

itu ! =

Beijinhos !!! ~~Q.~~
e abraços MJ.

jouerix

jQuery biblioteca JavaScript multi-plataforma

→ Simplificar a programação p/o cliente HTML

| Sintaxe → tornar + fácil a manipulação pelos elementos de um documento.

ex.: selecionar elementos DOM

* criar animações

* manipular eventos

* desenvolver aplicações AJAX.

Vantagens:

→ Separação entre o JavaScript e o HTML

Em Javascript: usa atributos HTML para identificar as funções pra manipulação de eventos

```
<form id="target" action="http://192.168.160.36/FormEcho.aspx">  
  <input type="submit" value="Enviar" class="btn btn-default" onclick="return validateForm()" />  
</form>
```

```
function validateForm() {  
  // TODO ...  
}
```

Em jQuery lida com eventos apenas em JavaScript

```
<form id="target" action="http://192.168.160.36/FormEcho.aspx">  
  <input type="submit" value="Enviar" class="btn btn-default" />  
</form>
```

```
$("#target").submit(function (event) {  
  // TODO ...  
});
```

→ Elimina incompatibilidades entre navegadores:

os motores JavaScript dos diferentes navegadores diferem

≠

o jQuery lida e/ todos as inconsistências entre browsers fornecendo uma interface consistente que funciona nos diferentes navegadores

→ Extensível

Através da adição de novas bibliotecas ao projeto.

(eventos, elementos e métodos)

Desvantagens

- Biblioteca grande ($\sim 88K$) para importar e muitas xs aplicações utilizamos uma pequena parte das funcionalidades disponibilizadas
- Esconde as partes complexas do JavaScript, dificultando a aprendizagem do mesmo.
- O JavaScript puro é mais rápido a aceder ao DOM.

A biblioteca jQuery é um único ficheiro JavaScript, contendo todas as suas funcionalidades: acesso aos elementos DOM, eventos, efeitos e funções comuns do Ajax.

Ficheiro local:

```
<script src="Scripts/jquery-3.6.0.min.js"></script>
```

{ jquery-3.6.0.js (282kb)
jquery-3.6.0.min.js (88kb)

Ficheiro remoto (CDN):

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
```

Lista de bibliotecas suportadas pela google:
<https://developers.google.com/speed/libraries/>

SINTAXE



("selector").action()

"consultar/encontrar"
elementos HTML no
documento

id, classes, tipos, atributos,
valores de atributos ...

ação a ser
executada no(s)
elemento(s)

definir/adicionar
à biblioteca jQuery

Seletores

elementos

Exemplo:

```
<p>Este é um parágrafo.</p>
<button type="button"><i class="fa fa-edit"></i></button>
```

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("p").hide();
    });
});
</script>
```

id

```
<div id="errorMessage">O nome deve ter pelo menos três letras.</div>
<button type="button"><i class="fa fa-edit"></i></button>
```

```
<script>
$(document).ready(function () {
    $("button").click(function () {
        $("#errorMessage").show();
    });
});
</script>
```

class

altera propriedades
ess

```
<div class="bannerTop"></div>
```

```
<script>
$(document).ready(function () {
    $(".bannerTop").css({"background-color": "yellow", "font-size": "200%"});
});</script>
```

option : selected

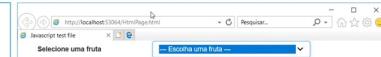
de seleção

```
<label for="fruta" class="col-md-4 form-control-static">Selecione uma fruta</label>
<div class="col-md-6">
    <select id="fruta" class="form-control">
        <option>--- Escolha uma fruta ---</option>
        <option value="1">Banana</option>
        <option value="2">Maçã</option>
        <option value="3">Pera</option>
    </select>
</div>
```

```
$(document).ready(function () {
    $("#fruta").change(function () {
        var retVal = $("#fruta option:selected").val() + " - " + $("#fruta option:selected").text();
        alert(retVal)
    });
});
```

ativado sempre que é mudada
a seleção

valor
selecionado



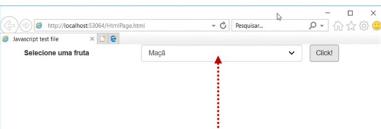
ou html()
texto da
opção

GET
seletores usados p/
verificar os valores
dos elementos HTML

SET
atuar/alterar
os elementos

Exemplo:

```
<label for="fruta" class="col-sm-4 form-control-static">Selecione uma fruta</label>
<div class="col-sm-6">
    <select id="fruta" class="form-control">
        <option>--- Escolha uma fruta ---</option>
        <option value="1">Banana</option>
        <option value="2">Maçã</option>
        <option value="3">Pera</option>
    </select>
</div>
<div class="col-sm-2">
    <button type="button" id="myButton" class="btn btn-default">Click!</button>
</div>
```



```
$(document).ready(function () {
    $("#myButton").click(function () {
        $("#fruta").val(2);
    });
});
```

12

Sintaxe	Descrição
<code>\$("")</code>	Seleciona todos os elementos do documento
<code>\$(this)</code>	Seleciona o elemento html que está a ser manipulado. Nota: this sem aspas("")!!!
<code>\$("p.intro")</code>	Seleciona todos os elementos <p> com class="intro"
<code>\$("p:first")</code>	Seleciona o primeiro elemento <p> do documento
<code>\$("ul li:first")</code>	Seleciona o primeiro element do primeiro elemento
<code>\$("ul li:first-child")</code>	Seleciona o primeiro elemento de todos os elementos
<code>\$("[href]")</code>	Seleciona todos os elementos que possuam o atributo "href"
<code>\$("a[target='_blank']")</code>	Seleciona todos os elementos <a> com o atributo target igual a "_blank"
<code>\$("a[target!='_blank']")</code>	Seleciona todos os elementos <a> com o atributo target diferente de "_blank"
<code>\$(":button")</code>	Seleciona todos os elementos <button> e/ou <input> com type="button"
<code>\$("tr:even")</code>	Seleciona todas as linhas pares dos elementos <tr>
<code>\$("tr:odd")</code>	Seleciona todas as linhas ímpares dos elementos <tr>

1 1

Todos os métodos de jQuery estão sempre dentro
do evento

`$(document).ready()`

= `$(().ready())` = `$("document").ready()`

Evita que qualquer código jQuery seja executado
antes do documento carregar completamente

Semão ações podem falhar, como : tentar alterar elementos
que ainda não foram criados ou carregados.

→ Noções de programação por objetos:

um objeto possui um conjunto de

- propriedades
- métodos
- eventos

Métodos p/ a manipulação do DOM e jQuery [GET]

3 métodos de recuperação de informações:

text () retorna o conteúdo de texto dos elementos selecionados

html () dos elementos selecionados

val () o valor dos campos de um formulário (<input>)

attr () método usado p/ obter valores de atributos

ex.: atributo href

```
$document.ready(function(){  
    $("button").click(function () {  
        alert($("#nextPage").attr("href"));  
    });  
});
```

... .

SET

Os métodos são os mesmos, mas a sintaxe é distinta

```
$(document).ready(function(){
    $("button").click(function () {
        $("title").text("This is a text");
        $("#pageTitle").html("This is a <strong>text</strong>");
        $("#name").val("Dolly Duck");
        $("#nextPage").attr("href", "http://www.ua.pt");
    });
});
```

Métodos p/ manipulação de CSS :

Obter e definir classes CSS
get set

addClass() adiciona 1 ou + classes aos elementos selecionados

removeClass() remove 1 ou mais classes

toggleClass() alterna entre adicionar / remover classes...

css() define ou retorna o atributo com todos os estilos

```
<script>
$(document).ready(function () {
    $(".bannerTop").css({"background-color": "yellow", "font-size": "200%"});
});
</script>
```

Isto é notação JSON

Eventos jQuery captura

→ Todas as ações que um visitante realiza numa página web geram eventos que simulam essas ações.

ex.: * mover o rato sobre um elemento

* Selecionar um botão de opção

* círcular um elemento

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

→ A maioria dos eventos JavaScript têm um método jQuery equivalente

atribuir um
evento de clique
associado a
todos os parágrafos

```
<script>
$(document).ready(function(){
    $("p").click(function(){
        // o que será feito deve ser programado aqui!
    });
});
</script>
```

dois eventos:
* document.ready()
* click()

Exemplo :



```

var errors = $("span.field-validation-error").length;
console.log("errors =", errors);
if (errors > 0) {
    $("#submit_form").html("<i class='fa fa-save'></i> <span class='position-absolute top-0 start-100 translate-middle badge rounded-pill bg-danger' title='O Formulário tem " + errors + " Erros'>" + errors + "</span>");
} else {
    $("#submit_form").html("<i class='fa fa-save'></i>");
    if (buttonId.indexOf("submit_form") == 0) {
        var JSONdata = generateJSON();
        //--- Envia dados para o servidor...
    }
}

```

JSON

JavaScript Object
Notation

é um formato de troca / intercâmbio de dados simples que é independente de linguagem de programação utilizada.

→ linguagem auto-descriptiva : fácil de entender

→ Usa a sintaxe JS mas o formato JSON é somente texto

→ O texto pode ser lido e usado como formato de dados por qualquer linguagem de programação

```

<html>
<body>
    <h2>Manipulação JSON em JavaScript</h2>
    <p id="demo"></p>
    <script>
        var text = '{"name":"Zé Maria Pincel","address":"Rua 8 de Maio, 5","phone":"351 123456789"}';
        var obj = JSON.parse(text);
        document.getElementById("demo").innerHTML =
            obj.name + "<br>" +
            obj.address + "<br>" +
            obj.phone;
    </script>
</body>
</html>

```

AJAX

funcionalidade que permite que páginas HTML troquem dados com um servidor para atualizar apenas partes dessa página mas sem ser necessário recarregar JavaScript and XML toda a página.

↳ O carregamento de dados é feito em 2º plano e o resultado é exibido na página Web, sem recarregar a página.

ex.: Facebook, Gmail, Google, Maps ...

(Sempre que desligamos a página novos dados vão sendo carregados dinamicamente.)

Exemplo de uma chamada AJAX em jquery

```
var data = "abc";
$.ajax({
  type: "GET",
  url: "http://somewhere/somepage/somedetails",
  data: {
    "data": data
  },
  dataType: "json",
  * success: function (datas, textStatus, jqXHR) {
    //if received a response from the server
  },
  * error: function (jqXHR, textStatus, errorThrown) {
    //if there was no response from the server
  },
  * beforeSend: function (jqXHR, settings) {
    //capture the request before it was sent to server (in send calls)
  },
  * complete: function (jqXHR, textStatus) {
    //this is called after the response or error functions are finished
    //so that we can take some action
  }
});
```

Exemplo

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
  <div id="div1"><h2>Let jQuery AJAX Change This Text</h2></div>
  <button>Get External Content</button>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
  <script>
    $(document).ready(function () {
      $("button").click(function () {
        $.ajax({
          url: "demo_test.txt",
          success: function (result) {
            $("#div1").html(result);
          }
        });
      });
    });
  </script>
</body>
</html>
```

Let jQuery AJAX Change This Text

[Get External Content](#)

jQuery and AJAX is FUN!

This is some text in a paragraph.

[Get External Content](#)

Métodos AJAX em jQuery

\$ get()

`$.get(URL,callback);`

```
$(“button”).click(function () {
  $.get(“getPageAddress”, function (data, status) {
    alert(“Data: ” + data + “\nStatus: ” + status);
  });
});
```

\$ post()

`$.post(URL,data,callback);`

```
$(“button”).click(function () {
  $.post(“postPageAddress”,
  {
    name: “Biden, Joe”,
    city: “Washington”
  },
  function (data, status) {
    alert(“Data: ” + data + “\nStatus: ” + status);
  });
});
```

Manipulação do DOM + AJAX

(jQuery + Asynchronous JavaScript And XML + JSON)

Exemplo: Elaborar um formulário onde o utilizador pode ir buscar, remotamente e sem voltar a carregar o documento as condições meteorológicas de uma qualquer cidade no mundo!

The screenshot shows a web browser window titled "jQuery Weather Test". The address bar displays "http://localhost:37875/jQueryWeather.html". The main content area shows the following information for Aveiro, PT:

- City Name: Aveiro / PT
- Coordinates: Lon (°): -8.65 / Lat (°): 40.64
- weather: Sky is Clear
- temp: 290.334°K / 17.18400000000026°C
- pressure: 1007.32

Below this, under the heading "Dados recolhidos", is a JSON object:

```
{"coord":{"lon":-8.65,"lat":40.64},"weather":[{"id":800,"main":"Clear","description":"Sky is Clear","icon":"01n"}],"base":"cmc stations","main":{"temp":290.334,"pressure":1007.32,"humidity":77,"temp_min":290.334,"temp_max":290.334,"sea_level":1040.01,"grnd_level":1007.32,"speed":0.92,"deg":204.501}, "clouds":{"all":0}, "dt":1447010744, "sys": {"message":0.0051, "country":"PT", "sunrise":1446966804, "sunset":1447003369}, "id":2742611, "name":"Aveiro", "cod":200}
```



Quem fornece a informação do tempo?

<http://api.openweathermap.org/>

Mensagem de sucesso "beautified"

Erro

```
{"cod": "404", "message": "Error: Not found city"}
```

Sucesso

```
{"coord": {"lon": -8.61, "lat": 41.15}, "weather": [{"id": 802, "main": "Clouds", "description": "scattered clouds", "icon": "03n"}], "base": "stations", "main": {"temp": 301.15, "pressure": 1018, "humidity": 58, "temp_min": 301.15, "temp_max": 301.15}, "visibility": 16093, "wind": {"speed": 7.2, "deg": 50}, "clouds": {"all": 40}, "dt": 1447008780, "sys": {"type": 1, "id": 819, "message": 0.03, "country": "PT", "sunrise": 1446966858, "sunset": 1447003295}, "id": 2735943, "name": "Porto", "cod": 200}
```

```
{ "coord": { "lon": -8.61, "lat": 41.15 },  
  "weather": [ { "id": 803, "main": "Clouds", "description": "broken clouds", "icon": "04d" } ],  
  "base": "stations",  
  "main": { "temp": 284.62, "pressure": 1020, "humidity": 81, "temp_min": 283.71, "temp_max": 285.15 },  
  "visibility": 10000,  
  "wind": { "speed": 1.5, "deg": 160 },  
  "clouds": { "all": 75 },  
  "dt": 1573556719,  
  "sys": { "type": 1, "id": 6900, "country": "PT", "sunrise": 1573543096, "sunset": 1573579138 },  
  "timezone": 0,  
  "id": 2735943, "name": "Porto", "cod": 200 }
```

{JSON}
JavaScript Object Notation

<http://api.openweathermap.org/data/2.5/weather?q=porto,pt&appid=b2b1df463182c3cca5276e9d3267cc95> (#valid@12nov19)

O código da interface

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>jQuery Weather Test</title>
    <link href="https://fonts.googleapis.com/css?family=Roboto:300" rel="stylesheet" type="text/css">
    <link href="../Content/bootstrap.min.css" rel="stylesheet" />
</head>
<body>
    <div class="container">
        <select id="citySelector" class="form-select">
            <option value="">Select a city name</option>
            <option value="Aveiro, PT">Aveiro</option>
            <option value="Porto, PT">Porto</option>
            <option value="Paris, FR">Paris</option>
            <option value="London, UK">Londres</option>
            <option value="New York, USA">Nova Iorque</option>
        </select>
        <table class="table table-striped d-none">
            <tr><td class="col-xs-2">City Name:</td><td class="col-xs-10" id="cityName"></td></tr>
            <tr><td class="col-xs-2">Coordinates:</td><td class="col-xs-10" id="coordinates"></td></tr>
            <tr><td class="col-xs-2">weather:</td><td class="col-xs-10" id="weather"></td></tr>
            <tr><td class="col-xs-2">temp:</td><td class="col-xs-10" id="temp"></td></tr>
            <tr><td class="col-xs-2">pressure:</td><td class="col-xs-10" id="pressure"></td></tr>
            <tr><td class="col-xs-2">Dados recolhidos</td>
                <td class="col-xs-10" ><pre id="allData"></pre></td>
            </tr>
        </table>
    </div>
```

O código de manipulação da informação

```
<script src="../Scripts/jquery-3.5.1.min.js"></script>
<script>
$(document).ready(function () {
    $("#citySelector").change(function () {
        $.ajax({
            url: "http://api.openweathermap.org/data/2.5/weather",
            data: {
                q: $("#citySelector").val(),
                APPID: 'b2b1df463182c3cca5276e9d3267cc95'
            },
            success: function (data) {
                if (data.name){
                    $('table').removeClass('d-none');
                    $("#cityName").html(data.name + ' / ' + data.sys.country);
                    $("#coordinates").html('Lon (°): ' + data.coord.lon + ' / Lat (°):' + data.coord.lat);
                    $("#weather").html(data.weather[0].description);
                    $("#temp").html(data.main.temp.toString() + '°K / ' + (data.main.temp - 273.15).toString() + '°C');
                    $("#pressure").html(data.main.pressure);
                    $("#allData").html(JSON.stringify(data, null, 4).replace(/\n/g, "<br>"));
                }
                else {
                    $('table').addClass('d-none');
                    alert(data.message);
                }
            },
            error: function () {
                $('table').addClass('d-none');
                alert('Erro!');
            }
        });
    });
});
</script>
</body>
```

jQuery UI
(biblioteca)

→ coleção de widgets de interface gráfica, efeitos visuais animados e temas implementados com jQuery, CSS's e HTML.

jQuery User Interface

pequena app com funcionalidade limitada que pode ser instalada e executada dentro de uma página web.

→ Assenta sobre a biblioteca jQuery e possui muitas funcionalidades cobertas pelo Bootstrap.



jQuery UI Widgets

Acordeão – grupo de conteúdos organizados na forma de um acordeão

Autocomplete – caixas que permitem o preenchimento automático com base no que o utilizador digita



Button – botão com apresentação melhorada.

Permite que botões rádio e caixas de seleção sejam convertidos em botões

Datepicker – componente com calendário para recolha de campos com datas



Dialog – caixas de diálogo colocadas em cima de outros conteúdos

Menu – componente que permite mostrar e gerir os elementos de um menu

Progressbar – barras de progresso – animadas, ou não

Slider – barras de arrastamento totalmente personalizáveis

Spinner – gera o valor de um número com setas

Tabs - manipulação interface com tabuladores

Tooltip - Mostrar uma dica sobre um determinado conteúdo ou operação

jQuery UI Widgets

Acordeão – grupo de conteúdos organizados na forma de um acordeão

Autocomplete - caixas que permitem o preenchimento automático com base no que o utilizador digita

Button - botão com apresentação melhorada.

Permite que botões rádio e caixas de seleção sejam convertidos em botões

Datepicker – componente com calendário para recolha de campos com datas

Dialog - caixas de diálogo colocadas em cima de outros conteúdos

Menu – componente que permite mostrar e gerir os elementos de um menu

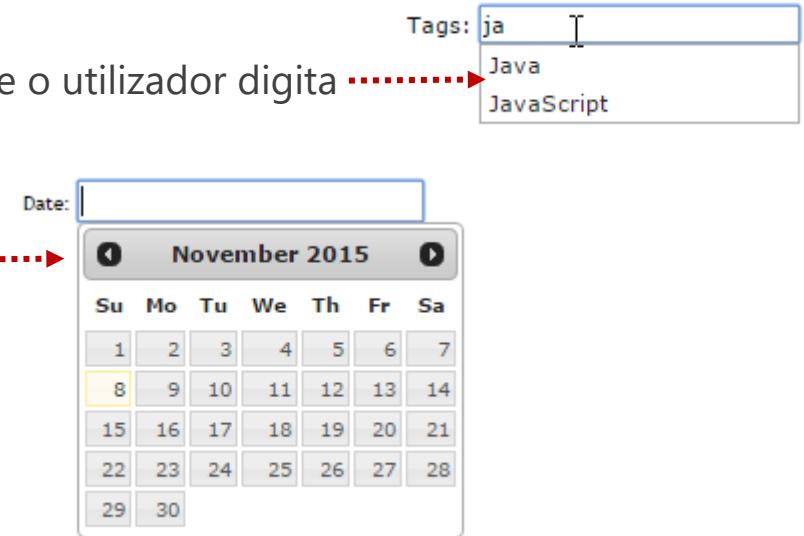
Progressbar - barras de progresso – animadas, ou não

Slider – barras de arrastamento totalmente personalizáveis

Spinner – gere o valor de um número com setas

Tabs - manipulação interface com tabuladores

Tooltip - Mostrar uma dica sobre um determinado conteúdo ou operação



Exemplos DOM

Para uma lista completa de widgets jQuery UI, ver <http://jqueryui.com/widget/>

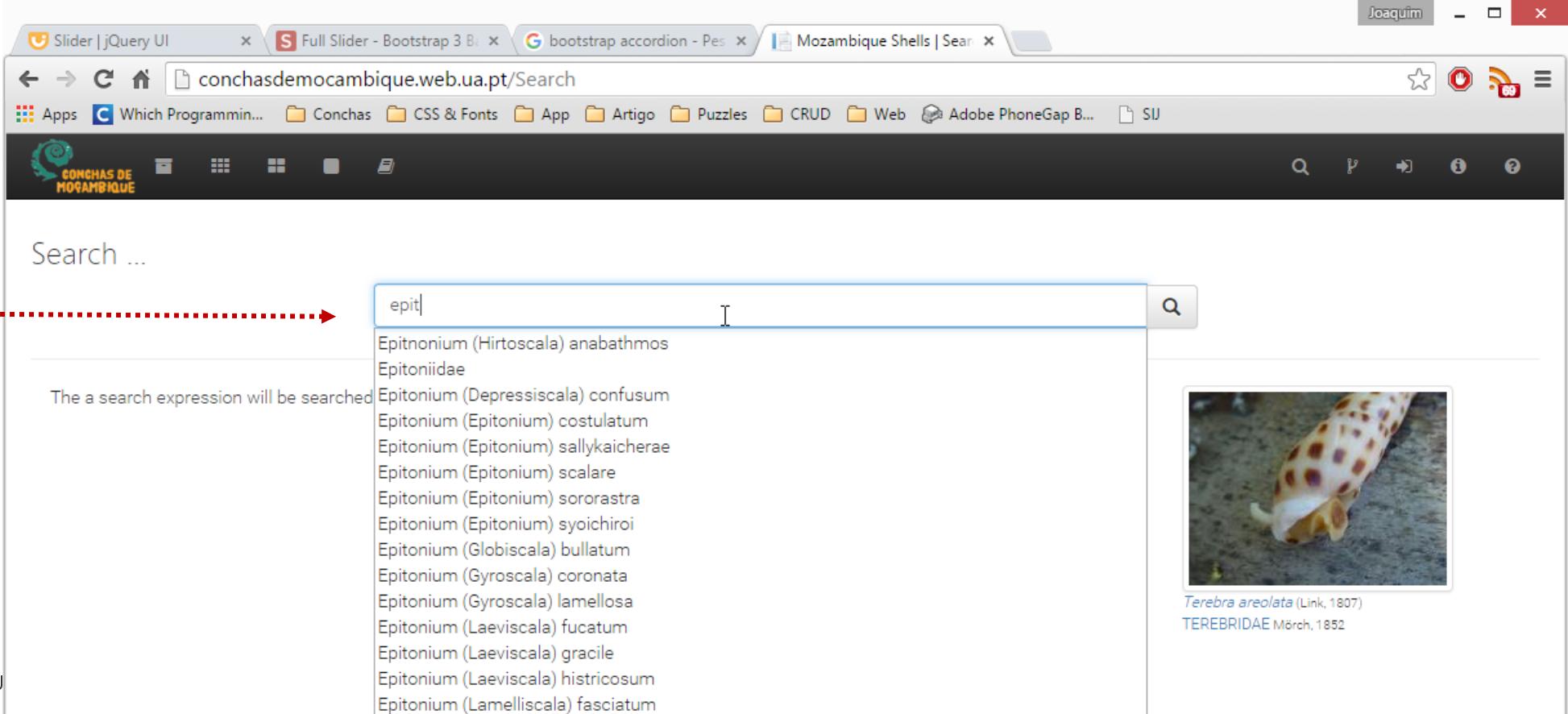
```
<div id="draggable" class="ui-widget-content">
  <p>Drag me around</p>
</div>
```

Esta propriedade pode ser “interessante” para aplicar às peças de xadrez da aula 2; isso faz com que as peças se possam mover dentro da página...

```
<script type="text/javascript">
// Make #draggable draggable
$(function () {
  $("#draggable").draggable();
});
</script>
```

Exemplo de autocomplete

Este exemplo foi retirado do site <http://conchasdemocambique.web.ua.pt> e permite ao utilizador procurar pelo nome de uma classe, família, subfamília ou espécie de conchas.



The screenshot shows a web browser window titled "Joaquim" displaying the "Mozambique Shells | Search" page at conchasdemocambique.web.ua.pt/Search. The search bar contains the text "epit". A dropdown menu lists various species names starting with "epit", such as Epitonium (Hirtoscala) anabathmos, Epitoniiidae, Epitonium (Depressiscala) confusum, Epitonium (Epitonium) costulatum, Epitonium (Epitonium) sallykaicherae, Epitonium (Epitonium) scalare, Epitonium (Epitonium) sororastra, Epitonium (Epitonium) syoichiroi, Epitonium (Globiscala) bullatum, Epitonium (Gyroscala) coronata, Epitonium (Gyroscala) lamellosa, Epitonium (Laeviscalata) fucatum, Epitonium (Laeviscalata) gracile, Epitonium (Laeviscalata) histricosum, and Epitonium (Lamelliscala) fasciatum. To the right of the dropdown, there is a small image of a brown and white patterned sea snail and the text "Terebra areolata (Link, 1807) TEREBRIDAE Mörch, 1852". A red arrow points from the text "The a search expression will be searched" to the search bar.

O código

Nota:

Este exemplo não é repetível fora do contexto porque o webService que serve a pesquisa ([DynamicShellSearch.asmx/SearchData](#)) está, intencionalmente, protegido de modo a só poder ser utilizado pelo próprio site.

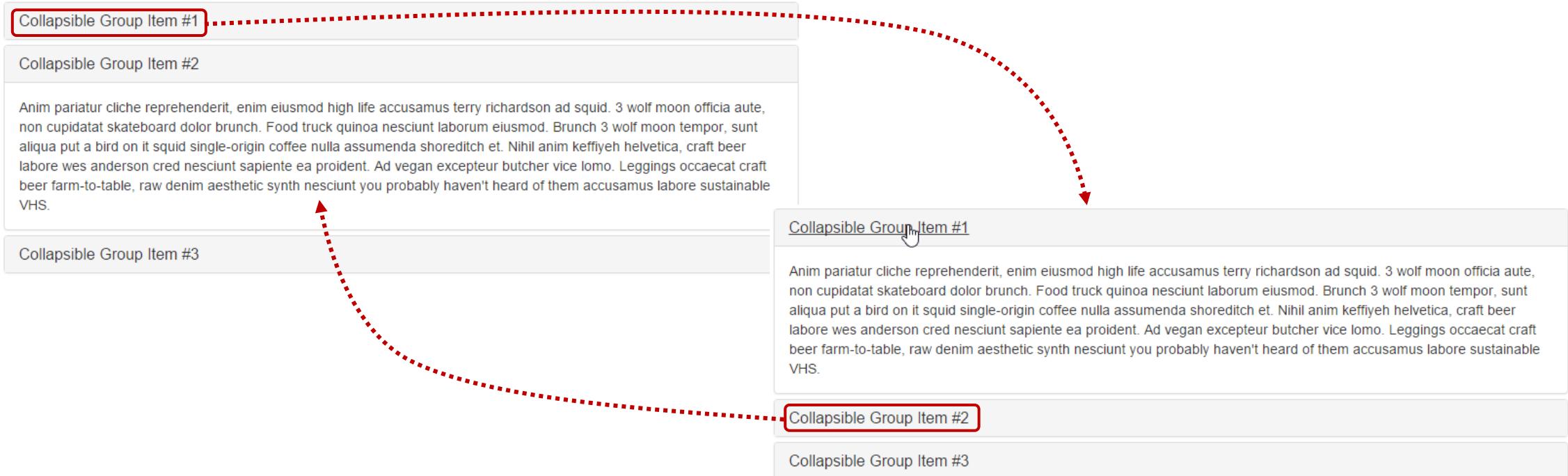


```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <link href="Content/bootstrap.min.css" rel="stylesheet" />
    <link href="Content/themes/ui-darkness/jquery.ui.base.css" rel="stylesheet" />
</head>
<body>
    <div class="container">
        <input class="form-control" ID="SearchText" placeholder="Search expression" />
    </div>
    <script src="Scripts/jquery-2.1.4.min.js"></script>
    <script src="Scripts/jquery-ui-1.11.4.min.js"></script>
    <script type="text/javascript">
        $(document).ready(function () {
            $("#SearchText").autocomplete({
                minLength: 4, ←.....→
                source: function (request, response) {
                    $.ajax({
                        type: "POST",
                        contentType: "application/json; charset=utf-8",
                        url: "DynamicShellSearch.asmx/SearchData",
                        data: "{ 'DName': '" + $('#SearchText').val() + "'}",
                        dataType: "json",
                        success: function (data) {
                            response(data.d);
                        },
                        error: function (result) {
                            alert(result.statusText);
                        }
                    });
                });
            });
        });
    </script>
</body>
</html>
```

Bootstrap com jQuery

Bootstrap Collapse ≈ jQueryUI Accordion

<https://getbootstrap.com/docs/5.0/components/collapse/>



Collapsible Group Item #1

Collapsible Group Item #2

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

Collapsible Group Item #3

Collapsible Group Item #1

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

Collapsible Group Item #2

Collapsible Group Item #3

Bootstrap Buttons ≈ jQueryUI Button

<https://getbootstrap.com/docs/5.0/components/buttons/>

EXAMPLE

Loading state

```
<button type="button" id="myButton" data-loading-text="Loading..." class="btn btn-primary"
autocomplete="off">
  Loading state
</button>

<script>
  $('#myButton').on('click', function () {
    var $btn = $(this).button('loading')
    // business logic...
    $btn.button('reset')
  })
</script>
```

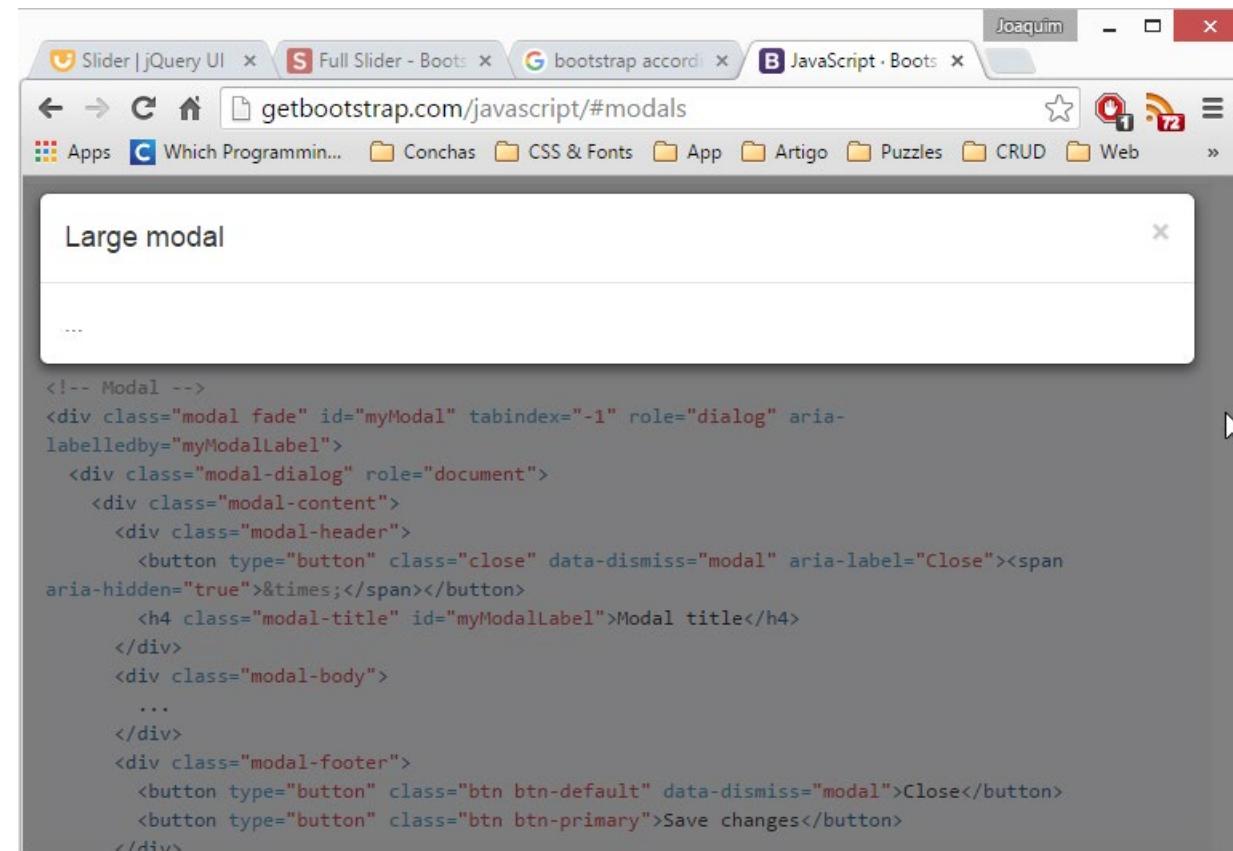
Copy

Checkbox 1 (pre-checked) Checkbox 2 Checkbox 3

Checkbox 1 (pre-checked) Checkbox 2 Checkbox 3

Bootstrap Modal ≈ jQueryUI Dialog

<https://getbootstrap.com/docs/5.0/components/modal/>



Bootstrap Tabs ≈ jQueryUI Tabs

<https://getbootstrap.com/docs/5.0/components/navs-tabs/>



Raw denim you probably haven't heard of them jean shorts Austin. Nesciunt tofu stumptown aliqua, retro synth master cleanse. Mustache cliche tempor, williamsburg carles vegan helvetica. Reprehenderit butcher retro keffiyeh dreamcatcher synth. Cosby sweater eu banh mi, qui irure terry richardson ex squid. Aliquip placeat salvia cillum iphone. Seitan aliquip quis cardigan american apparel, butcher voluptate nisi qui.



Food truck fixie locavore, accusamus mcsweeney's marfa nulla single-origin coffee squid. Exercitation +1 labore velit, blog sartorial PBR leggings next level wes anderson artisan four loko farm-to-table craft beer twee. Qui photo booth letterpress, commodo enim craft beer mlkshk aliquip jean shorts ullamco ad vinyl cillum PBR. Homo nostrud organic, assumenda labore aesthetic magna delectus mollit. Keytar helvetica VHS salvia yr, vero magna velit sapiente labore stumptown. Vegan fanny pack odio cillum wes anderson 8-bit, sustainable jean shorts beard ut DIY ethical culpa terry richardson biodiesel. Art party scenester stumptown, tumblr butcher vero sint qui sapiente accusamus tattooed echo park.

A funcionalidade dos Tabs é muito semelhante à do Accordeon, ou seja, seccionar a quantidade de informação apresentada mas uma faz o seccionamento na vertical (acordeon) enquanto a outra faz na horizontal (Tabs)

Exemplos:

Mozambique Shells | Spec x conchasdemocambique.web.ua.pt/Specie?Shell_ID=646

Identification Bibliography **Images**

Shell ID: 646
 Specie: Tutufa (*Tutufa*) *bubo* (Linnaeus, 1758)
 Family: BURSIDAE Thiele, 1929
 Author, Year: (Linnaeus, 1758)

Dimensions min-max (mm): 150-200 mm
 Occurrence area: Moçambique
 Habitat: Coral reef
 Depth: Shallow subtidal

Exists in IL collection:
 IL Collection number: 598
 Unidentified:
 Visible:

Last update: 2009/08/25 16:45:39

© 2011-2015 - Conchas de Moçambique

Mozambique Shells | Spec x conchasdemocambique.web.ua.pt/Specie?Shell_ID=646

Identification Bibliography **Images**

Shell owner: Isabel Leitão
 Photographer: Isabel Leitão
 Date taken: 2008/08/31
 Camera: Sony Cybershot
 Details: Default: | Habitat: | Detail: | Copyright:
 Last update: 2015/10/29 13:49:16



Photographer: Isabel Leitão
 Date taken: 2015/10/27
 Camera: Sony Cybershot
 Details: Default: | Habitat: | Detail: | Copyright:
 Last update: 2015/10/30 12:11:25



© 2011-2015 - Conchas de Moçambique
 conchasdemocambique.web.ua.pt/Specie?Shell_ID=646#tabs-3

Bootstrap Tooltip/Popover ≈ jQueryUI Tooltips

<https://getbootstrap.com/docs/5.0/components/tooltips/>

<https://getbootstrap.com/docs/5.0/components/popovers/>

Tooltip on the left

Tooltip on the top

Tooltip on the bottom

Tooltip on the right

EXAMPLE

Click to toggle popover

Popover title

And here's some amazing content. It's very engaging. Right?

EXAMPLE

Dismissible popover

Dismissible popover

And here's some amazing content. It's very engaging. Right?

Bootstrap Carousel ≈ jQueryUI Slider

<https://getbootstrap.com/docs/5.0/components/carousel/>

Funcionalidade parcial.

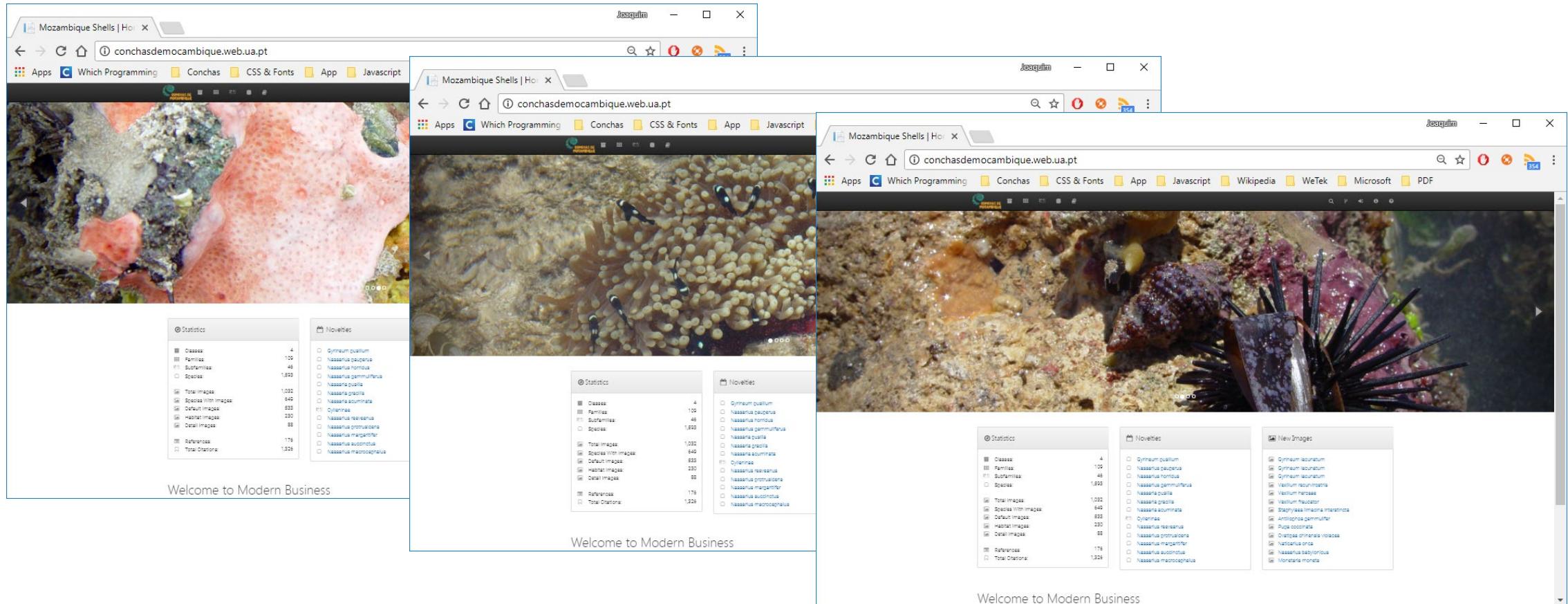
O Bootstrap Carousel já foi introduzido em aula anterior. Estamos agora em condições de perceber “como o programar”.

Exemplo:

```
<div id="carousel-example-generic" class="carousel slide" data-ride="carousel">
  <!-- Indicators -->
  <ol class="carousel-indicators">
    <li data-target="#carousel-example-generic" data-slide-to="0" class="active"></li>
    <li data-target="#carousel-example-generic" data-slide-to="1"></li>
    <li data-target="#carousel-example-generic" data-slide-to="2"></li>
    <li data-target="#carousel-example-generic" data-slide-to="3"></li>
  </ol>
  <!-- Carousel
  ----- -->
  <div id="myCarousel" class="carousel slide">...</div>
  <!-- /.carousel -->
</div>
<script>
  $('.carousel').carousel({
    interval: 10000
  })
</script>
```

O elemento com a classe `carousel`, deve mudar de 10.000 em 10.000 milissegundos, ou seja de 10 em 10 segundos.

Exemplo



Knockout

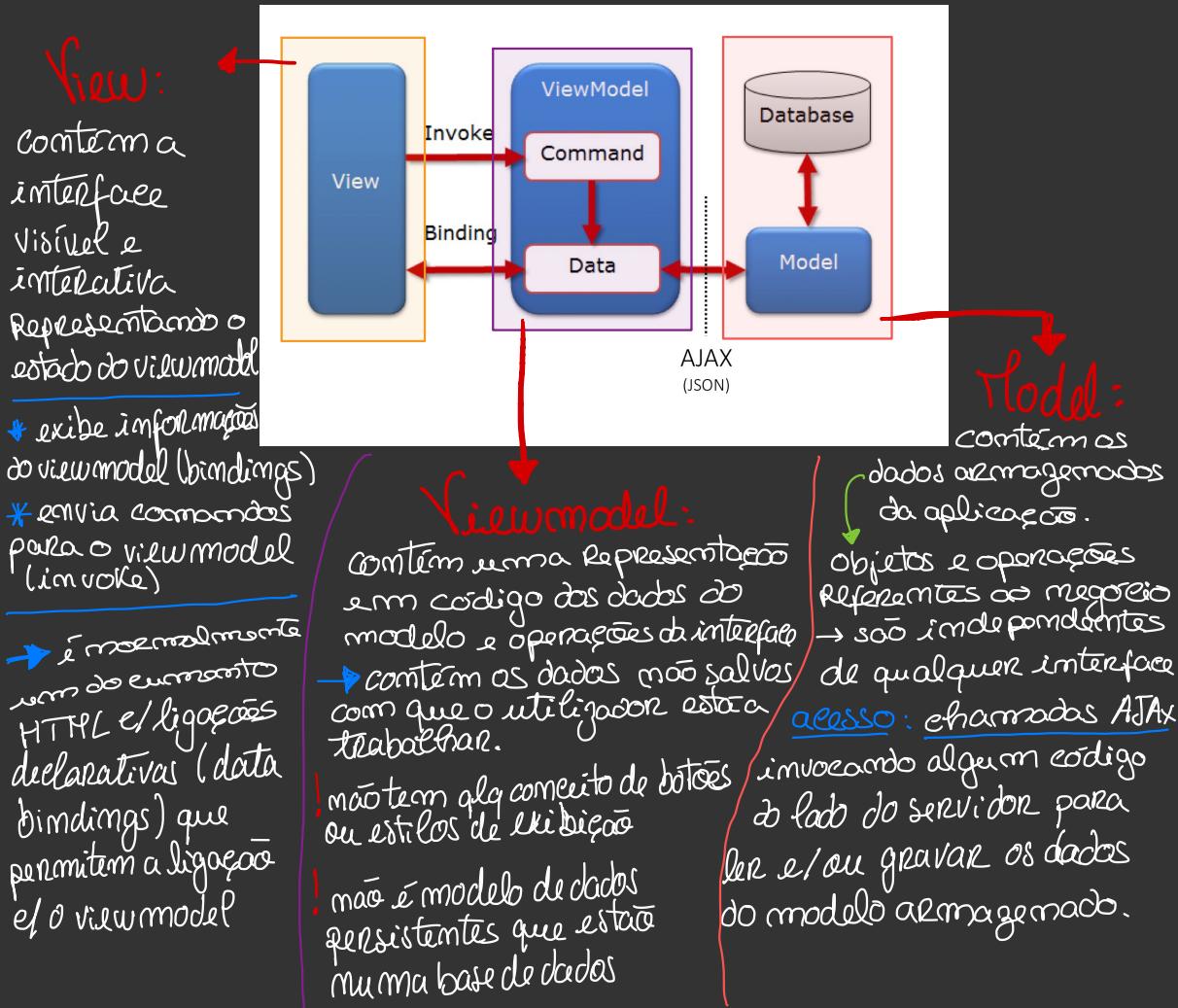
antes de Knockout:

Model-View-ViewModel (MVVM)

Sóndez
Sarah!!

→ Padrão de design para criar interfaces.

Deseja-se como manter uma interface de utilizador dividindo-a em 3 partes:



libraria Knockout JS



biblioteca JavaScript que ajuda a criar interfaces de utilizador de exibição e edição ricas e responsivas com um modelo de dados subjacente limpo.

→ Sempre que há secções da interface de utilizador que necessitam de actualização dinâmica



KO ajuda na implementação de forma mais simples e eficiente que utilizando apenas js ou jquery.

Principais características:

Vinculações declarativas associa elementos do DOM a um modelo de dados através de uma sintaxe concisa e legível

Actualização automática da interface com o utilizador quando o modelo de dados é alterado

Acompanhamento de dependências cadeias de relações entre os dados do modelo de modo a transformá-los e combiná-los

Templating gera rapidamente interfaces de utilizador sofisticadas como uma função dos dados do modelo

outras características:

- * livre, código aberto
- * JavaScript puro - função e/ou framework web
- + Sem dependências
- * pequeno e leve
- * Suporta todos os navegadores habituais, mesmo os antigos
- * totalmente documentado

Como usar o knockout? (2)

```
<!DOCTYPE html>
<html>
<head>
    <title>O meu primeiro teste knockout</title>
    <meta charset="utf-8" />
</head>
<body>
    O meu nome é <span data-bind="text: personName"></span>
    <script src="../Scripts/knockout-3.4.0.js"></script>
    <script>
        var myViewModel = {
            personName: 'Zé Maria',
            personAge: 45
        };
        ko.applyBindings(myViewModel);
    </script>
</body>
</html>
```



p/ criar um viewmodel e KO,
basta declarar qd objeto js (json)

View do viewmodel usando uma
vinculação declarativa

ATIVAR o
KNOCKOUT

Observáveis e dependências (`ko.observable()`)(1)

<http://knockoutjs.com/documentation/observables.html>

Já vimos como criar um viewmodel básico e como exibir uma das suas propriedades (text) usando uma ligação mas um dos principais benefícios do KO é que ele atualiza a interface (view) do utilizador automaticamente quando o viewmodel muda.

Pergunta: Como é que o KO pode saber quando as partes do viewmodel mudam?

Resposta: é preciso declarar as propriedades do seu modelo como observáveis!

Os observáveis são objetos JavaScript especiais que podem notificar os assinantes sobre as alterações e podem detectar dependências automaticamente.

Observáveis e dependências (`ko.observable()`)(2)

Para utilizar variáveis observáveis, reescreve-se o.viewmodel anterior da seguinte maneira:

```
var myViewModel = {  
    personName: ko.observable('Zé Maria'),  
    personAge: ko.observable(45)  
};
```

Não é preciso alterar a view - a sintaxe de ligação de dados é a mesma.

A diferença é que agora a view é capaz de detectar alterações da viewmodel, quando isso acontecer, atualizará a informação na view automaticamente.

Observáveis e dependências (`ko.observable()`)(3)

Problema: Nem todos os browser suportam suportam operações de leitura (get) e escrita (set) de JavaScript (incompatibilidades entre implementações do JavaScript), portanto, por questões de compatibilidade, os objetos `ko.observable` são funções.

- Para ler o valor atual do observável, basta chamar o observável sem parâmetros.
Do exemplo, `myViewModel.personName()` retornará 'Zé Maria', e `myViewModel.personAge()` retornará 45.
- Para escrever um novo valor no observável, invoca-se o observável e passa-se o novo valor como parâmetro.
Por exemplo, `myViewModel.personName('Maria')` irá alterar o valor de nome para 'Maria'.

Arrays de observáveis (ko.observableArray([]))

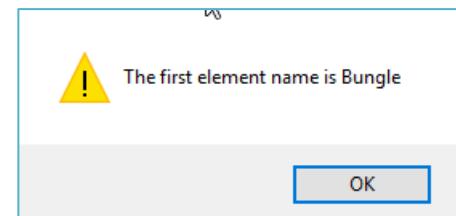
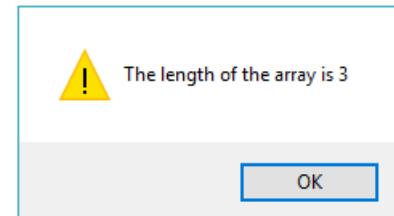
<http://knockoutjs.com/documentation/observableArrays.html>

Já vimos que, caso se pretenda detectar e responder a alterações num objeto, usamos observáveis.

Se pretendermos detectar e responder a alterações numa coleção de objetos, deveremos utilizar um `observableArray`.

Esta possibilidade é particularmente útil em cenários em que se exibem ou editam vários valores e são necessárias seções repetidas da interface para fazer aparecer e desaparecer à medida que os itens são adicionados e/ou removidos.

```
// This observable array initially contains three objects
var myObservableArray = ko.observableArray([
    { name: "Bungle", type: "Bear" },
    { name: "George", type: "Hippo" },
    { name: "Zippy", type: "Unknown" }
]);
alert('The length of the array is ' + myObservableArray().length);
alert('The first element name is ' + myObservableArray()[0].name);
```



Observáveis calculadas(ko.computed)

Suponha que já tem um observável para firstName, e outro para lastName, e deseja exibir o nome completo?

É aí que os **observáveis calculados** são úteis - são funções que dependem de um ou mais observáveis e serão atualizados automaticamente sempre que **alguma das suas dependências mudarem.**

```
O meu nome é <span data-bind="text: fullName"></span>
```

```
function AppViewModel() {
    var self = this;

    self.firstName = ko.observable('Bob');
    self.lastName = ko.observable('Smith');
    self.fullName = ko.computed(function () {
        return self.firstName() + " " + self.lastName();
    });
}
```

KO bindings

text() o elemento DOM associado exibe o valor de texto do seu parâmetro.

→ + útil: ex.: `<spans>` { elementos q/ exibem texto
``

html() o elemento DOM associado exibe o html do seu parâmetro.

→ útil quando os valores viewmodel são sequências de marcação HTML

css() adiciona ou remove uma ou mais classes CSS ao el. DOM ass.

style() adiciona ou remove um ou mais valores de estilo ao el. DOM ass.

```
<div data-bind="style: { color: currentProfit() < 0 ? 'red' : 'black' }>Profit Information</div>
```

attr() fornece uma maneira genérica de definir o valor de qlq atributo p/o el. DOM ass.

→ útil quando precisa de se definir o atributo de Título de um elemento ; o src de uma img ; href de um link e/ base em valores mo seu viewmodel , c/o valor do atributo sendo atualizado automaticamente quando a propriedade muda.

```
<a data-bind="attr: { href: url, title: details }>Relatório</a>

<script type="text/javascript">
    var viewModel = {
        url: ko.observable("http://somesite.com/yearReport.html"),
        details: ko.observable("Relatório e contas referente ao corrente ano")
    };
</script>
```

visible() permite fazer o binding da propriedade visible
a um elemento do qual ficará visível sempre que a
variável de controlo do viewmodel tomar um valor true.

KO controlo de fluxo

foreach() duplica uma marcação p/ cada entrada
em uma matriz e vincula cada cópia dessa
marcação ao item de matriz correspondente.
→ útil p/ renderizar listas ou tabelas

Assumindo que a matriz é um array de observáveis, sempre que adicionar,
remover ou reordenar as entradas da matriz, a ligação atualizará
eficientemente a UI mantendo o sincronismo entre elas - inserindo ou
removendo mais cópias da marcação ou reordenando elementos DOM
existentes, sem afetar quaisquer outros elementos DOM.

Pode aninhar-se qualquer número de bindings **foreach** junto com outras ligações de
controle-fluxo, como **if** ou **with**.

Exemplo de binding com foreach

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo foreach knockout</title>
    <link href="../Content/bootstrap.min.css" rel="stylesheet" />
    <meta charset="utf-8" />
</head>
<body>
    <table class="table table-striped table-condensed">
        <thead>
            <tr><th>First name</th><th>Last name</th></tr>
        </thead>
        <tbody data-bind="foreach: people">
            <tr>
                <td data-bind="text: firstName"></td>
                <td data-bind="text: lastName"></td>
            </tr>
        </tbody>
    </table>
</body>
```

First name	Last name
Bert	Bertington
Charles	Charlesforth
Denise	Dentiste

```
<script src="../Scripts/jquery-3.5.1.min.js"></script>
<script src="../Scripts/bootstrap.min.js"></script>
<script src="../Scripts/knockout-3.5.1.js"></script>
<script type="text/javascript">
    ko.applyBindings({
        people: [
            { firstName: 'Bert', lastName: 'Bertington' },
            { firstName: 'Charles', lastName: 'Charlesforth' },
            { firstName: 'Denise', lastName: 'Dentiste' }
        ]
    });
</script>
</body>
</html>
```

if () faz com que uma seção de marcação apareça no documento somente se a variável de controle especificada for avaliada como verdadeira.

ifnot() = ao **if()**, mas inverte o valor da expressão de avaliação especificada

with ()* cria um novo contexto de visualização, de modo que os elementos descendentes são visualizados no contexto de um objeto especificado.

* **KO binding elements**

click() permite associar um gerador de eventos cuja função JavaScript é chamada quando o elemento DOM associado for clicado.

Exemplo de binding com **with***

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo with knockout </title>
    <link href="../Content/bootstrap.min.css" rel="stylesheet" />
    <meta charset="utf-8" />
</head>
<body>
    <h1 data-bind="text: city"> </h1>
    <p data-bind="with: coords">
        Latitude: <span data-bind="text: latitude"> </span>,
        Longitude: <span data-bind="text: longitude"> </span>
    </p>
    <script src="../Scripts/jquery-3.6.0.min.js"></script>
<script src="../Scripts/knockout-3.5.1.js"></script>
    <script>
        ko.applyBindings({
            city: "London",
            coords: {
                latitude: 51.5001524,
                longitude: -0.1262362
            }
        });
    </script>
</body>
</html>
```

London

Latitude: 51.5001524, Longitude: -0.1262362

Exemplo de binding do evento click *

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>o meu primeiro teste knockout</title>
    <link href="../Content/bootstrap.min.css" rel="stylesheet" />
</head>
<body>
    <div class="container">
        Já carregou <span data-bind="text: numberOfClicks"></span> vezes
        <button data-bind="click: incrementClickCounter" class="btn btn-default">Carrega-me!!!</button>
    </div>
    <script src="../Scripts/jquery-3.6.0.min.js"></script>
    <script src="../Scripts/knockout-3.5.1.js"></script>
    <script>
        var viewModel = {
            numberOfClicks : ko.observable(0),
            incrementClickCounter : function() {
                var previousCount = this.numberOfClicks();
                this.numberOfClicks(previousCount + 1);
            }
        };
        ko.applyBindings(viewModel);
    </script>
</body>
</html>
```

Já carregou 6 vezes Carrega-me!!!

Desafio:

Fazer um formulário para a gestão da classe de uma passagem de avião e do seu respetivo preço – Cenário 1: usando jQuery; Cenário 2 : usando Knockout.

Dados para controlo do formulário:

```
tickets = [  
    { name: "Economy", price: 199.95 },  
    { name: "Business", price: 449.22 },  
    { name: "First Class", price: 1199.99 }  
];
```

Escolha a classe da passagem...

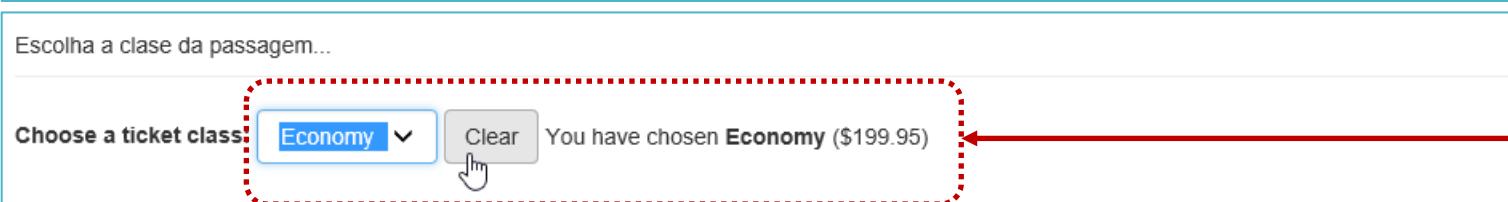
Choose a ticket class:



Enquanto não há uma escolha, o botão está desativado

Escolha a classe da passagem...

Choose a ticket class:



You have chosen Economy (\$199.95)

Quando há uma escolha, o botão fica ativo e é apresentada uma mensagem com a classe escolhida e o preço.

Cenário 1: usando jQuery

```

<!DOCTYPE html>
<html>
<head>
    <title>Exemplo de formulário usando jQuery</title>
    <meta charset="utf-8" />
    <link href="../Content/bootstrap.min.css" rel="stylesheet" />
</head>
<body>
    <div class="container">
        <div class="page-header">Escolha a classe da passagem...</div>
        <form class="form-inline">
            <div class="form-group">
                <label for="flightClasses" class="control-label">Choose a ticket class:</label>
                <select id="flightClasses" class="form-control"></select>
            </div>
            <div class="form-group">
                <button id="clearBtn" class="btn btn-default">Clear</button>
            </div>
            <div class="form-group">
                <p id="choosenTicket" class="form-control-static">You have chosen <b id="choosenClass"></b>
                    ($<span id="choosenPrice"></span>)</p>
            </div>
        </form>
    </div>
    <script src="../Scripts/jquery-3.6.0.min.js"></script>
    <script src="exemplo-jq.js"></script>
</body>
</html>

```

```
$document.ready(function () {
    tickets = [
        { name: "Economy", price: 199.95 },
        { name: "Business", price: 449.22 },
        { name: "First Class", price: 1199.99 }
    ];
    console.log("document ready");
    //--- Inicialização dos elementos html
    console.log("adding <select> options")
    //--- Lista de opções - elemento em branco (a pedir para selecionar ...)
    $('#flightClasses').append($('option', {
        value: '',
        text: 'Choose'
    }));
    //--- Lista de opções - inicialização dos elementos da lista
    $.each(tickets, function (i, ticket) {
        $('#flightClasses').append($('option', {
            value: ticket.price,
            text: ticket.name
        }));
    });
    //--- Disable do botão
    $("#clearBtn").prop("disabled", true);
    //--- Esconder a mensagem
    $("#choosenTicket").addClass("d-none");
});
```

```
//--- Inicialização terminada.
//--- Gestão de eventos ...
$("#flightClasses").change(function () {
    if ($("#flightClasses").val() == "") {
        //--- Disable do botão
        $("#clearBtn").prop("disabled", true);
        //--- Esconder a mensagem
        $("#choosenTicket").addClass("d-none");
    } else {
        //--- Enable do botão
        $("#clearBtn").prop("disabled", false);
        //--- Mostrar a mensagem
        $("#choosenTicket").removeClass("hidden");
        $("#choosenClass").text($("#flightClasses option:selected").text());
        $("#choosenPrice").text($("#flightClasses").val());
    }
});
```

```

<!DOCTYPE html>
<html>
<head>
    <title>Exemplo de formulário usando KO</title>
    <meta charset="utf-8" />
    <link href="../Content/bootstrap.min.css" rel="stylesheet" />
</head>
<body>
    <div class="container">
        <div class="page-header">Escolha a classe da passagem...</div>
        <form class="form-inline">
            <div class="form-group">
                <label for="" class="control-label">Choose a ticket class:</label>
                <select data-bind="options: tickets,
                    optionsCaption: 'Choose...',
                    optionsText: 'name',
                    value: chosenTicket" class="form-control"></select>
            </div>
            <div class="form-group">
                <button data-bind="enable: chosenTicket,
                    click: resetTicket" class="btn btn-default">Clear</button>
            </div>
            <div class="form-group">
                <p data-bind="with: chosenTicket" class="form-control-static">
                    You have chosen <b data-bind="text: name"></b>
                    ($<span data-bind="text: price"></span>)
                </p>
            </div>
        </form>
    </div>
    <script src="../Scripts/jquery-3.6.0.min.js"></script>
    <script src="../Scripts/knockout-3.5.1.js"></script>
    <script src="exemplo-ko.js"></script>
</body>
</html>

```

14/12/2021

©2014-21, JOAQUIM SOUSA PINTO

```
function TicketsViewModel() {
    this.tickets = [
        { name: "Economy", price: 199.95 },
        { name: "Business", price: 449.22 },
        { name: "First Class", price: 1199.99 }
    ];
    this.chosenTicket = ko.observable();
    this.resetTicket = function () { this.chosenTicket(null) }
}
ko.applyBindings(new TicketsViewModel());
```

Só isto ... e mais nada.

Descodificando...

A variável `this.chosenTicket`

fica com o valor escolhido na interface pelo `<select></select>` através da propriedade `value: chosenTicket`

O `<button></button>` é controlado também por este valor através da propriedade `enable: chosenTicket`

A função `this.resetTicket`

é atuada na interface pelo `<button></button>` ativa no código o método `click: resetTicket`" que coloca o valor da variável `this.chosenTicket` em `null`

em consequência dessa alteração na parte do código, na interface, o `<select></select>`, o `<button></button>` e o `<p></p>` são alterados