

Disciplina: Programação Cliente-servidor

Aula 4: JSON e JQuery

Apresentação

Hoje em dia é cada vez mais comum o uso de Web Services REST para comunicação B2C, principalmente no que se refere aos clientes móveis, como Android; isto trouxe para a sintaxe JSON um maior nível de importância, o que faz com que nós precisemos entender essa sintaxe, de forma a viabilizar a comunicação com diversos servidores. Outro assunto de grande relevância é a facilidade com que podemos criar páginas interativas e de visual extremamente agradável ao usarmos as bibliotecas JQuery, diminuindo nossa carga de trabalho na construção de interfaces modernas e responsivas. Portanto, é crucial para nós, desenvolvedores, aprendermos a lidar com as tecnologias JSON e JQuery.

Objetivos

- Descrever a sintaxe JSON (JavaScript Object Notation);
- Examinar os fundamentos das bibliotecas JQuery;
- Usar a biblioteca JQuery UI para a construção de páginas.



Fonte: Jantine Doornbos / Unsplash

JSON

A sintaxe JSON (JavaScript Object Notation) nos permite representar objetos de uma forma muito leve e simples. Note que falamos aqui de objetos, e não de classes, pois JSON não trabalha com classes, mas diretamente nos objetos.

Não trabalhamos com métodos em JSON, mas apenas atributos, os quais são definidos através de pares chave-valor, onde a chave é texto e o valor pode assumir diferentes formatos.

A definição do objeto JSON é iniciada com o uso de chaves, e os pares chave-valor são separados por vírgula.

```
var pessoa1 = {"nome":"Joao", "telefone":"1111-1111"};
```

Após a definição de um objeto neste formato, podemos utilizá-lo como qualquer objeto padrão do JavaScript.

```
alert(p1.nome);
```

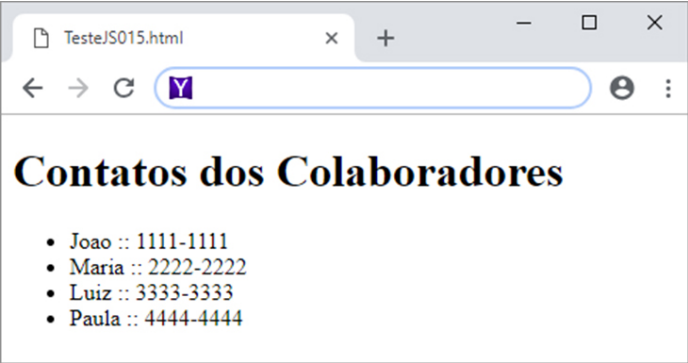
Nós podemos utilizar vários tipos diferentes para o preenchimento do valor, conforme se observa na tabela seguinte.

FORMATO	DESCRIÇÃO	EXEMPLO
Texto	Cadeia de caracteres delimitada por aspas	"nome":"Joao"
Número	Valor numérico qualquer	"ano":2018
Booleano	Aceita true e false	"ligado":true
Nulo	Representa valor nulo	"setor":null
Vetor	Lista de valores delimitada por colchetes	"lista":["red","green","blue"]
Objeto	Objeto JSON delimitado por chaves	{"autor":"Mister K","ano":2018}

Observe que os vetores aceitam diversos tipos de dados, inclusive objetos JSON, como podemos ver no exemplo seguinte.

Exemplo

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
  </head>
  <body>
    <h1>Contatos dos Colaboradores</h1>
    <ul id="alvo">
    </ul>
    <script>
var dados = {"empregados": [
  {"nome":"Joao" , "telefone":"1111-1111"},
  {"nome":"Maria" , "telefone":"2222-2222"},
  {"nome":"Luiz" , "telefone":"3333-3333"},
  {"nome":"Paula" , "telefone":"4444-4444"}
]};
var lista = document.querySelector("#alvo");
for(i=0; i<4; i++){
  var contato = dados.empregados[i].nome + " :: "+
  dados.empregados[i].telefone;
  var novoElemento = document.createElement("LI");
  var texto = document.createTextNode(contato)
  novoElemento.appendChild(texto);
  lista.appendChild(novoElemento);
}
</script>
</body>
</html>
```



Neste exemplo nós temos um objeto JSON denominado *dados*, com apenas um atributo, chamado *empregados*. Este atributo, por sua vez, é um vetor de objetos JSON, cada um com os atributos nome e telefone.

Os elementos da lista são inseridos dinamicamente com uso de DOM, e o texto é formado a partir dos dados presentes no vetor do objeto JSON, onde zero corresponde à primeira posição deste vetor e três corresponde à última.

```
var contato = dados.empregados[1].nome + " :: "
+
  dados.empregados[1].telefone;
// contato recebe o nome e o telefone do segundo elemento do vetor
// contato recebe o valor "Maria :: 2222-2222"
```

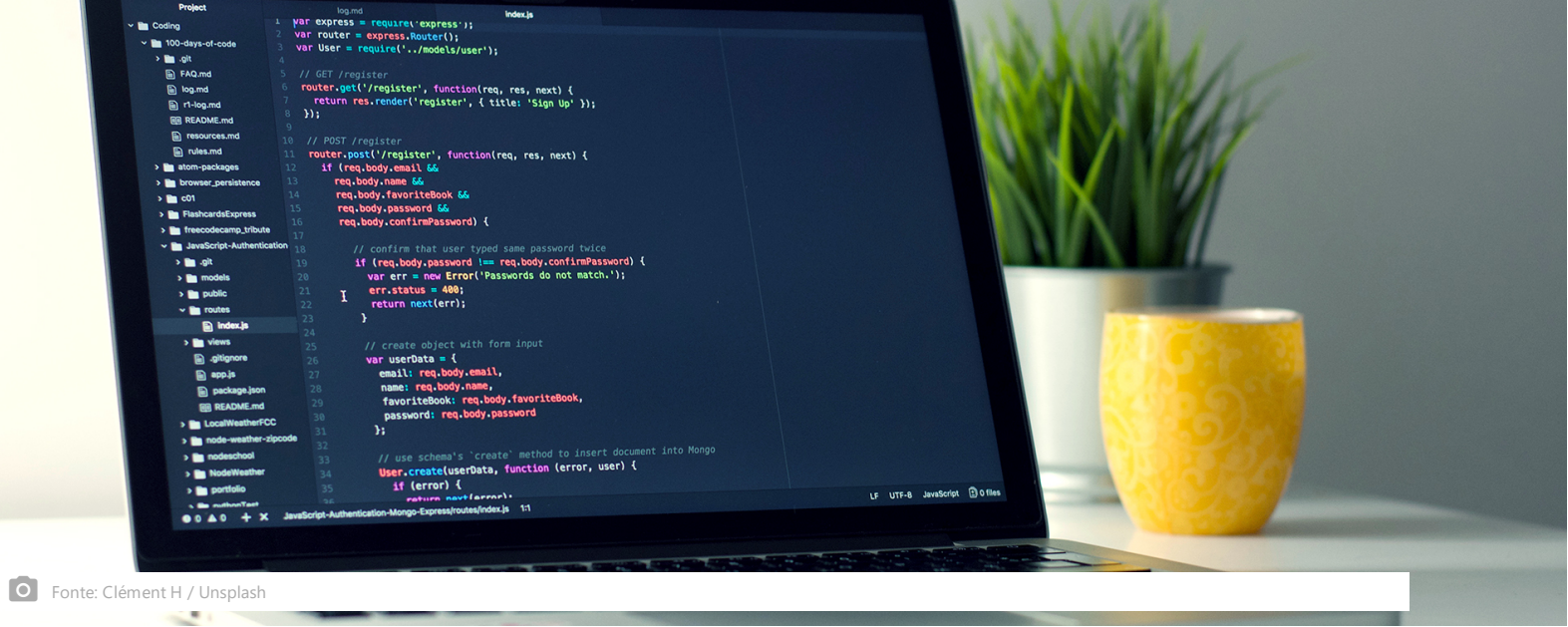
Várias tecnologias atuais utilizam o JSON como formato texto para troca de informações, das quais podemos destacar os Web Services REST, com o uso muito ampliado pelo advento das plataformas móveis, como Android e iOS.

Quando recebemos dados de plataformas como essa, os dados são texto e não objetos JSON, devendo ser transformados no JavaScript para utilização posterior. Para tal, devemos utilizar o método de conversão **JSON.parse**.



Fonte: Sven / Unsplash

```
alert(p1.nome);
```



Fonte: Clément H / Unsplash

Características Gerais do JQuery

Agora que conhecemos as sintaxes JavaScript e JSON, podemos iniciar a utilização de uma biblioteca amplamente adotada pelo mercado no desenvolvimento de sistemas, que é a JQuery. Esta é uma biblioteca que executa do lado cliente, interagindo com os elementos da página e facilitando muitas das tarefas usuais no desenvolvimento das funcionalidades da interface.

A primeira versão foi lançada em janeiro de 2006 e seu criador, Josh Resig, havia publicado anteriormente em seu blog os fundamentos básicos para a construção do JQuery:

“

A premissa para o módulo é a seguinte: Usando o poder dos Seletores do Pseudo-CSS, vincule suas funções Javascript a vários elementos HTML no DOM.

Josh Resig, em 22/08/2005

Saiba mais

Microsoft e Nokia incluíram a biblioteca JQuery em suas plataformas em 2008, sendo adotada posteriormente por outros fabricantes. Hoje em dia, diversos frameworks incorporaram a biblioteca, como dotNet e Prime Faces.


VANTAGENS COM O USO DO JQuery

- 1 Grande redução e reutilização de código, além de garantir a compatibilidade da página com os diversos navegadores do mercado;
- 2 Diversos plugins foram criados a partir da biblioteca, automatizando as mais diversas áreas do desenvolvimento.
- 3 Possibilidade de usarmos temas CSS facilita muito a padronização do ambiente e modificações visuais. Não precisamos modificar a estrutura da página ou o código, bastando alterar os arquivos CSS que constituem o tema para que todo o site criado com JQuery tenha seu aspecto alterado.

Para iniciarmos a utilização do JQuery, precisamos entrar no site jquery.com e baixar os arquivos necessários. O link para download fica disponível logo na primeira página.



Fonte: Alejandro Escamilla / Unsplash

 Clique nos botões para ver as informações.

Production



Biblioteca compactada, sem espaços, comentários e quebras de linha, possuindo cerca de 15% da versão completa.

Development



Versão completa, com todos os comentários e quebras de linha, ideal para o desenvolvedor.

É interessante que utilizemos a versão Development e depois troquemos pela versão Production, quando tivermos de colocar o sistema em produção. O uso da versão compactada irá diminuir o tempo de download para quem for acessar a página.

Seletores no JQuery

TQuando comparada ao JavaScript, a seleção de elementos DOM é muito mais simples com o uso de JQuery. Os seletores são a base funcional da biblioteca quando tratamos do relacionamento com elementos da página.

Observe a diferença:

JavaScript

`document.getElementById("XPTO")`



Jquery

`$("#XPTO")`

De forma geral, o JQuery utiliza seletores semelhantes ao CSS, sendo utilizado, inicialmente, hash para identificadores e ponto para classes.

Vamos observar um pequeno exemplo com seletores destes dois tipos. Não esqueça de copiar o arquivo “jquery.js” para o mesmo diretório do arquivo HTML.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
    <tittle>Exemplo JQuery</tittle>
    <script src="jquery.js">
    </script>
    <script>
      function exibir() {
        $("#alvo").fadeIn("slow");
        $(".par").fadeOut("slow");
      }
    </script>
  </head>
  <body>
    <input type="button" onclick="exibir()"
      value="Clique Aqui!" />
    <p style="display: none; background-color:yellow"
      id="alvo">Olá Mundo</p>
    <p>
      <span class="impar">1</span><span class="part">2</span>
      <span class="impar">3</span><span class="part">4</span>
      <span class="impar">5</span><span class="part">6</span>
      <span class="impar">7</span><span class="part">8</span>
    </p>
  </body>
</html>
```

Neste exemplo temos duas tags **<script>**, sendo a primeira para inclusão da biblioteca JQuery e a segunda para a codificação da função `exibir`, que é chamada a partir do clique sobre o botão.

O parágrafo com identificador **"alvo"** é inicializado, através do CSS, de forma oculta.

```
<p style="display: none; background-color:yellow" id="alvo">
```

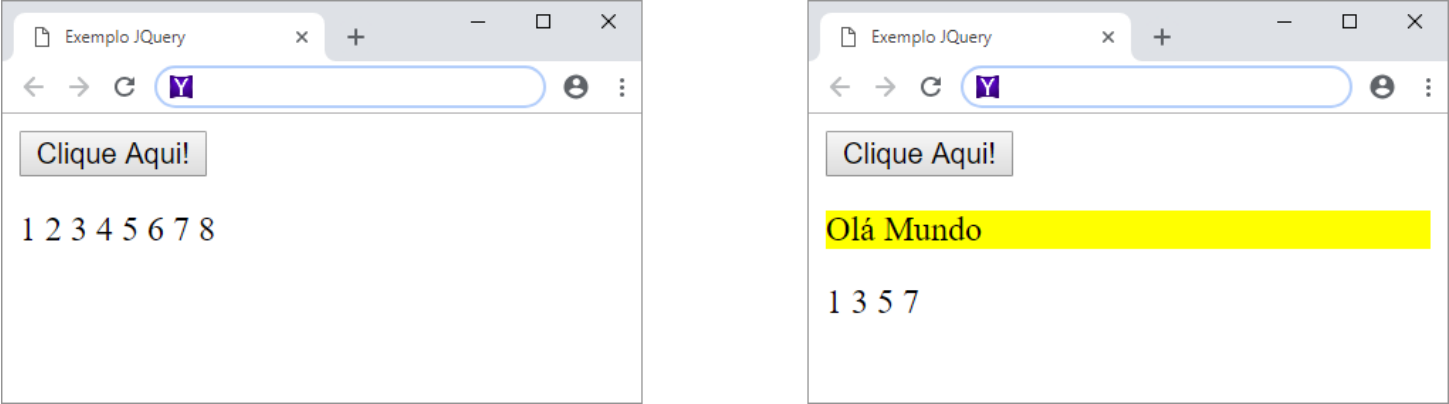
Na primeira linha da função utilizamos o seletor **"#alvo"** e exibimos esse parágrafo na página de forma lenta, com uso de **fadeIn**.

```
$("#alvo").fadeIn("slow");
```

Também temos várias tags ****, algumas da classe **"par"**, outras da classe **"impar"**, e com o uso do seletor **".par"** conseguimos ocultar todos os números pares através de **fadeOut**.

```
$(".par").fadeOut("slow");
```

É fácil perceber que a página é dinâmica e ficará diferente após clicarmos sobre o botão, o que podemos observar nas figuras seguintes, representando o momento inicial de exibição e o efeito do clique.



Podemos observar que o uso de um código muito simples nos proporcionou grande efeito visual e facilidade de controle sobre a página.

E o que pode ser adotado como um seletor?

A tabela seguinte exibe algumas opções.

Tipo	Descrição	Exemplos
Identificador	Faz referência ao id de uma tag.	<code>\$("#alvo")</code>
Classe	Compreende todas as tags com o atributo class equivalente.	<code>\$(".par")</code> <code>\$(".impar")</code>
Elemento	Define o acesso a um tipo de tag na página.	<code>\$("ul")</code> <code>\$("div")</code>
Agrupamento	Combinação de seletores separados por vírgula.	<code>\$(".par, .impar")</code> <code>\$("h1, #alvo, ul")</code>
<code>:first</code>	Primeira ocorrência do seletor.	<code>\$("div:first")</code>
<code>:last</code>	Última ocorrência do seletor.	<code>\$("div:last")</code>
<code>:even</code>	Ocorrências de índice par.	<code>\$("li:even")</code>
<code>:odd</code>	Ocorrências de índice ímpar.	<code>\$("li:odd")</code>
Atributo	Busca por atributo nas tags.	<code>\$("[title*='XPTO']")</code>
Aninhados	Seletores internos a outros seletores, como parágrafos dentro de camadas.	<code>\$("div p")</code> <code>\$(".menu li")</code>
<code>:has</code>	Seleciona caso contenha o segundo seletor.	<code>\$("div:has(ul)")</code>

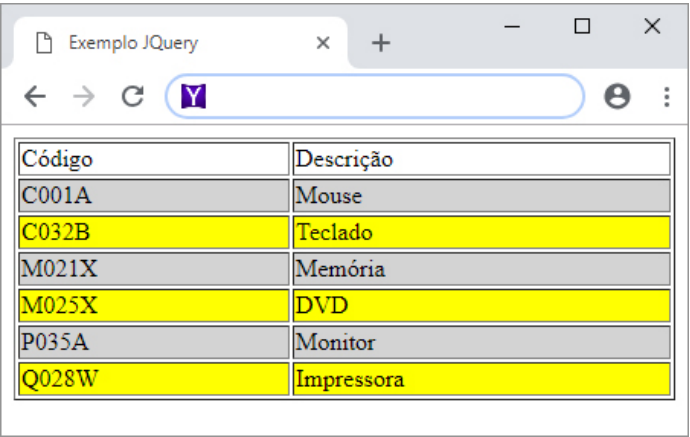
Saiba mais

Diversos outros exemplos de [seletores <https://www.w3schools.com/jquery/jquery_ref_selectors.asp>](https://www.w3schools.com/jquery/jquery_ref_selectors.asp) estão disponíveis no endereço.

Após selecionarmos os elementos que serão alvo de nossas ações, podemos utilizar as diversas propriedades do JQuery para efetuar mudanças no visual ou comportamento dos componentes da página.

Uma aplicação clássica dos seletores é a utilização de CSS para causar o efeito “zebrado” nos elementos de uma listagem qualquer, o que podemos observar no exemplo seguinte.


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
    <tittle>Exemplo JQuery</tittle>
    <script src="jquery.js">
    </script>
    <script>
function alterar() {
  $(".dados:odd").css("background-color", "yellow");
  $(".dados:even").css("background-color", "lightgray");
}
    </script>
  </head>
  <body onload="alterar()">
    <border="1" width="100%">
    <tr><td>Código</td><td>Descrição</td></tr>
    <tr class="dados" ><td>C001A</td><td>Mouse</td></tr>
    <tr class="dados" ><td>C032B</td><td>Teclado</td></tr>
    <tr class="dados" ><td>M021X</td><td>Memória</td></tr>
    <tr class="dados" ><td>M025X</td><td>DVD</td></tr>
    <tr class="dados" ><td>P035A</td><td>Monitor</td></tr>
    <tr class="dados" ><td>Q028W</td><td>Impressora</td></tr>
    </table>
  </body>
</html>
```



Neste exemplo chamamos a função alterar a partir do evento **onload** da página, fazendo com que, assim que a página é carregada, os elementos com a classe “**dados**” de índice ímpar fiquem com fundo amarelo, enquanto os de índice par assumam um fundo cinza.

Esta mudança da formatação é feita com o uso do método **css** do JQuery, o qual recebe como parâmetros a propriedade CSS e o valor da mesma.

```
$(".dados:odd").css("background-color", "yellow");
$(".dados:even").css("background-color", "lightgray");
```

Eventos no JQuery

Além de conseguirmos selecionar elementos da página e modificar suas características, podemos relacionar eventos a estes elementos a partir da sintaxe do JQuery.

Exemplo

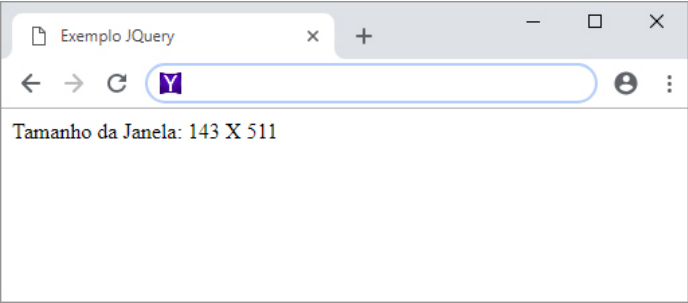
Alguns eventos estão direcionados para elementos globais. Por exemplo, ao invés de utilizar o evento **onload** e direcionar para uma função JavaScript, poderíamos utilizar algo como o código seguinte:

```
$(document).ready( function() {
  $("div, h1").css("color", "blue");
} );
```

Este fragmento de código indica que, ao carregar o documento, ou seja, evento **ready**, será executada a nossa função, fazendo com que todas as tags **<div>** e **<h1>** utilizem fonte na cor azul.

Além de eventos relacionados ao documento, temos alguns que trabalham com a janela, como **resize**. Vamos observar o exemplo seguinte.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
    <tittle>Exemplo JQuery</tittle>
    <script src="jquery.js">
    </script>
    <script>
function exibir(){
  $("#dimensao").html($(window).height() + " X " +
    $(window).width());
}
$(window).resize(function() {
exibir();
});
$(document).ready(function() {
exibir();
});
    </script>
  </head>
  <body>
    Tamanho da Janela: <span id="dimensao"/>
  </body>
</html>
```



Neste exemplo estamos apresentando as dimensões da janela ao acabar de carregar o documento e sempre que ela for redimensionada.

Como o processo é o mesmo em ambos os eventos, foi criada uma função chamada **exibir**, onde temos o código para atualizar o conteúdo do elemento ****, e esta função é chamada em ambos os eventos.

Para a mudança do conteúdo interno do elemento, utilizamos o método **html**; e para capturar a altura e largura da janela, utilizamos **height** e **width**.

Podemos observar no quadro seguinte alguns eventos de uso geral, relacionados à grande maioria dos seletores.

Evento	Utilização
click	Permite uma resposta ao clique do mouse.
dblclick	Permite uma resposta ao duplo-clique do mouse.
mouseenter	Ocorre quando o ponteiro do mouse está posicionado sobre o elemento.
mouseleave	Ocorre quando o ponteiro do mouse não está mais sobre o elemento.
toggle	Alterna entre cliques sucessivos.
contextmenu	Permite controle sobre a utilização do menu de contexto.
blur	Quando um elemento do tipo input perdeu o foco.
focus	Quando um elemento do tipo input recebeu o foco.
submit	Ocorre ao enviar o formulário.
change	Quando um elemento do tipo input tem seu valor alterado.
keydown	Quando uma tecla é pressionada.
keyup	Quando uma tecla é liberada.
keypress	Ação de “teclar”, envolvendo o pressionamento e liberação da tecla.

Um exemplo muito útil para o **contextmenu** é a proteção do conteúdo da página para ações que envolvam o clique com o botão direito do mouse.

```
$(document).contextmenu(function(){
  alert("Conteúdo não permite cópia");
  return false; // Impede o tratamento padrão
```

Podemos observar, no código seguinte, alguns dos eventos sendo utilizados.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
    <tittle>Exemplo JQuery</tittle>
    <script src="jquery.js">
    </script>
    <script>
      $(document).ready(function(){
        $("#Exemplo1").mouseenter(function(){
          $("#Exemplo1").css("color", "red");
        });
        $("#Exemplo1").mouseleave(function(){
          $("#Exemplo1").css("color", "black");
        });
        $("#Exemplo1").click(function(){
          alert("Apenas um Exemplo");
        });
      });
    </script>
  </head>
  <body>
    </button id="Exemplo1">APENAS UM EXEMPLO </button>
  </body>
</html>
```

Neste exemplo temos a utilização dos eventos **mouseenter** e **mouseleave** para controlar a cor da fonte de um botão, além do evento **click**, que mostra uma mensagem através de alert ao ocorrer o clique sobre este botão.

Atenção

É importante observarmos que os eventos foram programados dentro de um evento maior, que seria o **ready** do documento. Isto é feito para que a associação ocorra apenas quando os componentes tiverem sido criados, de forma a evitar qualquer inconsistência.

O interessante do uso de JQuery é que podemos intercalar elementos específicos da sintaxe com chamadas clássicas do JavaScript, o que traz grande poder para o ambiente, já que podemos combinar as mais diferentes bibliotecas JavaScript com as funcionalidades JQuery.

Outra forma de associar um evento é por delegação, uma prática interessante pois nos permite desligar os eventos a qualquer momento.

Para associar o evento de clique anterior utilizaríamos o método **on**; podemos desligá-lo com o uso de **off**.

```
$("#Exemplo1").on("click", function(){
  alert("Apenas um Exemplo");
  $("#Exemplo1").off("click");
});
```

Efetuando esta alteração, apenas o primeiro clique sobre o botão funcionaria, pois no código da function temos a utilização de **off** logo após a execução do **alert**.

JQuery UI

Através do módulo JQuery UI (User Interface) nós, desenvolvedores, podemos criar interfaces visuais extremamente atrativas com pouquíssimo esforço. Como todos os elementos JQuery, neste também trabalhamos com a aplicação de funções a seletores da página, e o resultado final é a mudança do aspecto original.



Para utilizar o JQuery UI, além de baixar o JQuery, temos que baixar a biblioteca JavaScript e um tema CSS específicos do módulo, os quais são fornecidos em forma de arquivo **zip**, disponível no endereço jqueryui.com <<https://jqueryui.com>>.

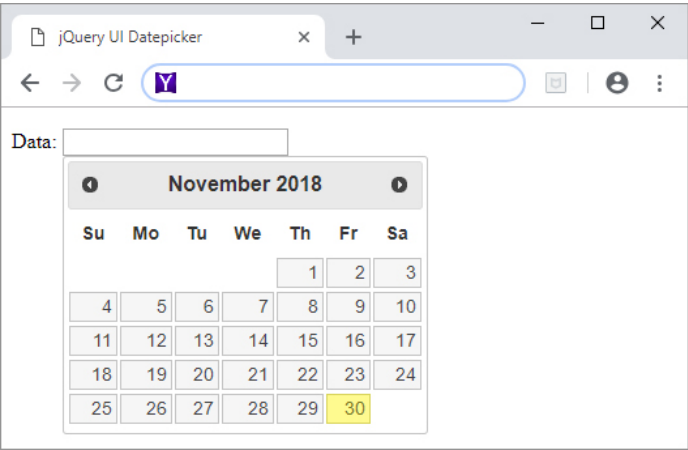
Após baixar, é só expandir e copiar o conjunto de arquivos e diretórios para o mesmo diretório de nossas páginas.

É muito fácil observarmos o poder do JQuery UI no exemplo seguinte.



Fonte: Aditya Chinchure / Unsplash

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <tittle>jQuery UI Datepicker</tittle>
    <link rel="stylesheet" href="jquery-ui.css">
    <script src="jquery.js"></script>
    <script src="jquery-ui.js"></script>
    <script>
      $( function() {
        $( "#data1" ).datepicker();
      } );
    </script>
  </head>
  <body>
    <p>Data: <input type="text" id="data1"></p>
  </body>
</html>
```



Note a simplicidade com que criamos este calendário com uso de JQuery UI.

Inicialmente incluímos as bibliotecas necessárias e arquivos CSS, algo que se repetirá em qualquer página criada com a utilização deste módulo.

```
<link rel="stylesheet" href="jquery-ui.css">
<script src="jquery.js"></script>
<script src="jquery-ui.js"></script>
```

Em seguida, efetuamos a transformação de uma tag **<input>** com identificador “**data1**” em calendário através do uso do método **datepicker**.

```
$( function() {
  $( "#data1" ).datepicker();
});
```

Hoje falamos bastante acerca de RIA (**Rich Internet Application**), ou interfaces ricas, que trata da utilização de comportamentos próximos ao desktop na criação de páginas Web, e o módulo JQuery UI traz uma grande quantidade de componentes que cumprem com esse padrão funcional.

Além do calendário, esse módulo traz opções como listas, planilhas, efeitos, diálogos, suporte às operações de arraste, além da possibilidade de utilização de temas personalizados com uso de CSS.

Alguns dos componentes JQuery UI podem ser observados no quadro seguinte.

	Evento	Descrição Geral
	accordion	Conteúdo disposto em seções retráteis.
	datepicker	Calendário com utilização personalizável.
	dialog	Diversos tipos de janelas de diálogo com possibilidade de arraste.
	menu	Criação de menus e submenus.
	progressbar	Barra de progresso, normalmente utilizada em downloads.
	slider	Seleção de valor por arraste, como no controle de volume ou brilho.
	tabs	Conteúdo sobreposto com abas de acesso.

Vamos observar agora um exemplo um pouco mais complexo, com o uso de alguns desses componentes.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <title>jQuery UI</title>
    <link rel="stylesheet" href="jquery-ui.css">
    <script src="jquery.js"></script>
    <script src="jquery-ui.js"></script>
    <script>
      $( function() {
        $("#menu1").menu();
        $("#dialogo1").dialog({
          autoOpen: false,
          show: { effect: "blind", duration: 1000
        },
          hide: { effect: "explode", duration: 1000
        }
        });
        $("#estacio").click(function(){
          location.href="//www.estacio.br";
        });
        $("#harvard").click(function(){
          location.href="//www.harvard.com/";
        });
        $("#abreDialogo").click(function() {
          $("#dialogo1").dialog("open");
        });
      });
    </script>
    <style>
      .ui-menu { width: 150px; }
    </style>
  </head>
  <body>
    <ul id="menu1">
      <li><div id="estacio">Estacio</div></li>
      <li><div id="harvard">Harvard</div></li>
      <li><div>Exemplos</div>
        <ul></ul>
        <li><div id="abreDialogo">Diálogo</div></li></li>
        <li><div>Nada Definido</div></li>
      </ul>
    </li>
    <li class="ui-state-disabled"><div>Desabilitado</div></li>
  </ul>
  <div id="dialogo1" title="Diálogo Simples"></div>
  <p>Apenas uma caixa de diálogo que permite arraste e pode ser fechada clicando no "X"</p>
</div>
</body>
</html>
```

Inicialmente, como podemos observar no exemplo, os menus são construídos com o uso de listas e divisões internas e configurados com o uso de **menu**. A parte visual é bem simples, mas devemos controlar o clique sobre os elementos constituintes.

```
$( "#estacio" ).click(function(){
    location.href="//www.estacio.br";
});
```

No clique sobre a opção de menu “Estácio”, será aberto o site da instituição através do atributo de navegador **location.href** do JavaScript.

O mesmo modelo serve para qualquer link, como para a opção que desvia para “Harvard”.

Quanto aos diálogos, a construção mais básica envolve apenas a apresentação de um texto, como uma janela de mensagem, mas permitindo, por padrão, o arraste e o fechamento da caixa de diálogo.

Para construirmos um diálogo, precisamos de uma tag **<div>**, e o conteúdo interno deverá ser colocado com a tag **<p>**. Observe o uso do atributo **title**.

```
<div id="dialogo1" title="Diálogo Simples">
<p>Apenas uma caixa de diálogo que permite arraste e pode ser fechada clicando no "X"</p>
</div>
```

Após definirmos o diálogo no HTML, devemos configurá-lo via JQuery.

```
$( "#dialogo1" ).dialog({
    autoOpen: false,
    show: { effect: "blind", duration: 1000
    },
    hide: { effect: "explode", duration: 1000
    }
});
```

Note que este diálogo adicionou uma série de configurações, como o uso de “**autoOpen: false**” para iniciar o diálogo invisível, e os efeitos para abertura (**show**) e fechamento (**hide**).

Ambos os efeitos utilizados terão duração de um segundo, sendo que a abertura utilizará **blind** e o fechamento **explode**.

Saiba mais

Existem diversos efeitos disponíveis, como: **fade, drop, bounce, puff, shake**, entre muitos outros. Podemos testá-los livremente com a substituição neste trecho do código.

Para abrir a caixa de diálogo já configurada, devemos utilizar o parâmetro “open”.

```
$( "#abreDialogo" ).click(function() {
    $( "#dialogo1" ).dialog("open");
});
```

Uma observação que deve ser feita é que existe a possibilidade de configurar algumas características via CSS, como a largura do menu.

```
<style>
.ui-menu { width: 150px; }
</style>
```


Esta utilização de CSS é muito importante, pois viabiliza a utilização e criação de temas, o que modifica de forma simples todo o visual da página criada.

Segundo o princípio de uso das folhas de estilo **em cascata**, o primeiro link é o padrão, para o arquivo `“jquery-ui.css”`, e depois podemos adicionar o link para o tema, já que qualquer elemento repetido será sobreposto ao inicial; caso não haja sobreposição será utilizado o valor original.

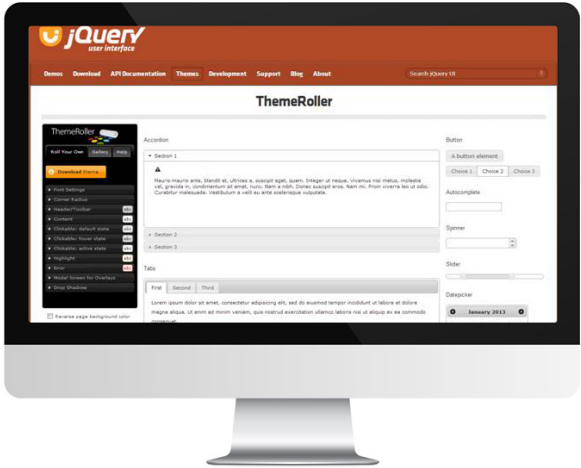
Podemos testar e baixar diversos temas de exemplo, além de criar nossos próprios temas, em <https://jqueryui.com/themeroller/> <<https://jqueryui.com/themeroller/>>

Saiba mais

Podemos testar e baixar diversos [temas <https://jqueryui.com/themeroller/>](https://jqueryui.com/themeroller/) de exemplo, além de criar nossos próprios temas.

Segundo o princípio de uso das folhas de estilo **em cascata**, o primeiro link é o padrão, para o arquivo `“jquery-ui.css”`, e depois podemos adicionar o link para o tema, já que qualquer elemento repetido será sobreposto ao inicial; caso não haja sobreposição será utilizado o valor original.

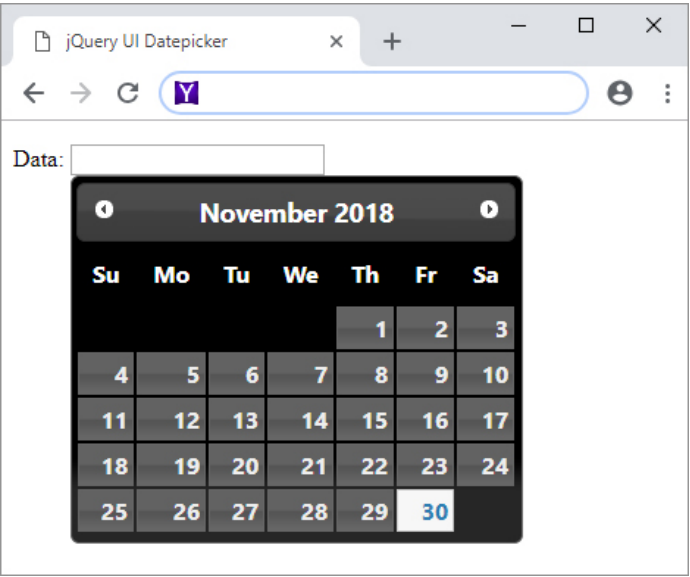
Experimente navegar pela galeria e baixar algum tema de seu interesse. Basta efetuar o download do arquivo **zip** e sobrepor os arquivos e diretórios anteriormente utilizados para uso da biblioteca JQuery UI, além de acrescentar o **link** para o tema.



Fonte: Gabriel Beaudry / Unsplash

```
<link rel="stylesheet" href="jquery-ui.css">
<link rel="stylesheet" href="jquery-ui.theme.css">
```

O exemplo de calendário inicial, com a aplicação do tema **UI Darkness**, passa a mostrar um visual bem diferenciado, como pode ser observado a seguir.



Com o uso de temas, podemos personalizar totalmente as características visuais da página, sem ter de alterar o código JavaScript, ou seja, mantendo toda a funcionalidade padrão enquanto a formatação é feita de forma externa.

Esse tipo de metodologia vai ao encontro dos princípios da W3C, pois promove a separação clara entre estrutura, funcionalidade e formatação.

Atividade

1. A sintaxe JSON é muito utilizada por Web Services REST e diversos frameworks do mercado, tratando de uma forma simples de representar objetos, e se baseia em elementos do tipo chave-valor. Ao definir o par chave-valor, qual tipo de dado NÃO pode ser representado diretamente como um atributo JSON?

- a) Texto
- b) Número
- c) Booleano
- d) Objeto
- e) Data

2. Você começou a utilizar JQuery e precisa acessar o primeiro item de uma lista, colocada diretamente na última camada da página. Qual seria o seletor correto?

```
<div>
  <ul>
    <li>A</li> <li>B</li>
  </ul>
</div>
<div>
  <ul>
    <li>C</li> <li>D</li>
  </ul>
</div>
```

- A)** \$("div div ul li ")
- B)** \$(".last div ul .first li")
- C)** \$("#last li #first")
- D)** \$("div:last ul li:first")
- E)** \$("li:last ul div")

3. Você começou a utilizar JQuery UI em um novo sistema, e deseja que o clique sobre um botão identificado como “Btn1” efetue a abertura de um diálogo identificado como “Dgn1”. Escreva o código necessário para esta ação.

Notas e Referências

DEITEL, P; DEITEL, H. **Ajax, rich internet applications e desenvolvimento web para programadores**. São Paulo: Pearson Education, 2009.

PLOTZE, R. **Tecnologias web**. Rio de Janeiro: Estácio, 2016.

SANTOS, F. **Tecnologias para internet II**. Vol. 1. Rio de Janeiro: Estácio, 2017.

Próxima aula

- As tecnologias XML;
- O funcionamento do AJAX;
- Utilização do DOM para tratamento de dados XML;

Explore mais

Não deixe de visitar as páginas abaixo:

- [Sintaxe JSON <https://www.w3schools.com/js/js_json_syntax.asp>](https://www.w3schools.com/js/js_json_syntax.asp)
- [Seletores do JQuery <https://api.jquery.com/category/selectors/>](https://api.jquery.com/category/selectors/)
- [Exemplos de JQuery UI <https://jqueryui.com/demos>](https://jqueryui.com/demos/)

