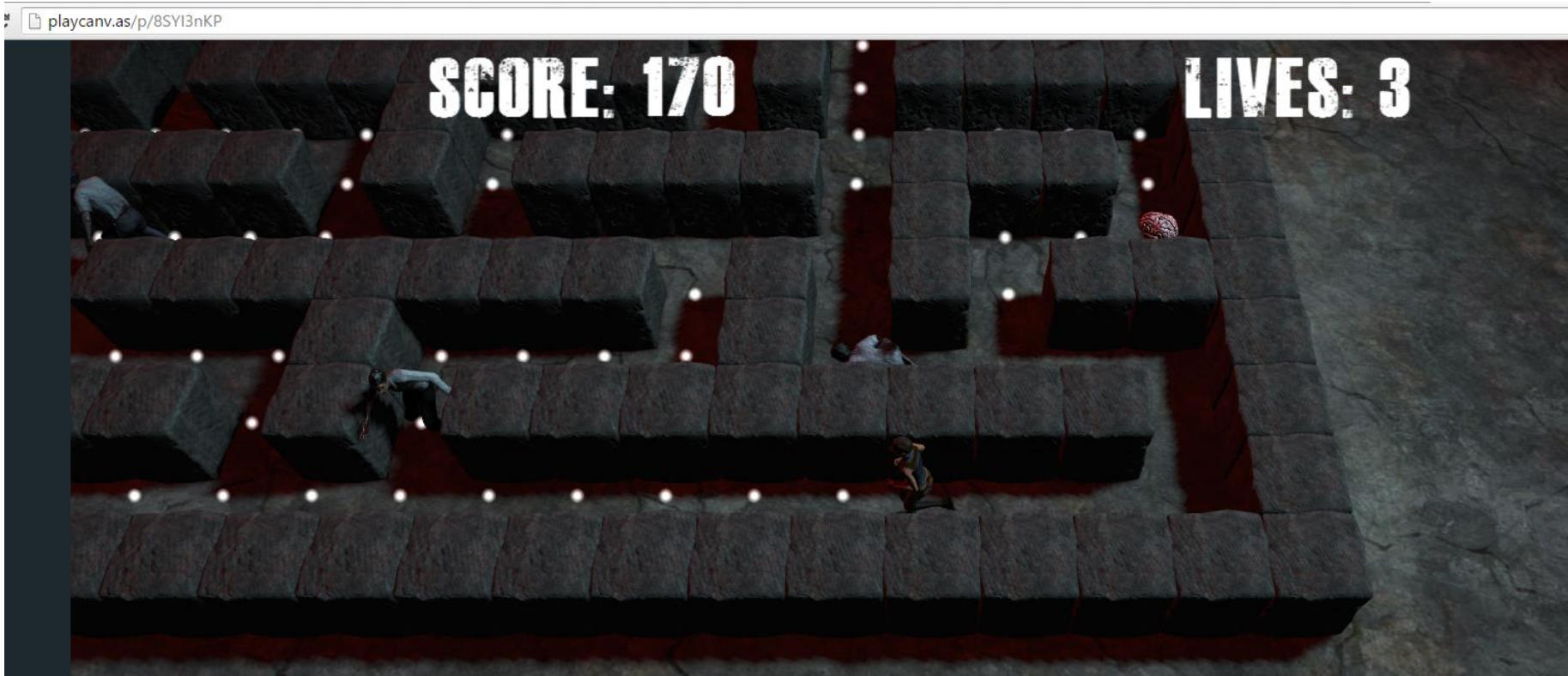
```
<TD style="WIDTH: 451px; HEIGHT: 38px"><asp:label id="Label1!"
<TD style="WIDTH: 337px; HEIGHT: 38px"><asp:radiobuttonlist
    AutoPostBack="True">
    <asp:ListItem Value="1">Sim</asp:ListItem>
    <asp:ListItem Value="0">Não</asp:ListItem>
</asp:radiobuttonlist></TD>
```

```
<TR>
```

```
<TD style="WIDTH: 451px; HEIGHT: 14px"><asp:label id="Label1!"
    ControlToValidate="txtMensagem" ErrorMessage="*"></asp:label>
    <asp:label id="Label2!" ControlToValidate="txtMensagem" ErrorMessage="*" Val
<TD style="WIDTH: 337px; HEIGHT: 14px"><asp:textbox id="txt1"
    MaxLength="255" TextMode="MultiLine" Height="50px">
```

# Evolução de tecnologias Web

## - Evolução – HTML 5



# Evolução de tecnologias Web

## - Evolução – Web App





# Evolução de tecnologias Web

---

- **Old HTML** – páginas estáticas retornadas pelo servidor. Rápido, simples, porém sem conteúdo dinâmico;
- **Web Application** – servidor processa toda a informação, devolvendo o *HTML* pronto para o navegador. Uso de *JavaScript* para funções específicas, especialmente componentes mais complexos.
- **RIA** – abstrai a construção de componentes estilosos e complexos, gerando processamento mais intenso do lado cliente. Porém, geralmente depende do servidor para controlar o estado da interface gráfica.
- **HTML 5** – basicamente eliminou ferramentas *RIA* como *Flash*, *Silverlight*, *Java FX* e *Applets*, com um padrão aberto, e rodando nativamente no navegador.
- **Web App** – o conteúdo é escrito uma vez, sendo adaptável a qualquer dispositivo.

## - Introdução

- *HTML 5*, *CSS 3* e *JavaScript* possuem diversos recursos interessantes, permitindo criar sites e aplicações com visuais elegantes. Contudo, **criar** estilos, **manipular** o *DOM*, e se **preocupar** em ter compatibilidade com *browsers* mais antigos são tarefas relativamente demoradas para se fazer sozinho, sem o auxílio de ferramentas.
- Os frameworks surgem como ferramentas **para auxiliar** neste trabalho.
- Serão mostradas nas próximas seções frameworks para diversas finalidades distintas em sistemas Web

## - Componentes Responsivos

- *Frameworks* que incluem kits com diversos componentes *web* prontos para desenvolver aplicações *web/mobile* responsivas. Está envolvido **unicamente com o desenho da tela HTML**, possuindo diversos *templates* prontos para serem utilizados.
- É possível até escrever aplicações completas **sem digitar nenhuma linha de CSS**. Por isso são tão aclamados em equipes que precisam de sistemas com telas não muito complexas, sem necessidade de tanto conhecimento em design e CSS.



Foundation  
Start here, build everywhere.



Bootstrap



## - JavaScript server-side

- Com a evolução da linguagem *JavaScript* e da sua *engine V8* (interpretador *JavaScript* criado pela *google*, que pode ser executado fora de um *browser*), servidores tradicionais como Apache e IIS ganharam novos concorrentes, que são servidores *JavaScript*.
- O *Node.js* é o servidor *JavaScript* mais conhecido. É extremamente escalável, *single threaded (non-blocking-thread)*, sem *deadlocks*, orientado a eventos, assíncrono



## - Componentes JavaScript server-side

- Assim como o *Ruby* tem o *Rails*, o *Python* tem o *Django*, o *Groovy* tem o *Grails*, o *Node.js* tem o *Meteor* ou *Express*.
- Estes frameworks ajudam na organização de uma aplicação Web MVC do lado servidor, fornecendo templates para diversas funções, como disponibilizar REST endpoints, conectar no banco de dados, etc...

express



## - Manipulação do DOM

- Uma das dificuldades de manipular o *DOM* diretamente com *JavaScript* se deve ao fato que cada navegador costuma implementar funções e propriedades de uma forma diferente, gerando incompatibilidade.
- Bibliotecas de manipulação do *DOM* foram criadas para blindar o programador das idiossincrasias do *DOM* entre os navegadores, além de adicionar recursos facilitadores. Uma das mais famosas é o *jQuery*.



## - Manipulação do DOM

- Comparativo *JavaScript* puro vs *jQuery*.

### JavaScript puro

```
var contatos = document.querySelector('.contatos');
var total = 0;
var botao = document.querySelector('.botao-grava');
botao.addEventListener('click', function(event) {
    total++;
    contatos.textContent = 'Contatos cadastrados: ' + total;
});
```

### jQuery

```
var contatos = $('.contatos');
contatos.data('total', 0);
$('.botao-grava').click(function() {
    var total = contatos.data('total') + 1;
    contatos.data('total', total);
    contatos.text('Contatos cadastrados: ' + total);
});
```

## - Manipulação do DOM

- Limitações do *jQuery*

1) **Ausência de Padrões** – as mesmas funcionalidades com *jQuery* podem ser escritas de diversas maneiras.

### Forma 1

```
var numerosEl = $('numero');
$(numerosEl).on('keyup', function() {
    var resultado = 1;
    $.each(numerosEl, function() {
        resultado*=$(this).val();
    });
    $('resultado').text(resultado);
});
```

### Forma 2

```
var numerosEl = $('input[type=text]');
$(numerosEl).keyup(function() {
    var resultado = 1;
    $.each(numerosEl, function() {
        resultado*=$(this).val();
    });
    $('resultado').text(resultado);
});
```

## - Manipulação do DOM

- Limitações do *Jquery* (cont.)

**2) Preso à estrutura do DOM** – É necessário saber qual função chamar para obter o valor do documento.

```
// se fosse um input, seria val()  
$('.resultado').text(resultado);
```

Também é necessário saber se cada elemento possui classe no documento HTML

```
var numerosEl = $('.numero');
```

```
<p>  
  Quando é <input class="numero"> vezes <input class="numero">  
</p>  
<p>Resultado: <span class="resultado"></span></p>
```

## - Manipulação do DOM

- Limitações do *Jquery* (cont.)

- 3) **Testabilidade** – Como não existe separação entre a lógica e os elementos do *DOM*, testes automatizados se torna uma tarefa complexa, como por exemplo, realizar testes de unidade simulando a página.

```
// obtém um elemento do DOM  
var numerosEl = $('numero');
```

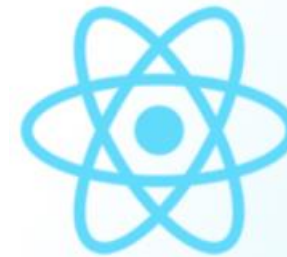
- 4) **Consistência entre Model e View** – toda vez que o *model* for atualizado, a apresentação na camada *view* também precisa ser atualizada.

```
// alterando o equivalente ao model  
$.each(numerosEl, function() {  
    resultado*=$(this).val();  
});
```

```
// sincronizando o model com a view  
$('.resultado').text(resultado);
```

## - Manipulação do DOM

- Outras opções – *Frameworks MVC client-side*



ReactJS



Vue.js

canjs



BACKBONE.JS



ANGULARJS  
by Google



ANGULAR



# Estudo de Caso: *AngularJS*

## - Definições

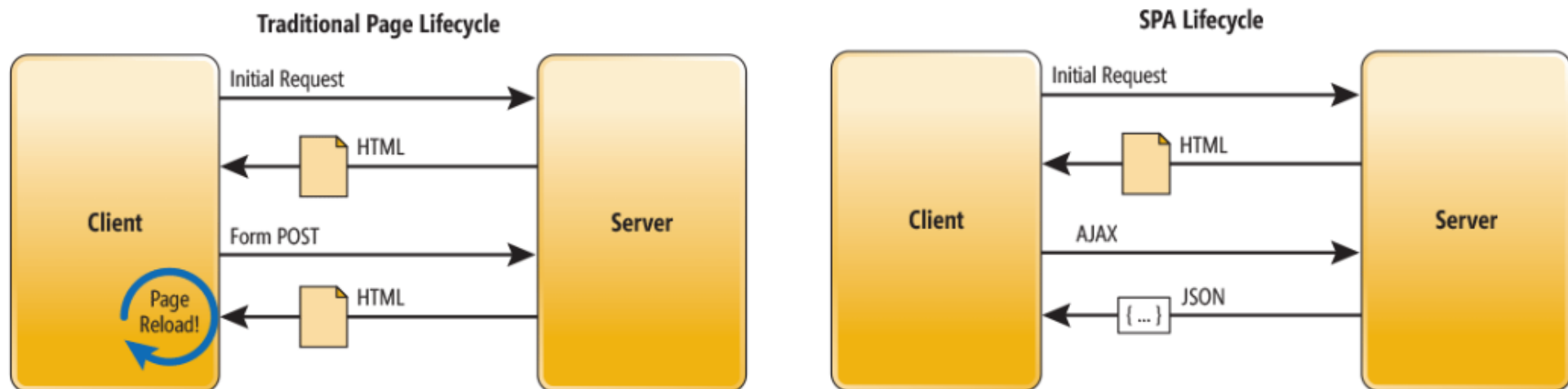
- *AngularJS* é um *framework JavaScript client-side*, criado pela *Google* e disponibilizado em 2009. Abstrai a manipulação do *DOM* de forma elegante, utilizando recursos como *two-way data binding*.



# Estudo de Caso: *AngularJS*

## - Características

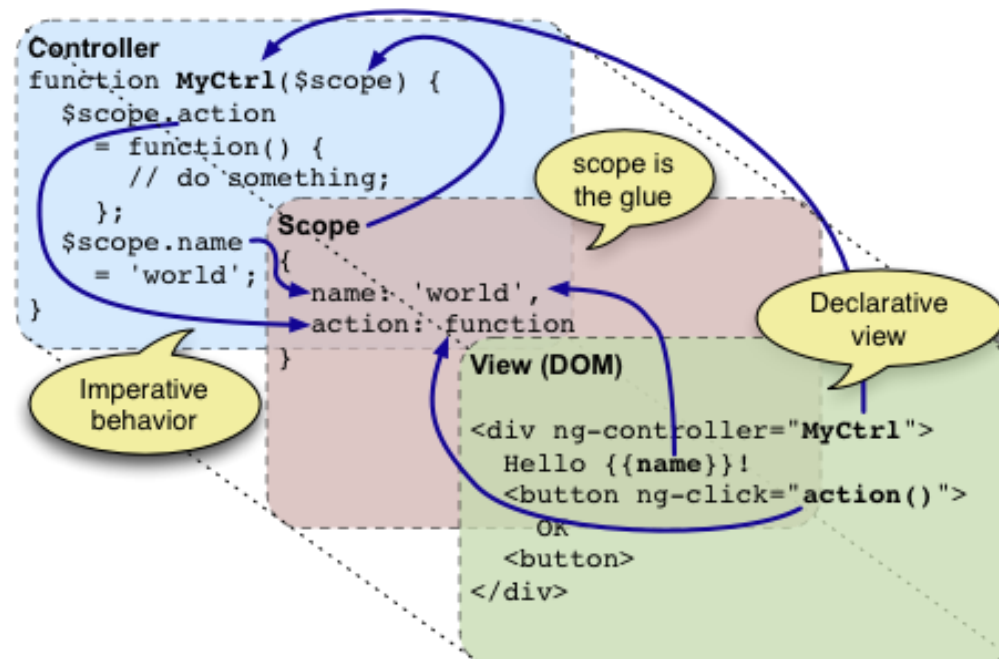
- *AngularJS* é **voltado para SPA** (*Single Page Web Applications*), que consiste basicamente em não recarregar a página durante o seu uso, ou seja, todo o *JavaScript* e *CSS* necessários já são carregados inicialmente. Com isso, escreve-se **menos código** *server-side*, **transferindo parte da lógica e dados** para o lado *client-side*.
- Neste contexto, o papel do *AngularJS* é **facilitar** o recebimento dos dados e **execução** da lógica de negócio diretamente no cliente.



# Estudo de Caso: *AngularJS*

## - Características

- *AngularJS* é um framework **MVC**. Com isso, toda a lógica é desacoplada dos elementos DOM, além de facilitar itens como manutenção, reusabilidade através de componentes e testabilidade.



# Estudo de Caso: *AngularJS*

## - Características

- *AngularJS* **força uma separação** entre o *front-end* e o *back-end*. Com isso, empresas podem ter equipes especializadas em cada parte da aplicação. Isso ajuda a eliminar situações onde a mesma equipe cria as duas camadas, especialmente quando se usa *frameworks* como *ASP.NET* e *JSF*.
- Em um projeto, as camadas **poderão evoluir em paralelo**.
- Esta separação pode inclusive envolver servidores, pois o *front-end* pode estar em um servidor razoavelmente escalável para tratar requisições *HTTP*, como o *Apache*.



# Estudo de Caso: *Angular*

---

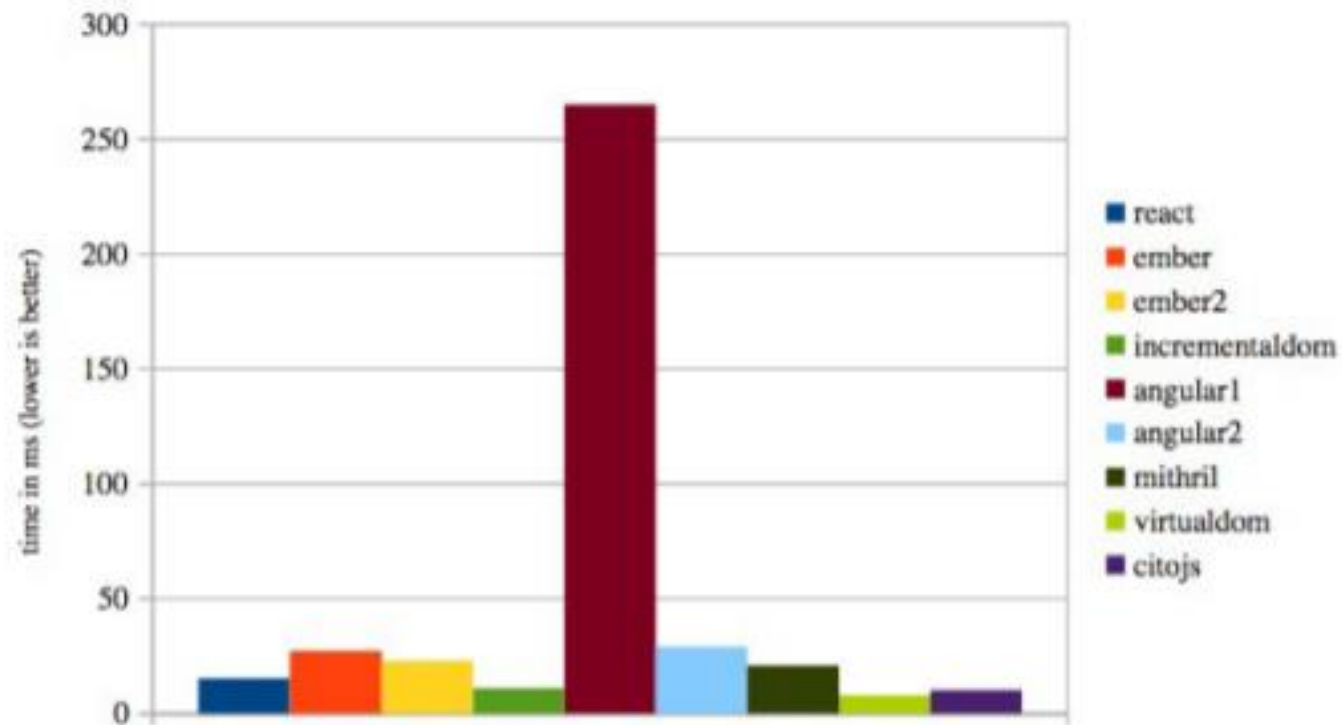
## - Definições

- Angular é um framework para desenvolvimento front-end, com HTML, CSS e Typescript, que no final é compilado para Javascript.
- Não é continuação do AngularJS. É um framework novo e remodelado, codificado do zero, com lições aprendidas do AngularJS.
- Aplicativos Mobile como Ionic possuem como base o Angular.



# Estudo de Caso: *Angular*

## - Tempo de resposta



# Estudo de Caso: *Angular*

---

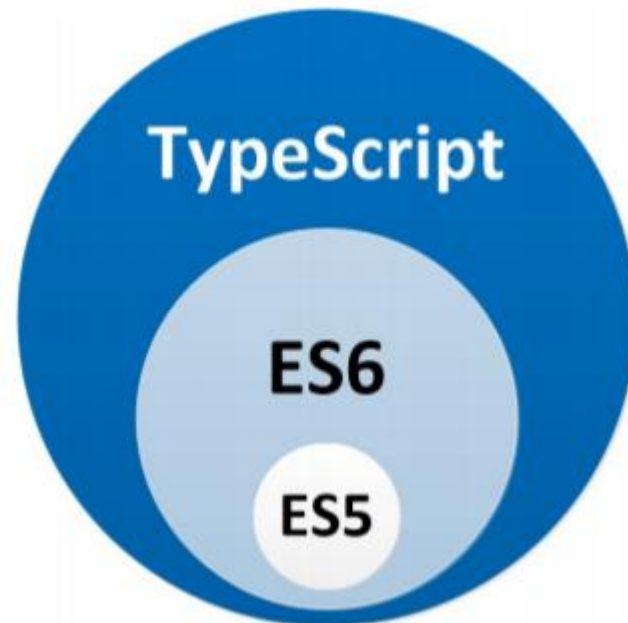
## - Definições

- O desenvolvimento em Angular é feito por meio de codificação TypeScript.
- Implementa funcionalidades do ES6
  - Tipagem de variáveis
  - Sintaxe clara e fácil de entender, parecendo-se com C# e Java.
- Javascript é somente um dialeto/apelido para a linguagem ECMAScript (ES)
- Em 2016 o ES chegou na versão 6
- Typescript permite escrever códigos utilizando estruturas fortemente tipadas e ter o código compilado para Javascript.

# Estudo de Caso: *Angular*

## - Definições

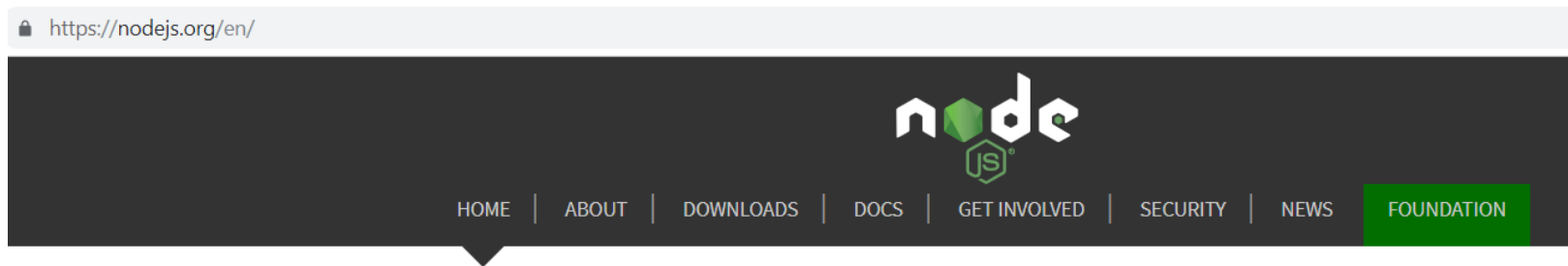
- TypeScript permite escrever o código na forma como se codifica no paradigma de Orientação a Objetos.





# Estudo de Caso: *Angular*

- Instalação
- NodeJs – o Angular2 roda em cima desta plataforma. Versão LTS



Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#).

Download for Windows (x64)

**10.15.1 LTS**

Recommended For Most Users

**11.10.0 Current**

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

# Estudo de Caso: *AngularJS*

## - Características

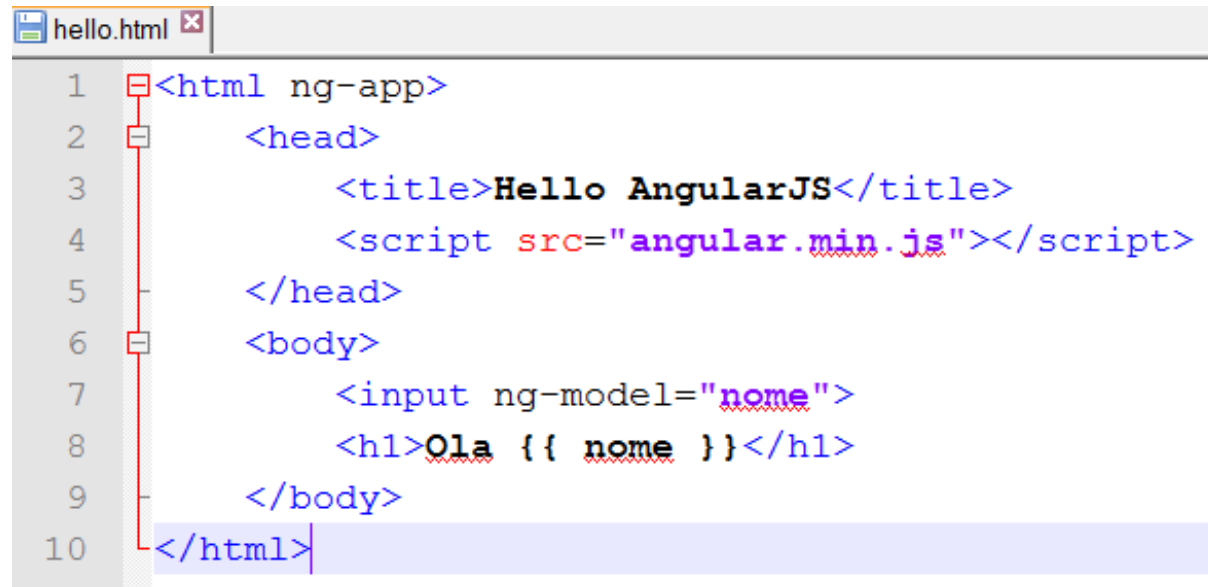
- *AngularJS* estende a sintaxe do *HTML*, **ampliando seu vocabulário** para expressar componentes usando diretivas. As diretivas atualizam partes da tela **sem qualquer intervenção manual**.
- Depois que o navegador transforma o texto da marcação na árvore DOM, o *AngularJS* percorre a estrutura DOM analisada. A cada diretiva encontrada, o *AngularJS* **executa sua lógica** para transformar diretivas em partes dinâmicas da tela.

```
<html ng-app="crudAtendimento">  
<div ng-controller="AtendimentoController">  
<button ng-click="salvar()">  
<input type="text" ng-model="atendimento.protocolo">  
<tr ng-repeat="atendimento in atendimentos | filter:criterio" ng-click="seleciona(atendimento)">
```

# Estudo de Caso: *AngularJS*

## - Hello World

- Deve-se incluir a biblioteca do *AngularJS* no documento
- Deve-se incluir a propriedade *ng-app* para “ativar” o *AngularJS*
- *Tags* personalizados com atributos para adicionar comportamento dinâmico no documento *HTML*
- Chaves duplas usadas como delimitador para expressões que exibem valores do modelo.



```
hello.html x
1 <html ng-app>
2   <head>
3     <title>Hello AngularJS</title>
4     <script src="angular.min.js"></script>
5   </head>
6   <body>
7     <input ng-model="nome">
8     <h1>Ola {{ nome }}</h1>
9   </body>
10 </html>
```

# Estudo de Caso: *AngularJS*

## - Utilizando AngularJS como MVW

- Possui estratégias envolvendo inicialização de dados e lógica, separando responsabilidades em camadas view e control.

```
helloMVW.html helloMVW.js
1 <html ng-app="helloMVWAPP">
2   <head>
3     <meta charset="utf-8">
4     <title>Hello MVW</title>
5     <script src="angular.min.js"></script>
6     <script src="helloMVW.js"></script>
7   </head>
8   <body>
9     <div ng-controller="helloMVWCtrl">
10       Como você está <input type="text" ng-model="nome"><br>
11       <h1>Eu estou muito bem, {{nome}}! e você?</h1>
12     </div>
13   </body>
14 </html>
```

```
helloMVW.html helloMVW.js
1 var app = angular.module('helloMVWAPP' , []);
2 app.controller('helloMVWCtrl', function($scope) {
3   $scope.nome = 'IFTM Campus UDI Centro' ;
4 });
```

# Estudo de Caso: *AngularJS*

## - \$Scope

- Um *\$Scope* é um objeto do *AngularJS* que expõe o modelo de domínio para a camada de visão. Tudo o que for atribuído para uma instância de escopo, sejam dados ou funcionalidades, poderão estar acessíveis para a camada de visão.
- Controla em qual parte do modelo as operações estão disponíveis para a camada de visão.

- Exe

```
var OlaCtrl = function ($scope) {  
    $scope.lerNome = function() {  
        return $scope.nome;  
    };  
};
```

```
<h1>Olá, {{ lerNome() }}!</h1>
```

# Estudo de Caso: *AngularJS*

## - Diretiva ng-repeat

- Permite iterar sobre uma coleção de objetos e criar novos elementos DOM para cada item em uma coleção.
- Exemplo:

```
var MundoCtrl = function ($scope) {  
    $scope.populacao = 700000000;  
    $scope.paises = [  
        {nome: 'Brasil', populacao: 63.1},  
        {nome: 'Alemanhã', populacao: 60.1}  
    ];  
};
```

```
<ul ng-controller="MundoCtrl">  
    <li ng-repeat="pais in paises">  
        {{pais.nome}} tem uma população de {{pais.populacao}}  
    </li>  
    <hr>  
    População do mundo: {{populacao}} Milhões de pessoas  
</ul>
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap

- Dentro do diretório *htdocs* do *Xampp*, criar um diretório *SistemaEcommerce*

Computador > Disco Local (C:) > softwares > xampp > htdocs >				
Nome	Data de modificaç...	Tipo	Tamanho	
dashboard	22/03/2016 14:29	Pasta de arquivos		
img	22/03/2016 14:29	Pasta de arquivos		
webalizer	22/03/2016 14:29	Pasta de arquivos		
xampp	22/03/2016 14:29	Pasta de arquivos		
applications	04/03/2016 06:51	Chrome HTML Do...	4 KB	
bitnami	21/07/2015 18:08	Documento de fol...	1 KB	
favicon	16/07/2015 12:32	Ícone	31 KB	
index.php	16/07/2015 12:32	Arquivo PHP	1 KB	
SistemaEcommerce	30/03/2016 13:19	Pasta de arquivos		

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- Criar a estrutura de diretórios, adicionando os arquivos abaixo (buscar em links – penúltimo slide)

Este Computador > Disco Local (C:) > xampp > htdocs > SistemaEcommerce

Nome	Data de modificaç...	Tipo	Tamanho
css	14/04/2018 10:23	Pasta de arquivos	
img	14/03/2019 16:46	Pasta de arquivos	
js	14/03/2019 17:13	Pasta de arquivos	
index.html	14/03/2019 17:13	Chrome HTML Do...	1 KB
menu.html	14/03/2019 17:13	Chrome HTML Do...	3 KB

e Computador > Disco Local (C:) > xampp > htdocs > SistemaEcommerce > css

Nome	Data de modificaç...	Tipo	Tamanho
bootstrap.css	13/02/2019 14:01	Documento de fol...	188 KB

Computador > Disco Local (C:) > xampp > htdocs > SistemaEcommerce > js

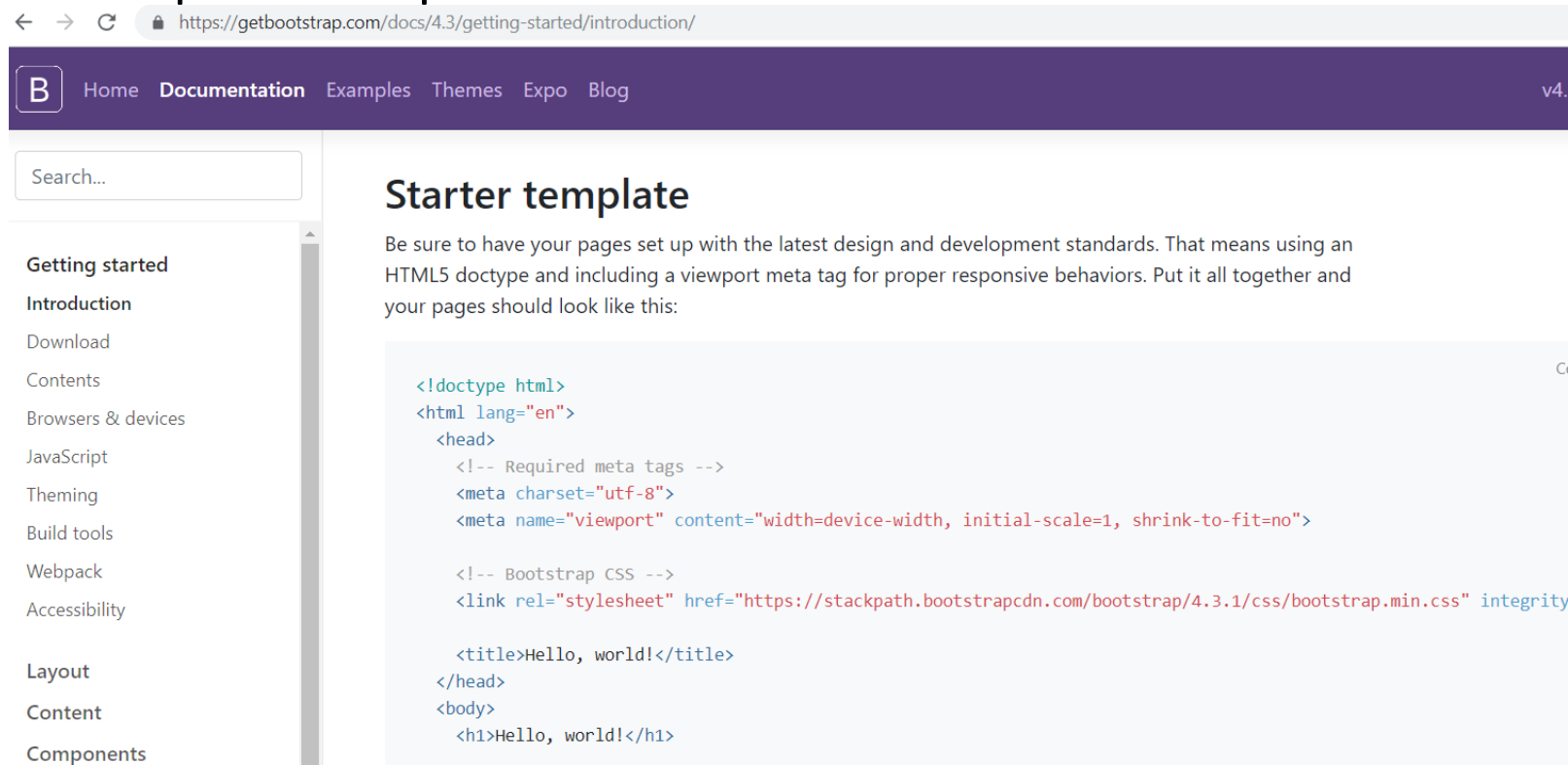
Nome	Data de modificaç...	Tipo	Tamanho
angular.js	13/03/2019 17:25	Arquivo JavaScript	1.340 KB
angular-resource.js	13/03/2019 17:25	Arquivo JavaScript	38 KB
bootstrap.js	09/04/2018 12:58	Arquivo JavaScript	120 KB
jquery-3.3.1.slim.min.js	14/03/2019 17:13	Arquivo JavaScript	69 KB
popper.min.js	14/03/2019 17:08	Arquivo JavaScript	21 KB



# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- No arquivo *index.html*, adicionar a estrutura que está no site do bootstrap, clicando em Get Started -> e pegando o código-fonte em Starter Template, modificando os links para os arquivos baixados



The screenshot shows the Bootstrap 4.3 documentation page. The browser address bar displays <https://getbootstrap.com/docs/4.3/getting-started/introduction/>. The navigation bar includes links for Home, Documentation, Examples, Themes, Expo, and Blog. A search bar is present on the left. The left sidebar lists various sections: Getting started, Introduction, Download, Contents, Browsers & devices, JavaScript, Theming, Build tools, Webpack, Accessibility, Layout, Content, and Components. The main content area is titled "Starter template" and contains the following HTML boilerplate code:

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyE0iFqWwA/VBzqiYpGhvFqL5NjuIXLGmhtPIgb7tt/L69IJvA">


    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- Efetuar o download de alguma imagem para adicionar na página inicial e colocar dentro do diretório *img*;

te Computador > Disco Local (C:) > softwares > xampp > apache > SistemaEcommerce > img

<input type="checkbox"/>	Nome	Data de modificaç...	Tipo	Tamanho
<input checked="" type="checkbox"/>	 logo	30/10/2015 01:07	Arquivo JPG	330 KB

- Adicionar na página index.html o suporte para AngularJS;
- Adicionar importação para o arquivo menu.html que será criado (próximo slide).
- Ao criar o arquivo menu.html, copie todo o conteúdo do index.html, retirando apenas a imagem e o include para o menu.

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

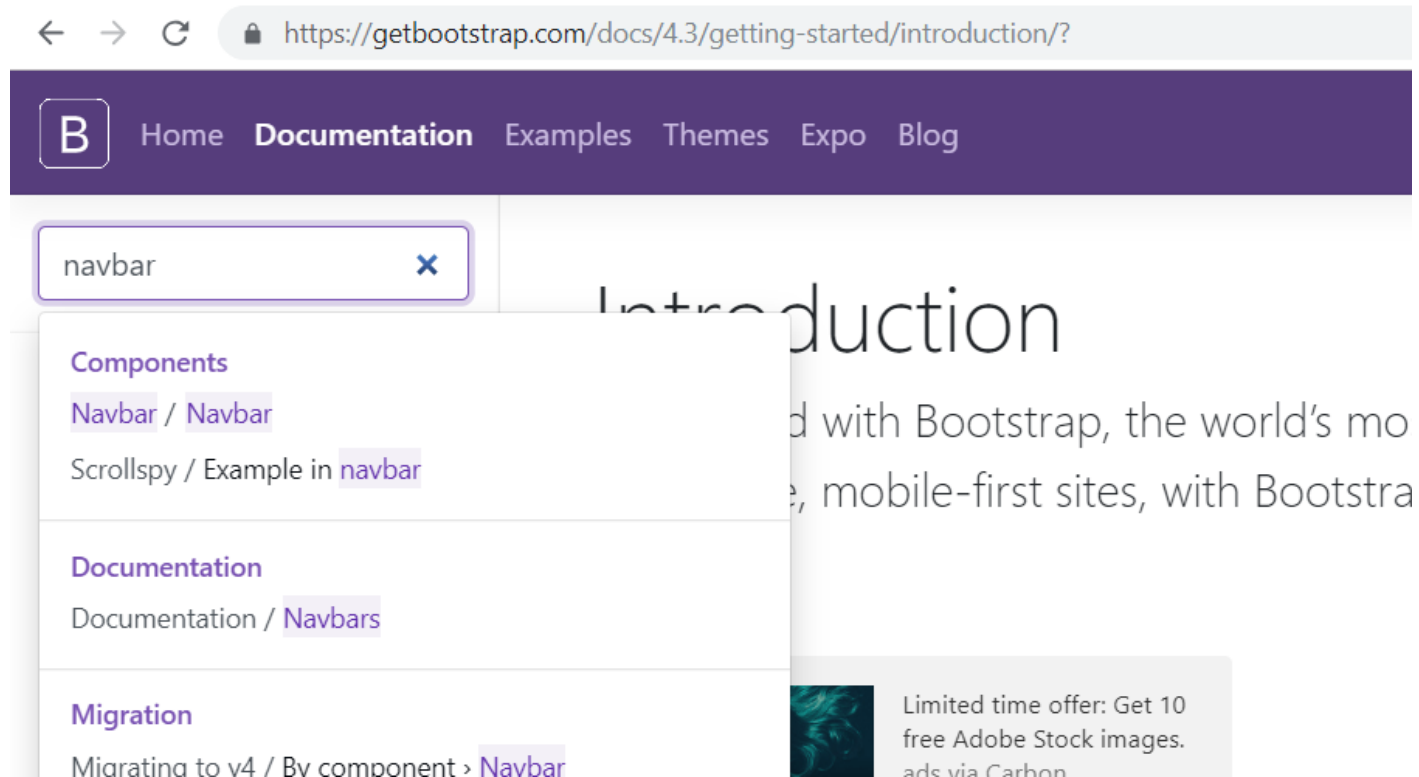
- Conteúdo do arquivo index.html

```
1 <html lang="en" ng-app>
2 <head>
3   <link rel="stylesheet" href="css/bootstrap.css">
4   <title>Sistema Ecommerce</title>
5 </head>
6 <body>
7   <script src="js/jquery-3.3.1.slim.min.js"></script>
8   <script src="js/popper.min.js"></script>
9   <script src="js/bootstrap.js"></script>
10  <script src="js/angular.js"></script>
11  <div ng-include src="'menu.html'"></div>
12  
13 </body>
14 </html>
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela **CRUD** usando **bootstrap**(cont.)

- No site do Bootstrap (<http://getbootstrap.com/>), acessar a aba *documentation*
- Nos links à direita, clicar em *navbar*



# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

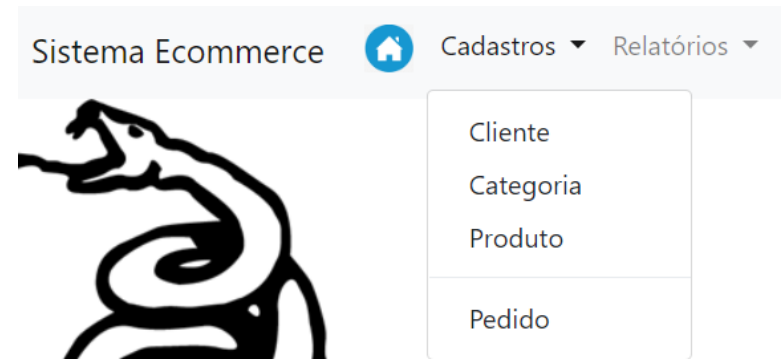
- Copie o código-fonte mostrado e cole na página *menu.html*.



The image shows a Bootstrap navbar with a light background. It contains links for 'Navbar', 'Home', 'Link', 'Dropdown' (with a dropdown arrow), and 'Disabled'. To the right is a search bar with the text 'Search' and a green 'Search' button. Below the navbar, the HTML code is displayed in a light blue box. A 'Copy' button is visible in the top right corner of the code block.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-cont
    <span class="navbar-toggler-icon"></span>
</button>
```

- Edite para ser mostrado conforme figura



The image shows the final result of editing the Bootstrap navbar. It is titled 'Sistema Ecommerce' and includes a home icon, a 'Cadastros' dropdown menu, and a 'Relatórios' dropdown menu. The 'Cadastros' dropdown menu is open, showing a list of items: 'Cliente', 'Categoria', 'Produto', and 'Pedido'. Below the navbar, there is a large black and white illustration of a snake.

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- Em *menu.html*, adicione um *link* para imagem

```
<li class="nav-item active">
  <a class="nav-link" href="index.html">
    
    <span class="sr-only">(current)</span></a>
</li>
```

- Nos links, adicione os caminhos abaixo

```
<li class="nav-item dropdown">
  <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button"
  data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Cadastros
  </a>
  <div class="dropdown-menu" aria-labelledby="navbarDropdown">
    <a class="dropdown-item" href="cliente.html">Cliente</a>
    <a class="dropdown-item" href="categoria.html">Categoria</a>
    <a class="dropdown-item" href="produto.html">Produto</a>
    <div class="dropdown-divider"></div>
    <a class="dropdown-item" href="pedido.html">Pedido</a>
  </div>
</li>
<li class="nav-item dropdown">
  <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button"
  data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Relatórios
  </a>
  <div class="dropdown-menu" aria-labelledby="navbarDropdown">
    <a class="dropdown-item" href="relatorioPedido.html">Pedido</a>
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- Crie o arquivo *cliente.html* (módulo *ng-app="clienteModule"*), com conteúdo quase idêntico ao *index.html*, retirando apenas a tag de imagem.
- No site do *Bootstrap-Documentation*, pesquise por Cards e cole o código-fonte abaixo

Bootstrap

Getting started

CSS

Components

JavaScript

Customize

Getting started

Layout

Content

Components

Alerts

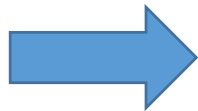
Badge

Breadcrumb

Buttons

Button group

Card



```
<div class="card border-secondary mb-3">  
  <div class="card-header">Cadastro de Clientes</div>  
</div>
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- Pesquisa por Tabs    Tabs

Takes the basic nav from above and adds the `.nav-tabs` class to generate a tabbed interface. Use them to

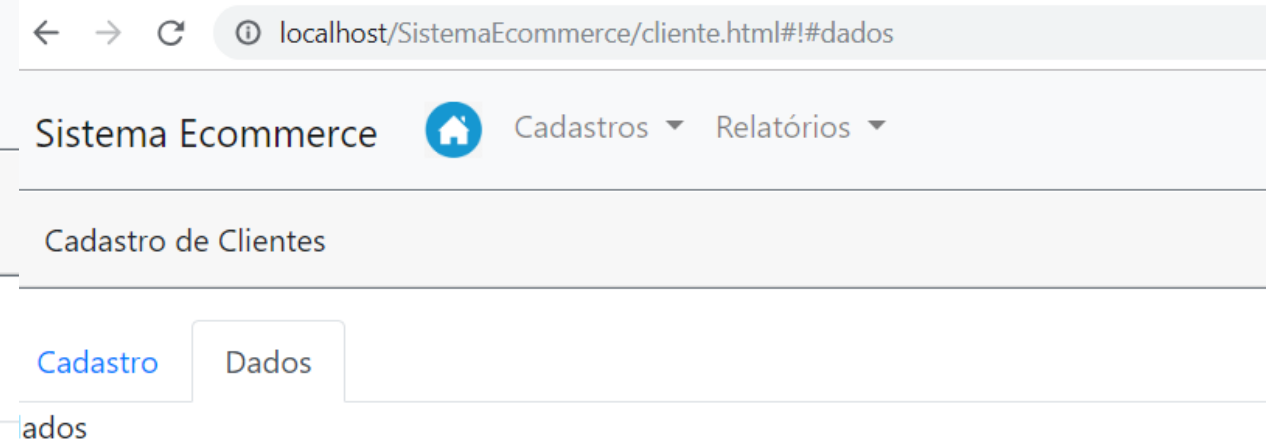
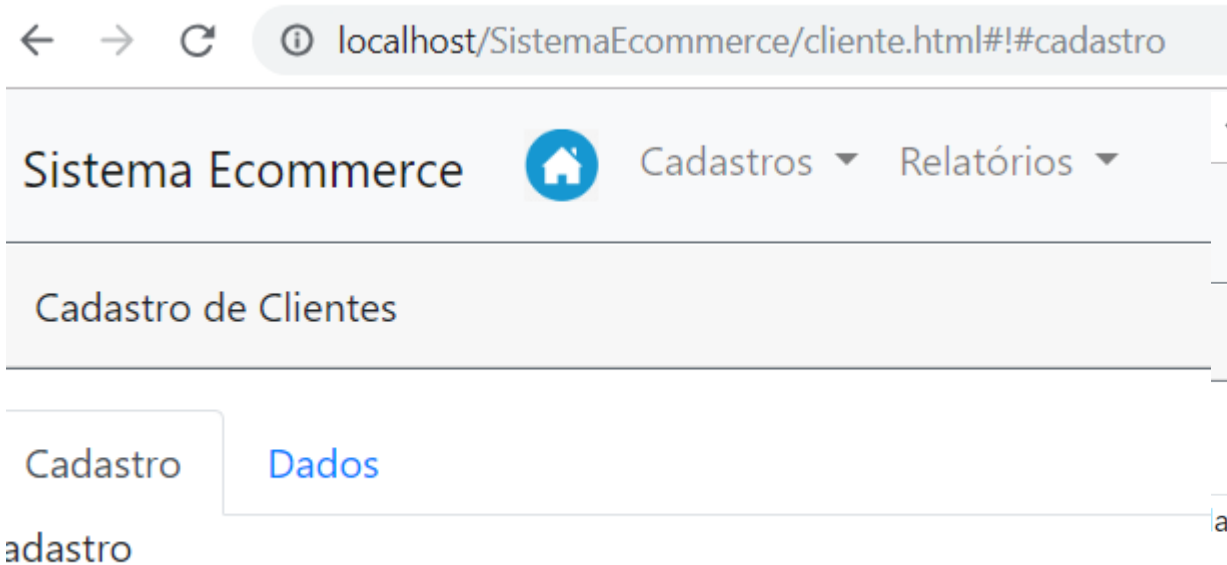
```
<ul class="nav nav-tabs" id="clienteTab" role="tablist">
  <li class="nav-item">
    <a class="nav-link active" id="cadastro-tab" data-toggle="tab" href=
      "#cadastro" role="tab" aria-controls="tabCadastro" aria-selected="true">
      Cadastro</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" id="dados-tab" data-toggle="tab" href="#dados" role=
      "tab" aria-controls="tabDados" aria-selected="false">Dados</a>
  </li>
</ul>
<div class="tab-content" id="clienteTabContent">
  <div class="tab-pane fade show active" id="cadastro" role="tabpanel"
    aria-labelledby="cadastro-tab">cadastro</div>
  <div class="tab-pane fade" id="dados" role="tabDados" aria-labelledby=
    "dados-tab">dados</div>
</div>
```



# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- Resultado:



# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- No ícone superior do site do *Bootstrap*, clique em *Components*

Bootstrap   Getting started   CSS   Components   JavaScript   Customize

- Selecione *Button Groups* *Tab*, e cole o código na tela de clientes.

### Button groups

Basic example

Button toolbar

Sizing

Nesting

Vertical variation

Justified button groups

```
<!-- Tab panes -->
<div role="tabpanel" class="tab-pane active" id="cadastro">
<br>
    <button type="button" class="btn btn-info">Novo</button>
    <button type="button" class="btn btn-success">Salvar</button>
    <button type="button" class="btn btn-danger">Excluir</button>
    <button type="button" class="btn btn-warning">Pesquisar</button>
</div>
<div role="tabpanel" class="tab-pane" id="dados">dados</div>
</div>
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- No ícone superior do site do *Bootstrap*, clique em *Css*

Bootstrap   Getting started   CSS   Components   JavaScript   Customize

- Selecione *Forms*, e cole o código na tela de clientes.

### Forms

- Basic example
- Inline form
- Horizontal form
- Supported controls
- Static control
- Focus state
- Disabled state
- Readonly state
- Help text
- Validation states
- Control sizing

```
<button type="button" ng-click="pesquisar();" class="btn btn-warning">Pesquisar</button>
<form>
  <hr/>
  <div class="form-group col-md-2">
    <label for="inputCodigo">Código:</label>
    <input type="text" class="form-control" id="inputCodigo" placeholder="Código">
  </div>
  <div class="form-group col-md-6">
    <label for="inputNome">Nome:</label>
    <input type="text" class="form-control" id="inputNome" placeholder="Nome">
  </div>
  <div class="form-group col-md-4">
    <label for="inputCargo">Cargo</label>
    <input type="text" class="form-control" id="inputCargo" placeholder="Cargo">
  </div>
  <div class="form-group col-md-6">
    <label for="inputEndereco">Endereço</label>
    <input type="text" class="form-control" id="inputEndereco" placeholder="Endereço">
  </div>
</form>
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

```
<div class="form-group col-md-4">
  <label for="inputCidade">Cidade</label>
  <input type="text" class="form-control" id="inputCidade" placeholder="Cidade">
</div>
<div class="form-group col-md-2">
  <label for="inputCEP">CEP</label>
  <input type="text" class="form-control" id="inputCEP" placeholder="CEP">
</div>
<div class="form-group col-md-4">
  <label for="inputPaís">País</label>
  <input type="text" class="form-control" id="inputPaís" placeholder="País">
</div>
<div class="form-group col-md-3">
  <label for="inputTelefone">Telefone</label>
  <input type="text" class="form-control" id="inputTelefone" placeholder="Telefone">
</div>
<div class="form-group col-md-3">
  <label for="inputFax">Fax</label>
  <input type="text" class="form-control" id="inputFax" placeholder="Fax">
</div>
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- No ícone superior do site do *Bootstrap*, clique em *Css*

Bootstrap   Getting started   CSS   Components   JavaScript   Customize

- Selecione *Tables - Hover rows*, e cole o código na tela de clientes.

```
<div role="tabpanel" class="tab-pane" id="dados">
<input type="text" class="form-control" ng-model="criterio"
placeholder="O que você está procurando?" /><br />
<table class="table table-hover">
  <th>Código</th>
  <th>Nome</th>
  <th>Cargo</th>
  <th>Endereço</th>
  <th>Cidade</th>
  <th>CEP</th>
  <th>País</th>
  <th>Telefone</th>
  <th>Fax</th>
```

### Tables

Basic example  
Striped rows  
Bordered table  
Hover rows  
Condensed table  
Contextual classes  
Responsive tables

## - Criação de uma tela CRUD usando bootstrap(cont.)

Sistema Ecommerce

file:///C:/xampp/htdocs/SistemaEcommerce/cliente.html

Pesquisar

Sistema Ecommerce Cadastros ▾

Cadastro de Clientes

Cadastro Dados

Novo Salvar Excluir Pesquisar

**Código:**  **Nome:**  **Cargo**

**Endereço**  **Cidade**  **CEP**

**País**  **Telefone**  **Fax**

# Estudo de Caso: *AngularJS*

- Para que o DOM seja modificado facilmente com esta tela CRUD, será necessário adicionar diretivas AngularJS no html

- 1) Diretiva ng-app

```
<html lang="en" ng-app="clienteModule">
```

- 2) Diretiva ng-controller

```
<body ng-controller="clienteControl">
```

- 3) Diretiva ng-click

```
<button type="button" ng-click="novo()" class="btn btn-info">Novo</button>  
<button type="button" ng-click="salvar()" class="btn btn-success">Salvar</button>  
<button type="button" ng-click="excluir()" class="btn btn-danger">Excluir</button>  
<button type="button" ng-click="pesquisar()" class="btn btn-warning">Pesquisar</button>
```

# Estudo de Caso: *AngularJS*

- Para que o DOM seja modificado facilmente com esta tela CRUD, será necessário adicionar diretivas AngularJS no html

- 4) Diretiva ng-model

```
<div class="form-group col-md-2">
  <label for="inputCodigo">Código:</label>
  <input type="text" class="form-control" ng-model="cliente.codigo" id="inputCodigo" pla
</div>
<div class="form-group col-md-6">
  <label for="inputNome">Nome:</label>
  <input type="text" class="form-control" ng-model="cliente.nome" id="inputNome" placehc
</div>
<div class="form-group col-md-4">
  <label for="inputCargo">Cargo</label>
  <input type="text" class="form-control" ng-model="cliente.cargo" id="inputCargo" place
</div>
<div class="form-group col-md-6">
  <label for="inputEndereco">Endereço</label>
  <input type="text" class="form-control" ng-model="cliente.endereco" id="inputEndereco"
</div>
<div class="form-group col-md-4">
  <label for="inputCidade">Cidade</label>
  <input type="text" class="form-control" ng-model="cliente.cidade" id="inputCidade" pla
</div>
```



# Estudo de Caso: *AngularJS*

- Para que o DOM seja modificado facilmente com esta tela CRUD, será necessário adicionar diretivas AngularJS no html
  - 4) Diretiva ng-model (cont.)

```
<div class="form-group col-md-2">
  <label for="inputCEP">CEP</label>
  <input type="text" class="form-control" ng-model="cliente.cep" id="inputCEP" placeholder="CEP" />
</div>
<div class="form-group col-md-4">
  <label for="inputPais">País</label>
  <input type="text" class="form-control" ng-model="cliente.pais" id="inputPais" placeholder="País" />
</div>
<div class="form-group col-md-3">
  <label for="inputTelefone">Telefone</label>
  <input type="text" class="form-control" ng-model="cliente.telefone" id="inputTelefone" placeholder="Telefone" />
</div>
<div class="form-group col-md-3">
  <label for="inputFax">Fax</label>
  <input type="text" class="form-control" ng-model="cliente.fax" id="inputFax" placeholder="Fax" />
</div>
```

# Estudo de Caso: *AngularJS*

- Para que o DOM seja modificado facilmente com esta tela CRUD, será necessário adicionar diretivas AngularJS no html
  - 5) Diretiva ng-repeat

```
<tr ng-repeat="clienteTabela in clientes | filter:criterio"
    ng-click="seleciona(clienteTabela)">
  <td>{{clienteTabela.codigo}}</td>
  <td>{{clienteTabela.nome}}</td>
  <td>{{clienteTabela.cargo}}</td>
  <td>{{clienteTabela.endereco}}</td>
  <td>{{clienteTabela.cidade}}</td>
  <td>{{clienteTabela.cep}}</td>
  <td>{{clienteTabela.pais}}</td>
  <td>{{clienteTabela.telefone}}</td>
  <td>{{clienteTabela.fax}}</td>
</tr>
```

# Estudo de Caso: *AngularJS*

- Para que o DOM seja modificado facilmente com esta tela CRUD, será necessário adicionar diretivas AngularJS no html
  - 6) Diretiva ng-show

```
</table>
```

```
<span ng-show="(fornecedores | filter:criterio).length == 0">Infelizmente  
não temos o item que você está procurando!</span>
```

```
</div>
```

# Estudo de Caso: *AngularJS*

- Para construir os métodos e objetos demandados pelo view, será criado o arquivo clienteControl.js

1) Crie a pasta control e o arquivo clienteControl.js dentro da mesma

Computador > Disco Local (C:) > xampp > htdocs > SistemaEcommerce

Nome	Data de modificaç...	Tipo	Tamanho
control	30/03/2016 22:23	Pasta de arquivos	
css	30/03/2016 19:32	Pasta de arquivos	
img	30/03/2016 19:51	Pasta de arquivos	
js	30/03/2016 19:33	Pasta de arquivos	
cliente	01/04/2016 01:45	Chrome HTML Do...	5 KB
index	30/03/2016 20:26	Chrome HTML Do...	1 KB
menu	30/03/2016 20:02	Chrome HTML Do...	3 KB

Computador > Disco Local (C:) > xampp > htdocs > SistemaEcommerce > control

Nome	Data de modificaç...	Tipo
clienteControl	01/04/2016 01:50	Arquivo JavaScript

# Estudo de Caso: *AngularJS*

- Para construir os métodos e objetos demandados pelo view, será criado o arquivo clienteControl.js
- 2) Adicione uma referência no arquivo cliente.html para o control

```
<!-- The above 3 meta tags *must* come first in the h
<title>Sistema Ecommerce</title>

<!-- Bootstrap -->
<link href="css/bootstrap.min.css" rel="stylesheet">

<script src="js/angular.min.js"></script>
<script src="control/clienteControl.js"></script>

</head>
```

# Estudo de Caso: *AngularJS*

- Para construir os métodos e objetos demandados pelo view, será criado o arquivo clienteControl.js

## 3) Crie o conteúdo abaixo:

```
1 var app = angular.module('clienteModule',[]);
2 app.controller('clienteControl',function($scope) {
3
4     $scope.clientes = [{ 'codigo':'1', 'nome':'Carlos', 'cargo':'Professor', 'endereco':'Ria xxx', 'cidade':
5     'Udia', 'cep':'38400000', 'pais':'Brasil', 'telefone':'98849492', 'fax':'23232323'},
6     { 'codigo':'2', 'nome':'Marcos', 'cargo':'Analista', 'endereco':'Ria xxx', 'cidade':
7     'Udia', 'cep':'38400000', 'pais':'Brasil', 'telefone':'98849492', 'fax':'23232323'}
8 ];
9
10 $scope.salvar = function() {
11     $scope.clientes.push($scope.cliente);
12     $scope.novo();
13 }
14
15 $scope.pesquisar = function() {
16 }
```

# Estudo de Caso: *AngularJS*

- Para construir os métodos e objetos demandados pelo view, será criado o arquivo clienteControl.js

## 3) Crie o conteúdo abaixo (cont):

```
20 $scope.excluir = function() {  
21     $scope.clientes.splice($scope.clientes.indexOf($scope.cliente), 1);  
22     $scope.novo();  
23 }  
24  
25 $scope.novo = function () {  
26     $scope.cliente = "";  
27 };  
28  
29 $scope.seleciona = function (cliente) {  
30     $scope.cliente = cliente;  
31 };  
32  
33 });
```

# Estudo de Caso: *AngularJS*

- Com o conteúdo criado, é possível testar a camada *front-end*, sem necessidade do *back-end* estar pronto

Cadastro de Clientes

Cadastro

Dados

Código	Nome	Cargo	Endereço	Cidade	CEP	País
1	Carlos	Professor	Ria xxx	Udia	38400000	Brasil
2	Marcos	Analista	Ria xxx	Udia	38400000	Brasil
3	asd	asd1	asd2	asd3eee	asd	asd



# Estudo de Caso: *AngularJS*

## - Observações:

- Foi criada uma lista estática de clientes em formato JSON para testar a tabela e os campos do CRUD;
- Todos os métodos do control estão simplesmente manipulando uma lista estática em memória. A grande diferença é que **esta abordagem é transparente ao HTML**, logo quando for modificado para requisições REST, não haverá mudança no código html;
- Com a propriedade *two-way-data-binding*, **qualquer objeto que seja alterado simboliza mudança automática no DOM**. Com isso, ao selecionar um elemento de uma tabela, assim que o objeto `$scope.cliente` recebe o objeto da tabela, os campos do tipo *input type* automaticamente serão atualizados com o valor do objeto. Da mesma forma que, modificando o valor nos campos *input type*, o objeto irá ter seu valor alterado.
- O método *pesquisar()* ainda não possui conteúdo porque o objeto `$scope.criterio` está sendo usado para filtrar na lista de clientes. Com as requisições REST, a intenção é que este método pesquise no servidor.

# Estudo de Caso: *AngularJS*

---

## - Recurso \$http

- Serviço que permite a comunicação com servidores HTTP remotos via objetos *XMLHttpRequest* ou *JsonP*
- Existem outras opções com um nível mais alto de abstração, como o serviço *\$resource*
- Métodos
  - `$http.get`
  - `$http.head`
  - `$http.post`
  - `$http.put`
  - `$http.delete`
  - `$http.jsonp`
  - `$http.patch`
- Maiores informações sobre este recurso: [https://docs.angularjs.org/api/ng/service/\\$http](https://docs.angularjs.org/api/ng/service/$http)

# Estudo de Caso: *AngularJS*

## - Recurso \$http

- Para utilizar o recurso no control, basta adicionar uma referência à variável \$http

```
app.controller('clienteControl', function($scope, $http) {
```

- Será implementado o conteúdo abaixo:

```
var url = 'http://localhost:8080/SistemaEcommerce/rs/cliente';
```

```
$scope.pesquisar = function(){  
    $http.get(url).success(function (clientesRetorno) {  
        $scope.clientes = clientesRetorno;  
    }).error(function(mensagemErro) {  
        alert(mensagemErro);  
    });  
}
```

```
$scope.pesquisar();
```

# Estudo de Caso: *AngularJS*

---

## - Recurso \$http

- A variável url foi criada porque os métodos get, post e put compartilharão da mesma url;
- O método \$http.get é executado, e em caso de sucesso, é retornada uma lista de clientes do servidor em formato JSON. A lista que antes era inicializada estaticamente, receberá essa lista. Os códigos de sucesso são de 200 a 299;
- Em caso de erro, um alert é impresso na tela
- Após a definição do método, este será executado ao carregar o script.

# Estudo de Caso: *AngularJS*

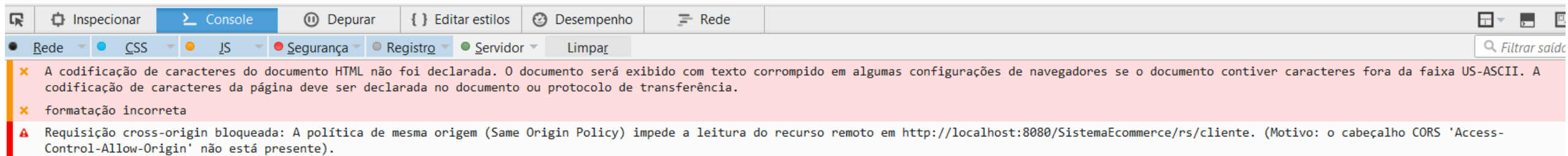
---

## - Recurso \$http

- A variável url foi criada porque os métodos get, post e put compartilharão da mesma url;
- O método \$http.get é executado, e em caso de sucesso, é retornada uma lista de clientes do servidor em formato JSON. A lista que antes era inicializada estaticamente, receberá essa lista. Os códigos de sucesso são de 200 a 299;
- Em caso de erro, um alert é impresso na tela
- Após a definição do método, este será executado ao carregar o script.

# Estudo de Caso: *AngularJS*

- Recurso \$http
  - Ao executar, é mostrado o seguinte erro:



- O problema ocorre porque a requisição está buscando um recurso de um domínio diferente. Por padrão, uma aplicação web só pode fazer requisições para seu próprio domínio.
- Para resolver tal questão, será criada uma classe no servidor que implementa o mecanismo CORS

# Estudo de Caso: *AngularJS*

## - Recurso \$http

```
CORSFilter.java
1 package model.facade.rs;
2
3 import java.io.IOException;
4
5
6
7
8
9
10 @Provider
11 public class CORSFilter implements ContainerResponseFilter {
12
13     @Override
14     public void filter(final ContainerRequestContext requestContext,
15                       final ContainerResponseContext cres) throws IOException {
16         cres.getHeaders().add("Access-Control-Allow-Origin", "*");
17         cres.getHeaders().add("Access-Control-Allow-Headers", "origin, content-type, accept, authorization");
18         cres.getHeaders().add("Access-Control-Allow-Credentials", "true");
19         cres.getHeaders().add("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE, OPTIONS, HEAD");
20         cres.getHeaders().add("Access-Control-Max-Age", "1209600");
21     }
22
23 }
24
```

# Estudo de Caso: *AngularJS*

## - Recurso \$http

- A anotação `@Provider` faz com que a engine do JAX-RS detecte automaticamente a existência dessa classe `CORSFilter`.
- A classe implementa a interface `ContainerResponseFilter`, para que, toda vez que chegar uma requisição REST, o filtro seja executado, adicionando atributos que permitam domínios distintos.

Sistema Ecommerce Cadastro ▾

Cadastro de cliente

[Cadastro](#) [Dados](#)

O que voce esta procurando?

Codigo	Nome	Cargo	Endereco	Cidade	CEP	Pais	Telefone	Fax
1	Maria Anders	Sales Representative	Obere Str. 57	Berlin	12209	Germany	030-0074321	030-0076545
2	Ana Trujillo	Owner	Avda. de la ConstituciÃ³n 2222	MÃ©xico D.F.	05021	Mexico	(5) 555-4729	(5) 555-3745
3	Antonio Moreno	Owner	Mataderos 2312	MÃ©xico D.F.	05023	Mexico	(5) 555-3932	
4	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	WA1 1DP	UK	(171) 555-7788	(171) 555-6750
5	Christina Berglund	Order Administrator	BerguvsvÃ¥gen 8	LuleÃ¥	S-958 22	Sweden	0921-12 34 65	0921-12 34 67



# Estudo de Caso: *AngularJS*

## - Recurso \$http

```
21  $scope.salvar = function() {  
22      if ($scope.cliente.codigo == '') {  
23          $http.post(url,$scope.cliente).success(function(cliente) {  
24              $scope.clientes.push($scope.cliente);  
25              $scope.novo();  
26          }).error(function (erro) {  
27              alert(erro);  
28          });  
29      } else {  
30          $http.put(url,$scope.cliente).success(function(cliente) {  
31              $scope.pesquisar();  
32              $scope.novo();  
33          }).error(function (erro) {  
34              alert(erro);  
35          });  
36      }  
37  }
```

# Estudo de Caso: *AngularJS*

## - Recurso \$http

```
39  $scope.excluir = function() {  
40      if ($scope.cliente.codigo == '') {  
41          alert('Selecione um cliente');  
42      } else {  
43          urlExcluir = url+'/'+$scope.cliente.codigo;  
44          $http.delete(urlExcluir).success(function () {  
45              $scope.pesquisar();  
46              $scope.novo();  
47          }).error(function (erro) {  
48              alert(erro);  
49          });  
50      }  
51  }
```

# Estudo de Caso: *AngularJS*

- Além do Apache, o projeto também pode ser executado diretamente no Wildfly.



# EXERCÍCIO VALENDO 10 PONTOS

---

- Implementar a camada front-end consumindo todos os serviços criados no trabalho do back-end do capítulo 2:
  - Uso de angularjs e bootstrap
  - Telas de menu
  - Requisições REST
- Data de entrega: 02/05/2016
- O projeto precisa estar comitado no repositório informado por email no trabalho anterior.

# LINKS DOS SOFTWARES UTILIZADOS

---

- **Jquery** - <https://code.jquery.com/jquery-3.3.1.min.js>
- **Bootstrap** - <https://github.com/twbs/bootstrap/archive/v4.3.1.zip>
- **AngularJS** - <https://code.angularjs.org/1.7.7/angular-1.7.7.zip>
- **PopperJS** - <https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js>

# REFERÊNCIAS

- GRIGORIK, I. High Performance Browser Networking. O'Reilly, 2013. Disponível em <http://chimera.labs.oreilly.com/books/12300000000545>. Acesso em 05/08/2015.
- MICROSOFT. Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET. Disponível em <https://msdn.microsoft.com/en-us/magazine/dn463786.aspx>. Acesso em 05/08/2015.
- Front End Brasil. A história do HTML. Disponível em <http://www.frontendbrasil.com.br/artigos/a-historia-do-html/>. Acesso em 05/08/2015.
- Modern Web. 8 bootstrap alternatives. Disponível em <http://modernweb.com/2014/02/17/8-bootstrap-alternatives/>. Acesso em 05/08/2015.
- TutorialZine. 50 plugin for extending Twitter Bootstrap. Disponível em <http://tutorialzine.com/2013/07/50-must-have-plugins-for-extending-twitter-bootstrap/>. Acesso em 07/08/2015.