FIAP ON

RESPONSIVE WEB DEVELOPMENT

JQUERY

5

LISTA DE FIGURAS

Figura 5.1 – Exemplo de código jQuery no estilo “Alô mundo”.....8
Figura 5.2 – Exemplo de seletor de elemento e do tipo “id” 9
Figura 5.3 – Exemplo de seletor utilizando contains() e not().....11
Figura 5.4 – Exemplo de efeito visual utilizando hide() e show().....12
Figura 5.5 – Exemplo de efeito visual utilizando animate() 15
Figura 5.6 – Analogia a formulário 16
Figura 5.7 – Exemplo de dica de preenchimento de formulário 18
Figura 5.8 – Exemplo de manipulação de *checkboxes* 20
Figura 5.9 – Exemplo de requisição Ajax usando jQuery.....24
Figura 5.10 – Exemplo de uso do *plug-in TimeDropper* 27
Figura 5.11 – Exemplo de uso do *plug-in Knob*.....28
Figura 5.12 – Exemplo de criação de *plug-in* no jQuery.....29
Figura 5.13 – Exemplo de criação de *plug-in* no jQuery (2) 30
Figura 5.14 – Exemplo de criação de *plug-in* no jQuery (3) 31
Figura 5.15 – Exemplo de criação de *plug-in* no jQuery (4) 32
Figura 5.16 – Exemplo de criação de *plug-in* no jQuery (5) 35
Figura 5.17 – Exemplo de efeitos de Explode usando jQueryUI 39
Figura 5.18 – Exemplo de efeitos de *size* usando jQueryUI.....39
Figura 5.19 – Exemplo de redimensionamento usando jQueryUI 40
Figura 5.20 – Exemplo de redimensionamento e arrasto usando jQueryUI 41
Figura 5.21 – Exemplo de classificação de elementos usando jQueryUI.....43
Figura 5.22 – Exemplo de *widget* de acordeão usando jQueryUI 44
Figura 5.23 – Exemplo de *widget* de menu usando jQueryUI 45
Figura 5.24 – Exemplo de *widget* de datePicker usando jQueryUI 47

LISTA DE CÓDIGOS-FONTE

Código-fonte 5.1 – Exemplo de jQuery aplicado a um documento HTML.....7
Código-fonte 5.2 – Exemplo de código jQuery no estilo “Alô mundo” 7
Código-fonte 5.3 – Exemplo de seletor de elemento e do tipo “id”.....9
Código-fonte 5.4 – Exemplo de seletor utilizando constains() e not() 10
Código-fonte 5.5 – Exemplo de efeito visual utilizando hide() e show() 12
Código-fonte 5.6 – Exemplo de efeito visual utilizando slideToggle() 14
Código-fonte 5.7 – Exemplo de efeito visual utilizando animate().....15
Código-fonte 5.8 – Exemplo de dica de preenchimento de formulário.....17
Código-fonte 5.9 – Exemplo de manipulação de checkboxes.....19
Código-fonte 5.10 – Script dinâmico para o exemplo de Ajax com jQuery.....22
Código-fonte 5.11 – Exemplo de requisição Ajax usando jQuery.....23
Código-fonte 5.12 – Exemplo de requisição Ajax usando jQuery, método load().....25
Código-fonte 5.13 – Exemplo de uso do plug-in TimeDropper.....26
Código-fonte 5.14 – Exemplo de uso do plug-in Knob 28
Código-fonte 5.15 – Exemplo de criação de plug-in no jQuery 29
Código-fonte 5.16 – Exemplo de criação de plug-in no jQuery (2).....30
Código-fonte 5.17 – Exemplo de criação de plug-in no jQuery (3).....31
Código-fonte 5.18 – Exemplo de criação de plug-in no jQuery (4).....32
Código-fonte 5.19 – Exemplo de criação de plug-in no jQuery (5).....34
Código-fonte 5.20 – Exemplo de criação de plug-in no jQuery (6).....35
Código-fonte 5.21 – Exemplo de efeitos usando jQueryUI.....38
Código-fonte 5.22 – Exemplo de redimensionamento usando jQueryUI.....40
Código-fonte 5.23 – Exemplo de redimensionamento e arrasto usando jQueryUI....41
Código-fonte 5.24 – Exemplo de classificação de elementos usando jQueryUI.....42
Código-fonte 5.25 – Exemplo de widget de acordeão usando jQueryUI.....44
Código-fonte 5.26 – Exemplo de widget de menu usando jQueryUI.....45
Código-fonte 5.27 – Exemplo de widget de datePicker usando jQueryUI.....46

SUMÁRIO

5 JQUERY.....5
5.1 Considerações a respeito da versão.....6
5.2 Pré-requisito deste capítulo.....6
5.3 Como aplicar o jQquery em um projeto HTML.....6
5.4 Alô Mundo e o Atalho $.....7
5.5 A magia dos seletores.....8
5.6 Efeitos visuais práticos.....11
5.7 Facilidades em formulários.....15
5.7.1 Dica de preenchimento de campos.....16
5.7.2 Lidando com checkboxes.....18
5.8 Ajax mais fácil.....20
5.9 Plug-ins.....25
5.9.1 Timedropper.....25
5.9.2 Knob.....27
5.9.3 Crie seu próprio plug-in!.....28
5.10 JQueryUI: as vantagens do framework no frontend.....35
5.10.1 Baixe apenas o que você precisa.....36
5.10.2 Efeitos visuais.....36
5.10.3 Interações.....39
5.10.4 Widgets.....43
CONCLUSÃO.....48
REFERÊNCIAS.....49

5 JQUERY

JavaScript é uma linguagem indispensável para as soluções *web*, isso é inquestionável. No entanto, programadores *web* nutrem uma relação de “amor e ódio” com a linguagem.

No topo das reclamações, está a incompatibilidade entre os navegadores *web*: embora seja uma linguagem padronizada e mantida pela ECMA INTERNACIONAL, ao longo dos anos alguns fabricantes insistiram em criar suas próprias funções personalizadas, ou não oferecer o suporte completo à linguagem, gerando uma série de inconsistências. Você já deve ter ouvido falar que, para se ter certeza de que uma aplicação *web* realmente funciona, faz-se necessário testá-la em todos os navegadores do mercado (e em suas várias versões). Nada divertido.

Além disso, as instruções da linguagem não são muito enxutas e, por isso, muitas linhas de código são necessárias para implementar soluções para problemas simples. Como exemplo, temos o método estático `document.getElementById()`, que é um dos mais utilizados da linguagem. Todos concordam que é um comando grande demais para se escrever dezenas ou centenas de vezes.

Frameworks são estruturas de desenvolvimento muito comuns que, ao abstrair e agrupar instruções, têm o potencial de resolver esses e outros problemas. Usando *frameworks*, ganhamos produtividade: atingimos resultados maiores com uma menor quantidade de códigos.

O jQuery é um *framework* criado em dezembro de 2006 com o seguinte slogan: “*write less, do more*” (escreva menos, faça mais). Ao utilizá-lo, resolvemos a incompatibilidade entre os navegadores, reduzimos o tamanho do código e reutilizamos códigos por meio de *plug-ins* criados por milhares de programadores do mundo inteiro.

Segundo um levantamento de abril de 2016, feito pelo site W3techs.com, 70% de todos os sites monitorados usam jQuery, tornando-o o *framework* mais utilizado do mercado.

5.1 Considerações a respeito da versão

Atualmente, o jQuery é distribuído em duas versões maiores: 1.x e 2.x. A API das duas versões é a mesma, e a principal diferença é que as versões 2.x não suportam o Microsoft Internet Explorer na versão 8 ou inferiores. Concentraremos nossos estudos nas versões 2.x, pois utilizam versões mais recentes do JavaScript e, por isso, trarão mais performance. Se precisar de suporte a navegadores mais antigos, utilize a versão 1.x. Por serem mais básicos, a maioria dos exemplos apresentados aqui funcionará em ambas as versões.

5.2 Pré-requisito deste capítulo

O material a seguir assume a premissa de que o leitor tem noções de lógica de programação ou desenvolvimento de algoritmos e, além disso, conhece tudo que foi abordado sobre HTML, CSS e JavaScript em nossos materiais.

5.3 Como aplicar o jQquery em um projeto HTML

O primeiro passo é entrar no site oficial do projeto <http://www.jquery.com/> e clicar no botão de download. A versão mais indicada é a 2.x de produção, comprimida, feita dessa maneira para não “pesar” no projeto *web*: a versão atual tem menos de 90 kbytes.

O arquivo com o código-fonte do jQuery pode ser aplicado como qualquer outro arquivo jQuery, utilizando a tag `<script>`:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de jQuery</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
  </head>
  </body>
  </body>
</html>
```

Código-fonte 5.1 – Exemplo de jQuery aplicado a um documento HTML
Fonte: Elaborado pelo autor (2016)

Não utilize o arquivo *core* ou nativo para incluir os seus próprios códigos. Crie outros arquivos JavaScript, pois eles acessarão as API do jQuery normalmente. Mantendo o núcleo intacto, poderá atualizar com mais facilidade a versão do *framework*.

5.4 Alô Mundo e o Atalho $

O *framework* foca em produtividade, e já começa muito bem: em vez de utilizar o nome da classe padrão – jQuery –, sempre que é chamado, o *framework* dá a opção de se utilizar um atalho, o símbolo "$" (cifrão). Sendo assim, sempre que avistarmos um cifrão no código, o *framework* está sendo acionado.

Veja o exemplo:

ALOMUNDO.HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de jQuery</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
  </head>
  <body>
  </body>
</html>
```

ALOMUNDO.JS

```
$(document).ready(function(){
    console.log("Alô mundo!");
});
```

Código-fonte 5.2 – Exemplo de código jQuery no estilo "Alô mundo"
Fonte: Elaborado pelo autor (2016)

Identificamos rapidamente que se trata de um código jQuery graças ao cifrão; a palavra "`document`" entre parênteses é um seletor, a partir do qual podemos acessar o documento como um todo; `ready()` é um dos métodos disponíveis: a linha é o equivalente exato de `window.onload = function()` no JavaScript tradicional, sendo assim, `ready()` só será acionado quando todo o documento HTML estiver devidamente renderizado e pronto para uso. O `function()` dentro de

`ready()` acompanha o JavaScript clássico, trata-se de uma declaração anônima, declarada e chamada no mesmo ponto do código. O resultado é simples:

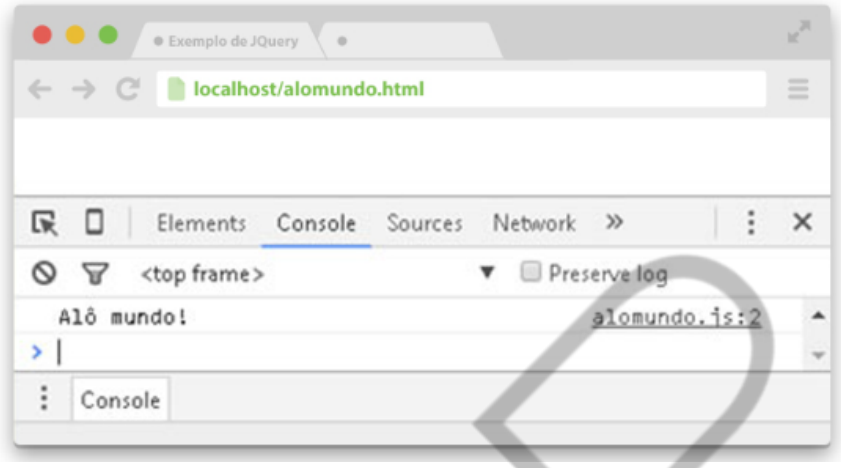

Figura 5.1 – Exemplo de código jQuery no estilo “Alô mundo”
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

5.5 A magia dos seletores

Como mencionado, o par de parênteses logo após o cifrão é um seletor. Lembra-se dos seletores do CSS? Pois então, o *framework* torna a programação em JavaScript tão simples quanto aplicar uma folha de estilo. Todos os seletores do padrão CSS 2.2 e outros estão disponíveis no jQuery.

Vamos ao primeiro exemplo: podemos aplicar um estilo CSS, criando um estilo (*class*) e aplicando-o por meio do método `addClass()`. O estilo transformará em vermelho todos os parágrafos escritos em cor preta, acionado por um clique de botão. O evento a ser aplicado no botão é, surpreendentemente, `click()`:

SELETOR1.HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de jQuery</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="seletor1.js"></script>
    <link rel="stylesheet" type="text/css"
href="seletor1.css">
  </head>
  <body>
    <p>Parágrafo 1</p>
    <p>Parágrafo 2</p>
```

```
          <p>Parágrafo 3</p>
          <input type="button" id="botao" value="Aplicar">
        </body>
      </html>

      SELETOR1.CSS
      .vermelho {
          color: red;
      }

      SELETOR1.JS
      $(document).ready(function() {
        //Implementa o evento de click no elemento cujo id é
"botao".
        $("#botao").click(function() {
          //Aplica a folha de estilo "vermelho" em todos os
elementos <p>
          $("p").addClass("vermelho");
        });
      });
```

Código-fonte 5.3 – Exemplo de seletor de elemento e do tipo “id”
Fonte: Elaborado pelo autor (2016)

Repare que os seletores são rigorosamente os mesmos utilizados em CSS. E o resultado é gerado com poucas linhas de código:

Figura 5.2 – Exemplo de seletor de elemento e do tipo “id”
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

Para se ter uma ideia do “poder de fogo” dos seletores específicos do jQuery, o exemplo a seguir utiliza o seletor `elemento:contains('termo’)`, a partir do qual conseguimos aplicar um estilo (novamente `addClass()`) em um elemento do HTML que contenha uma palavra ou trecho de frase. O exemplo é enriquecido com o seletor `elemento:not()`, que permite negar ou inverter a lógica de outros seletores (como o próprio *contains*):

SELETOR2.HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de jQuery</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="seletor2.js"></script>
    <link        rel="stylesheet"        type="text/css"
href="seletor2.css">
  </head>
  <body>
    <div>azul</div>
    <div>outra cor</div>
    <div>outra cor</div>
    <input type="button" id="botao" value="Aplicar">
  </body>
</html>
```

SELETOR2.CSS

```
.azul {
    color: blue;
}
.naoAzul {
    color: red;
}
```

SELETOR2.JS

```
$(document).ready(function(){
  //Implementa o evento de click no elemento cujo
  //id é "botao".
  $("#botao").click(function() {
    //Aplica a folha de estilo "azul" em todas as
    // divs que contiverem a palavra "azul"
    $("div:contains('azul')").addClass("azul");
    //Aplica a folha de estilo "naoAzul" em todas as
    // divs que NÃO contiverem a palavra "azul"

$("div:not(:contains('azul'))").addClass("naoAzul");
  });
});
```

Código-fonte 5.4 – Exemplo de seletor utilizando constains() e not()
Fonte: Elaborado pelo autor (2016)

E temos como resultado:

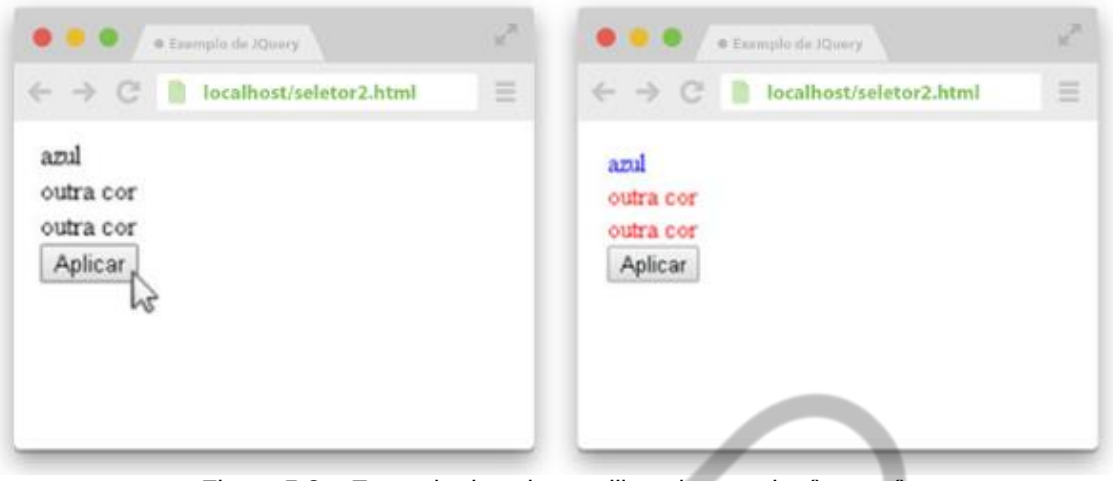

Figura 5.3 – Exemplo de seletor utilizando contains() e not()
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

5.6 Efeitos visuais práticos

Um dos grandes ganhos do jQuery é facilitar uma série de efeitos visuais corriqueiros. Comecemos pelos métodos `hide()` e `show()`, respectivamente: sumir com um elemento e mostrar. O exemplo a seguir possui uma área que chamaremos de "alvo" e, quando o mouse passar sobre ela (evento `mouseover`), outro elemento que chamaremos de "quadrado" surgirá na tela (assim que o documento for renderizado, vamos fazê-lo sumir). Por outro lado, quando o mouse sair do elemento "alvo" (evento `mouseout`), o elemento "quadrado" sumirá novamente. Veja o código-fonte:

EFEITOSVISUAIS1.HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de jQuery</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="efeitosvisuais1.js"></script>
    <link rel="stylesheet" type="text/css"
href="efeitosvisuais1.css">
  </head>
  <body>
    <div id="alvo">Passe o mouse</div>
    <div id="quadrado">Surpresa!</div>
  </body>
</html>
```

EFEITOSVISUAIS1.CSS

```
#alvo {
    border: 2px solid black;
    width: 200px;
    height: 40px;
    text-align: center; padding-top: 15px;
}
#quadrado {
    border: 2px solid black;
    width: 200px;
    height: 100px;
    text-align: center; padding-top: 50px;
}
```

EFEITOSVISUAIS1.JS

```
$(document).ready(function() {
  //quadrado começa invisível
  $("#quadrado").hide();
  //quando o mouse passar pelo alvo, mostra quadrado
  $("#alvo").mouseover(function(){
    $("#quadrado").show();
  });
  //quando o mouse sair do alvo, some com o quadrado
  $("#alvo").mouseout(function(){
    $("#quadrado").hide();
  });
});
```

Código-fonte 5.5 – Exemplo de efeito visual utilizando hide() e show()
Fonte: Elaborado pelo autor (2016)

Veja o exemplo funcionando:

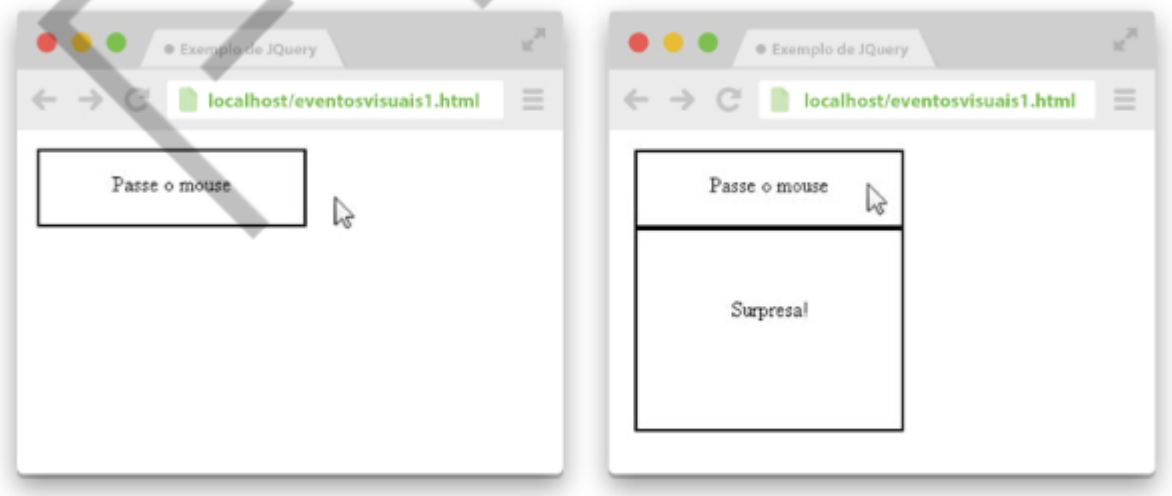

Figura 5.4 – Exemplo de efeito visual utilizando hide() e show()
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

No entanto, o efeito é instantâneo e, portanto, muito “brusco”. Para dar um efeito de “deslizar”, tornando-o uma transição animada, podemos utilizar métodos como `slideDown()` (deslizar para baixo), `slideUp()` (deslizar para cima) e `slideToggle()` (deslizar para baixo e para cima, com alternância).

O método `slideToggle()` possui três parâmetros opcionais: o primeiro determina a velocidade da animação; o segundo é o tipo (se ela é linear ou não); e o terceiro é uma função de *callback* a ser utilizada no final da animação. Veja o exemplo reescrito com `slideToggle()`:

EFEITOSVISUAIS2.HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de jQuery</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="efeitosvisuais2.js"></script>
    <link rel="stylesheet" type="text/css"
href="efeitosvisuais2.css">
  </head>
  <body>
    <div id="alvo">Passe o mouse</div>
    <div id="quadrado">Surpresa!</div>
  </body>
</html>
```

EFEITOSVISUAIS2.CSS

```
#alvo {
    border: 2px solid black;
    width: 200px;
    height: 40px;
    text-align: center; padding-top: 15px;
}
#quadrado {
    border: 2px solid black;
    width: 200px;
    height: 100px;
    text-align: center; padding-top: 50px;
}
```

EFEITOSVISUAIS2.JS

```
$(document).ready(function() {
  //quadrado começa invisível
  $("#quadrado").hide();
  //quando o mouse passar pelo alvo, mostra quadrado
  $("#alvo").mouseover(function(){
    $("#quadrado").slideToggle("slow");
```

```
        });
        //quando o mouse sair do alvo, some com o quadrado
        $("#alvo").mouseout(function(){
          $("#quadrado").slideToggle("slow");
        });
      });
```

Código-fonte 5.6 – Exemplo de efeito visual utilizando slideToggle()
Fonte: Elaborado pelo autor (2016)

Outro método interessante é o `animate()`, que permite algumas alterações em certas propriedades de estilos CSS (não em todos) de forma animada, dando um efeito de transição de "morph", transformação. O exemplo que demonstra isso é um quadrado vermelho de 100 pixels por 100 pixels, que dobra de tamanho quando o mouse passa por ele. Segue o código-fonte:

EFEITOSVISUAIS3.HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de jQuery</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="efeitosvisuais3.js"></script>
    <link rel="stylesheet" type="text/css"
href="efeitosvisuais3.css">
  </head>
  <body>
    <div id="quadrado"></div>
  </body>
</html>
```

EFEITOSVISUAIS3.CSS

```
#quadrado
{
    border: 2px solid black;
    background-color: red;
    width: 100px;
    height: 100px;
    text-align: center;
}
```

EFEITOSVISUAIS3.JS

```
$(document).ready(function() {
  $("#quadrado").mouseover(function(){
    $("#quadrado").animate({width: "200px",
                            height: "200px"});
  });
```

```
        });
```

Código-fonte 5.7 – Exemplo de efeito visual utilizando animate()
Fonte: Elaborado pelo autor (2016)

Veja o exemplo funcionando:

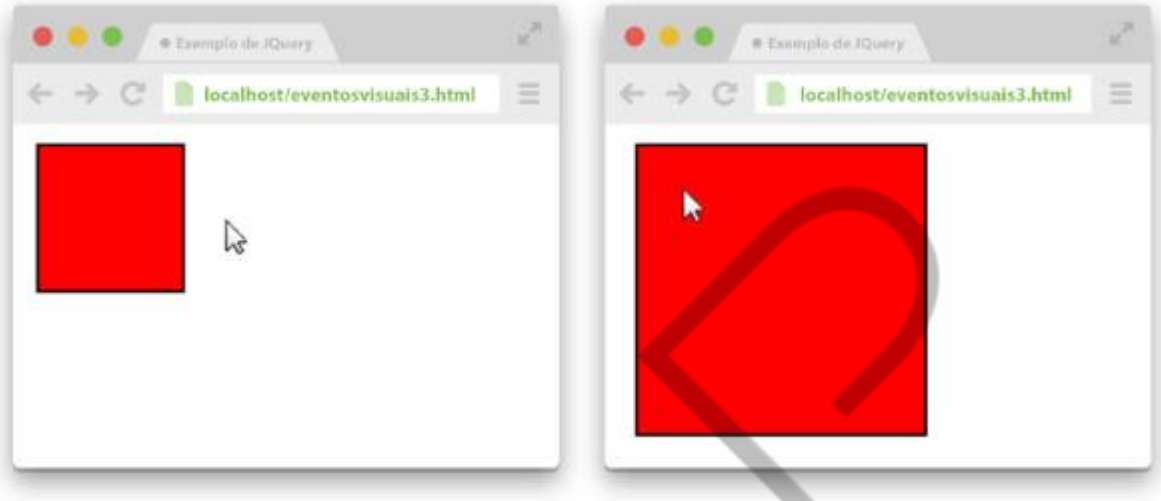

Figura 5.5 – Exemplo de efeito visual utilizando animate()
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

5.7 Facilidades em formulários

Uma das coisas repetitivas e, portanto, CHATAS com que um programador de JavaScript precisa lidar é a manipulação de formulários: personalizá-los e validá-los são ações absolutamente necessárias para uma boa interface com o usuário.

Muitos avanços já ocorreram no HTML5, como o uso de *placeholders*, validação de campos obrigatórios, números, entre outras novidades. No entanto, o jQuery pode ser útil ao preencher outras lacunas.

Figura 5.6 – Analogia a formulário
Fonte: FIAP (2017)

5.7.1 Dica de preenchimento de campos

Trata-se de um exemplo simples de se fazer, que utiliza os eventos hide() e show(); ele também ajuda a mostrar dois recursos importantes: a palavra **this**, que no JavaScript representará o objeto do DOM que disparou o evento e a possibilidade de se "navegar" de um elemento para outro, utilizando o método `next()`, que chama o elemento seguinte do DOM. Ambos serão muito úteis para se generalizar o exemplo, permitindo aplicar a dica em vários campos ao mesmo tempo.

Neste exemplo, teremos duas caixas de texto e um botão; aplicaremos o evento `blur`, que é disparado quando o elemento em questão perde o foco. Se o usuário deixar o campo e ele estiver vazio, será exibida a dica obrigando-o a preenchê-lo. O exemplo começa com as dicas todas ocultas, e quando o usuário voltar ao campo para preenchê-lo, faremos a dica sumir com o evento `focus`:

FORM1.HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de jQuery</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="form1.js"></script>
```

```
        </head>
        <body>
          <form>
            <label for="email">E-mail: </label>
            <input type="text" id="email">
            <span>Preenchimento de e-mail é
obrigatório!</span><br>
            <label for="senha">Senha: </label>
            <input type="text" id="senha">
            <span>Preenchimento de senha é
obrigatório!</span><br>
            <input type="submit" value="Autenticar">
          </form>
        </body>
      </html>

      FORM1.JS
      $(document).ready(function() {
        // Sumir com todos os span's
        $("span").hide();
        // Aplica evento de blur (perda de foco) em todos os
inputs
        // do tipo "type"
        $("input[type='text']").blur(function(){
          // this representa o objeto que disparou o evento,
no
         // caso, a caixa de texto que o usuário acabou de
sair
          // (seja qual da duas for!)
          if ($(this).val().length == 0) {
            //  Exibe o elemento seguinte (o span logo
depois!)
            $(this).next().show();
          }
        });
        // Aplicado no ganho de foco de todas as caixas de
texto,
        //  some com o span imediatamente após (Caso sua
mensagem
        // esteja sendo exibida!)
        $("input[type='text']").focus(function() {
          $(this).next().hide();
        });
      });
```

Código-fonte 5.8 – Exemplo de dica de preenchimento de formulário
Fonte: Elaborado pelo autor (2016)

Veja-o em funcionamento:

Figura 5.7 – Exemplo de dica de preenchimento de formulário
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

5.7.2 Lidando com checkboxes

Uma das coisas mais chatas de se manipular em formulários HTML são os *checkboxes*: a necessidade de verificar seus valores, marcá-los ou desmarcá-los, entre outras ações. O exemplo a seguir mostra quanto o *framework* facilita esse trabalho: os *checkboxes* de interesse, como “música”, “cinema” e “teatro”, exibirão seus conteúdos no console, se clicados. O *checkbox* “marcar todos” possuirá uma verificação: se ele estiver marcado (elemento. `.is(':checked') == true`), chamará o evento de clique de todos os outros *checkboxes*, utilizando um método conhecido como `trigger()`. Caso ele seja desmarcado, todos os outros campos também o serão, utilizando um método chamado de `prop()`.

FORM2.HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de jQuery</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="form2.js"></script>
  </head>
  <body>
    <form>
      <input type="checkbox" id="todos">Marcar
todos<br><br>
      <span>Interesses: </span><br>
```

```
            <input      type="checkbox"      name="interesses"
value="Música">Música<br>
            <input      type="checkbox"      name="interesses"
value="Cinema">Cinema<br>
            <input      type="checkbox"      name="interesses"
value="Teatro">Teatro<br>
          </form>
        </body>
      </html>
```

FORM2.JS

```
$(document).ready(function() {
  $("#todos").click(function() {
    // Se o checkbox todos estiver marcado.
    if ($("#todos").is(':checked')) {
      // Este comando marcaria todos os checkboxes

//$("input[name='interesses']").prop('checked',true);
      // mas o ideal é disparar um evento de clique em
cada
      // um deles e acionar a programação do evento.
      $("input[name='interesses']").trigger("click");
    } else {
      // Se o checkbox todos estiver desmarcado.
      // Desmarca todos os outros checkboxes

$("input[name='interesses']").prop('checked',false);
    }
  });
  // Quando um dos checkboxes de interesse é clicado,
  // exibe o conteúdo no console.
  $("input[name='interesses']").click(function(){
    console.log($(this).val());
  });
});
```

Código-fonte 5.9 – Exemplo de manipulação de checkboxes
Fonte: Elaborado pelo autor (2016)

E seu funcionamento, após poucas linhas de código:

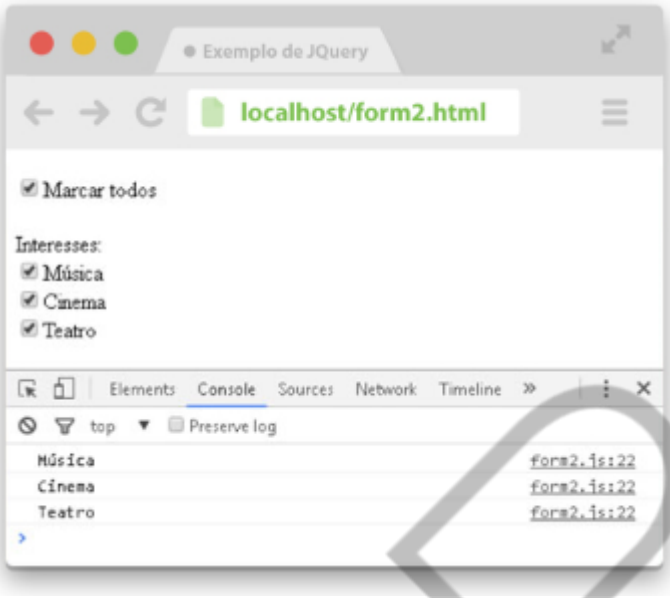

Figura 5.8 – Exemplo de manipulação de *checkboxes*
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

5.8 Ajax mais fácil

Uma das coisas que o jQuery facilitou muito são as requisições Ajax. É fato que as novas versões de JavaScript facilitaram muito as requisições (com a classe `xmlHttpRequest` na versão 2.0), mas o *framework* já melhorava a vida de muitos programadores em tempos mais sombrios. Além disso, o *framework* cria uma pilha de requisições, útil quando muitas chamadas Ajax são realizadas quase simultaneamente – em um código simples demais, os retornos das chamadas se "atropelam" e são perdidos.

Embora existam vários métodos para esse fim, focaremos no JQuery.ajax() (ou $.ajax()) e suas possibilidades. O método recebe como parâmetro duas informações: a URL que será chamada e, em seguida, as configurações dessa chamada.

Essas configurações são opcionais e passadas em formato JSON, seguem abaixo as principais possibilidades:

- **contentType:** configura o tipo de conteúdo que será enviado. Como é padrão no protocolo HTTP, "application/x-www-form-urlencoded; charset=UTF-8" é o que será usado quando nada for configurado, e as

possibilidades são “multipart/form-data” e “text/plain”. A mudança cabe nos casos de envio de arquivos.

- **method:** possibilita configurar o método em que a requisição HTTP será realizada. Como é padrão do protocolo, “GET” será utilizado, e temos como opções “POST”, “PUT” e “DELETE”.
- **data:** permite passar quais informações serão transportadas na requisição, são os dados de entrada. Podem ser enviados por PlainObject, String ou um Array.
- **success:** é um evento disparado em caso de sucesso, ou seja, a requisição Ajax foi ao servidor e voltou com sucesso. Definimos aqui os procedimentos que serão executados nesse caso.
- **error:** é um evento disparado em caso de fracasso, ou seja, a requisição Ajax foi ao servidor e, por alguma razão, retornou um erro. Definimos aqui os procedimentos que serão executados nesse caso.
- **timeout:** tempo em milissegundos que o navegador ficará esperando por um retorno da chamada Ajax.
- **username e password:** podem ser informados em uma eventual autenticação HTTP.

Vamos a um exemplo, para facilitar o entendimento. Optamos por um exemplo mais dinâmico: um *script* PHP que, ao receber a sigla de um estado, analisa e retorna o nome do estado por extenso. Para não tornar o exemplo muito grande, faremos isso apenas com estados da Região Sudeste (se você for de outro estado, não fique ofendido, é apenas um problema de tamanho de código).

Para que o exemplo funcione, faz-se necessária a instalação de um software do tipo WAMP (pacote contendo Apache Web Server, MySQL e PHP para Windows). Segue:

AJAX1_SERVER.PHP

```
<?php
// Pega a informação 'sigla' da requisição HTTP POST.
// Transforma em letras maiúsculas com strtoupper()
$sigla = (isset($_POST['sigla']) ? strtoupper($_POST['sigla']) : "");
// Analisa a sigla e devolve o estado por extenso.
```

```
      // A cláusula 'default' devolve exceções.. siglas
inválidas // fora da região sudeste não são realizadas (o
exemplo
      // ficaria muito grande!)
      switch($sigla) {
        case "SP":
          echo "São Paulo";
        break;
        case "RJ":
          echo "Rio de Janeiro";
        break;
        case "MG":
          echo "Minas Gerais";
        break;
        case "ES":
          echo "Espírito Santo";
        break;
        default:
          echo "Não é um estado do sudeste.";
      }
      ?>
```

Código-fonte 5.10 – Script dinâmico para o exemplo de Ajax com jQuery
Fonte: Elaborado pelo autor (2016)

O lado cliente conterá uma caixa de texto para receber os dois caracteres da sigla do estado; esse exemplo não possuirá um botão para disparar a requisição, fará uso de um evento conhecido como `keyUp`, que é disparado quando algo é digitado e a tecla digitada está subindo, após seu acionamento. Se a caixa em questão contiver os dois caracteres, a requisição Ajax com o conteúdo da caixa será enviada e, caso retorne do *script* PHP com sucesso, pegará o nome por extenso do estado e exibirá no `<div>` de resultado presente no exemplo. Veja:

```
      AJAX1.HTML
      <!DOCTYPE html>
      <html lang="pt-br">
        <head>
          <title>Exemplo de JQuery</title>
          <meta charset="utf-8">
          <script src="jquery-2.min.js"></script>
          <script src="ajax1.js"></script>
        </head>
        <body>
          <label for="sigla">Digite a sigla do estado
:</label>
          <input type="text" id="sigla" style="width: 30px"
maxlength="2">
          <br><br>
          <div id="resultado"></div>
```

```
    </body>
  </html>

  AJAX1.JS
  $(document).ready(function() {
    // Evento keyup ao digitar na caixa de texto.
    $("#sigla").keyup(function() {
      // Se o tamanho do conteúdo da caixa for igual a 2
      // (sigla digitada completamente)
      if($("#sigla").val().length == 2) {
        $.ajax({
          url: "ajax1_server.php",
          method: "POST",
          // enviar a sigla
          data: "sigla="+$("#sigla").val(),
          // Se a requisição voltar com sucesso.
          success: function(result){
            $("#resultado").html(result);
          },
          // Se a requisição NÃO voltar com sucesso.
          error: function(){
            $("#resultado").html("Houve uma falha na
requisição.");
          }
        });
      } else {
        $("#resultado").html("");
      }
    });
  });
```

Código-fonte 5.11 – Exemplo de requisição Ajax usando jQuery
Fonte: Elaborado pelo autor (2016)

E ao executar e informar siglas válidas e inválidas:

Figura 5.9 – Exemplo de requisição Ajax usando jQuery
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

A outra possibilidade é o método load(): se a requisição Ajax pode ser mais simples, ou seja, não precisa contar com controles como procedimentos em caso de fracasso, ela pode ser muito útil.

Considere a mesma situação do exemplo anterior; o mesmo *script* PHP e a mesma necessidade. Veja como ela seria escrita, usando o método `load()`:

AJAX2.HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de jQuery</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="ajax2.js"></script>
  </head>
  <body>
    <label for="sigla">Digite a sigla do estado
:</label>
```

```
          <input type="text" id="sigla" style="width: 30px"
maxlength="2">
          <br><br>
          <div id="resultado"></div>
        </body>
      </html>

      AJAX2.JS
      $(document).ready(function() {
        // Evento keyup ao digitar na caixa de texto.
        $("#sigla").keyup(function() {
          // Se o tamanho do conteúdo da caixa for igual a 2
          // (sigla digitada completamente)
          if($("#sigla").val().length == 2) {
            $("#resultado").load("ajax1_server.php",        {
"sigla": $("#sigla").val() });
          } else {
            $("#resultado").html("");
          }
        });
      });
```

Código-fonte 5.12 – Exemplo de requisição Ajax usando jQuery, método load()
Fonte: Elaborado pelo autor (2016)

Mais simples e com o mesmo resultado do primeiro exemplo.

5.9 Plug-ins

Uma das coisas mais interessantes de um *framework* é sua capacidade de poder ser expandido, ganhando novas funcionalidades que vão além das concebidas originalmente por seus criadores. Essas soluções são empacotadas e distribuídas com o nome de *plug-ins*.

Existem muitos *plug-ins* diferentes, criados por toda a comunidade de desenvolvedores. Vejamos a seguir alguns exemplos de uso e até mesmo como criar seu próprio *plug-in*.

5.9.1 Timedropper

Vamos começar os exemplos de *plug-ins* com o *TimeDropper*, que é um *plug-in* muito simples de se implementar. Sua proposta é uma interface diferente para o

preenchimento de campos em que se deseja informar um intervalo de tempo usando horas e minutos.

O primeiro passo é baixar seu código fonte em <http://felicegattuso.com/projects/timedropper/>. Dentro do pacote em zip, dois arquivos nos interessam: `timedropper.min.js` (código JavaScript do *plug-in*) e `timedropper.min.css` (estilo necessário) – esses serão copiados para nossa pasta de projeto.

O código-fonte para colocá-lo em funcionamento é bem simples:

TIMEDROPPER.HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de plug-in em JQuery</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="timedropper.js"></script>
    <!-- Arquivos do plug-in -->
    <script src="timedropper.min.js"></script>
    <link rel="stylesheet" href="timedropper.min.css">
  </head>
  <body>
    <form>
      <label for="tempo">Tempo Gasto: </label>
      <input   type="text"   id="tempo"   style="width:
50px">
    </form>
  </body>
</html>
```

TIMEDROPPER.JS

```
$(document).ready(function() {
  //  Chamada  do  plug-in.  A  opção  "format"  permite
configurar
  // a hora para o formato 0-23 horas.
  $( "#tempo" ).timeDropper({format: 'H:mm'});
});
```

Código-fonte 5.13 – Exemplo de uso do plug-in TimeDropper
Fonte: Elaborado pelo autor (2016)

Simples, não é? Toda a complexidade foi encapsulada dentro do *plug-in* e seu CSS correspondente. Caso esteja curioso e queira dar uma olhada, veja-o funcionando:

Figura 5.10 – Exemplo de uso do *plug-in TimeDropper*
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

5.9.2 Knob

Outro exemplo de *plug-in* é o Knob: que cria um botão de dial com um visual atraente, manipulável pelo mouse ou eventos de toque, sem o uso de imagens.

Para fazê-lo funcionar, é necessário baixar o arquivo `jquery.knob.js`, em <https://github.com/aterrien/jQuery-Knob/>, e armazená-lo na pasta de projeto. Veja um exemplo de uso:

KNOB.HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de plug-in em JQuery</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="knob.js"></script>
    <!-- Arquivos do plug-in -->
    <script src="jquery.knob.js"></script>
  </head>
  <body>
    <form>
      <input type="text" value="50" id="dial">
    </form>
  </body>
</html>
```

KNOB.JS

```
$(document).ready(function() {
  $("#dial").knob({
```

```
            'min':0,
            'max':100,
            'lineCap': 'round'
        });
    });
```

Código-fonte 5.14 – Exemplo de uso do plug-in Knob
Fonte: Elaborado pelo autor (2016)

Também bem simples. Esta é a grande vantagem de um *plug-in*: criam-se métodos que são aplicáveis em elementos da mesma forma que qualquer método nativo do *framework*. Veja-o funcionando:

Figura 5.11 – Exemplo de uso do *plug-in Knob*
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

5.9.3 Crie seu próprio plug-in!

Uma das coisas mais interessantes que podemos fazer usando jQuery é criar um *plug-in*. Mas por que você faria isso? Ora, para encapsular toda a complexidade de um procedimento que será usado dezenas de vezes, ou seja, simplificamos o código e promovemos o reúso de código, inclusive entre projetos diferentes – componentizar o procedimento!

Parece algo complexo, mas na verdade é extremamente simples. Vamos a um exemplo:

```
JQUERY_MEUPLUGIN.HTML
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de plug-in em JQuery</title>
```

```
        <meta charset="utf-8">
        <script src="jquery-2.min.js"></script>
        <script src="jquery_meuplugin.js"></script>
      </head>
      <body>
        <form>
          <label for="nome">Nome: </label>
          <input type="text" id="nome">
        </form>
      </body>
    </html>

    JQUERY_MEUPLUGIN.JS
    //Declaração do plug-in
    $.fn.meuPlugin = function() {
      console.log("Eureka!");
    };
    $(document).ready(function() {
      //Chamada do plug-in
      $("#nome").meuPlugin();
    });
```

Código-fonte 5.15 – Exemplo de criação de plug-in no jQuery
Fonte: Elaborado pelo autor (2016)

O objeto jQuery (ou $, como estamos acostumados a simplificar) recebe seus métodos do objeto `$.fn`. Além de seus próprios métodos, ele permite que criemos métodos novos, como fizemos ao declarar “meuPlugin”.

Veja-o funcionando:

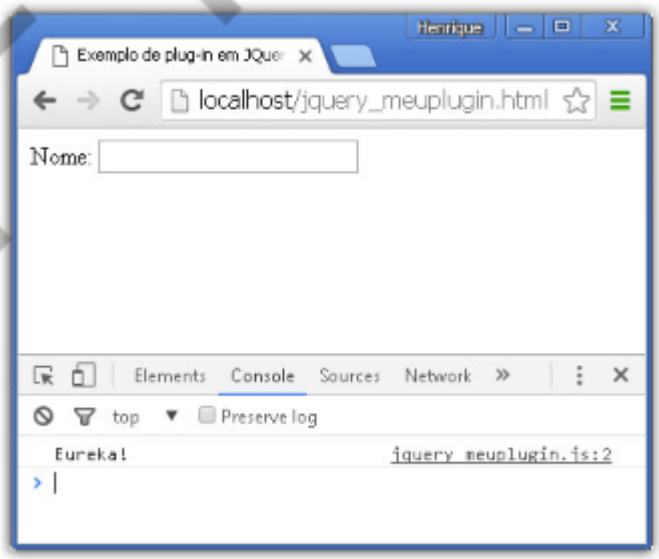

Figura 5.12 – Exemplo de criação de *plug-in* no jQuery
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

Bem legal, não é? Certo, ele não estava fazendo nada muito interessante, era apenas uma prova de conceito. Vamos evoluir o exemplo, `meuPlugin` pintará o fundo dos elementos na cor amarela:

JQUERY_MEUPLUGIN2.HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de plug-in em JQuery</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="jquery_meuplugin2.js"></script>
  </head>
  <body>
    <form>
      <label for="nome">Nome: </label>
      <input type="text" id="nome">
    </form>
  </body>
</html>
```

JQUERY_MEUPLUGIN2.JS

```
//Declaração do plug-in
$.fn.meuPlugin = function() {
  //Plug-in pinta o fundo do elemento de amarelo
  this.css({ backgroundColor: 'yellow' });
};
$(document).ready(function() {

  //Chamada do plug-in
  $("#nome").meuPlugin();
});
```

Código-fonte 5.16 – Exemplo de criação de plug-in no jQuery (2)
Fonte: Elaborado pelo autor (2016)

Uma mudança simples, mas que produz uma grande diferença:

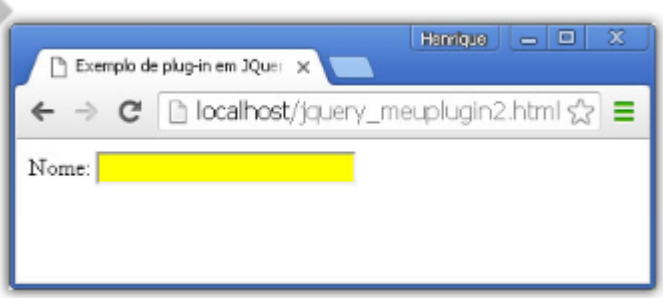

Figura 5.13 – Exemplo de criação de *plug-in* no jQuery (2)
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

Perceba que não importa quantos elementos temos no documento, nem qual seletor é usado:

JQUERY_MEUPLUGIN3.HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de plug-in em JQuery</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="jquery_meuplugin3.js"></script>
  </head>
  <body>
    <form>
      <label for="nome">Nome: </label>
      <input type="text" id="nome"> <br>
      <label for="idade">Idade: </label>
      <input type="number" id="idade">
    </form>
  </body>
</html>
```

JQUERY_MEUPLUGIN3.JS

```
$.fn.meuPlugin = function() {
  //Plug-in pinta o fundo do elemento de amarelo
  this.css({ backgroundColor: 'yellow' });
};

$(document).ready(function() {
  //Chamada do plug-in
  $("input").meuPlugin();
});
```

Código-fonte 5.17 – Exemplo de criação de plug-in no jQuery (3)
Fonte: Elaborado pelo autor (2016)

Será tão eficaz quanto:

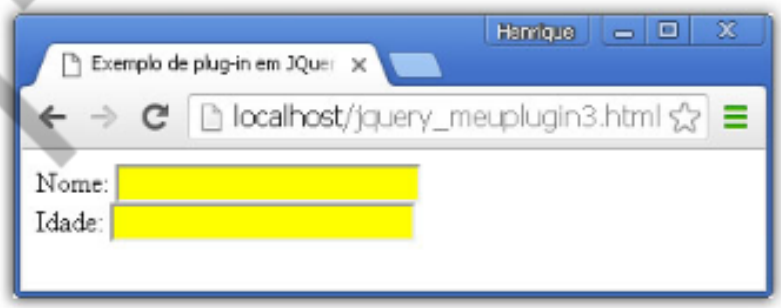

Figura 5.14 – Exemplo de criação de *plug-in* no jQuery (3)
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

Mas vamos dar um passo adiante, tornando o *plug-in* mais útil. E se ao chamá-lo, o desenvolvedor pudesse colocar sua cor preferida? Caso ele não informe nenhuma, continuamos pintando de amarelo:

JQUERY_MEUPLUGIN4.HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de plug-in em JQuery</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="jquery_meuplugin4.js"></script>
  </head>
  <body>
    <form>
      <label for="nome">Nome: </label>
      <input type="text" id="nome"> <br>
      <label for="idade">Idade: </label>
      <input type="number" id="idade">
    </form>
  </body>
</html>
```

JQUERY_MEUPLUGIN4.JS

```
$.fn.meuPlugin = function(options) {
  //Plug-in espera por parametrização da cor a ser
  //pintada. Caso não seja definida, pinta de amarelo
  if (typeof options === "undefined") {
    var options = {};
    options.corDeFundo = "yellow";
  } else {
    if (typeof options.corDeFundo === "undefined") {
      options.corDeFundo = "yellow";
    }
  }

  //pinta com a cor definida no objeto "options"
  this.css({ backgroundColor: options.corDeFundo });
};
$(document).ready(function() {
  //Chamada do plug-in passando parâmetro
  $("input").meuPlugin({corDeFundo: "red"});
});
```

Código-fonte 5.18 – Exemplo de criação de plug-in no jQuery (4)
Fonte: Elaborado pelo autor (2016)

Veja como o *plug-in* se torna mais útil:

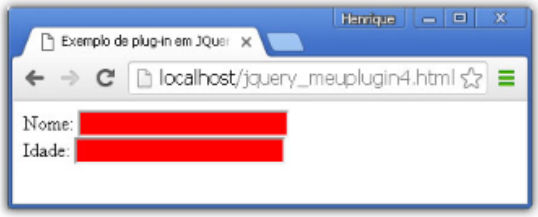

Figura 5.15 – Exemplo de criação de *plug-in* no jQuery (4)
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

Se nenhuma parametrização for feita na chamada do *plug-in*, ele pinta de amarelo. No entanto, se outros parâmetros forem criados depois, essa estrutura condicional crescerá consideravelmente.

O código-fonte pode ser refeito por meio do método `$.extend`: o método pega as propriedades do objeto padrão (*defaults*) e as mescla com as propriedades do objeto `options` passado como parâmetro, dando preferência a este último, se foi realmente criado:

JQUERY_MEUPLUGIN5.HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de plug-in em JQuery</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="jquery_meuplugin5.js"></script>
  </head>
  <body>
    <form>
      <label for="nome">Nome: </label>
      <input type="text" id="nome"><br>
      <label for="idade">Idade: </label>
      <input type="number" id="idade">
    </form>
  </body>
</html>
```

JQUERY_MEUPLUGIN5.JS

```
$.fn.meuPlugin = function(options) {
  //Plug-in espera por parametrização da cor a ser
  //pintada. Caso não seja definida, pinta de amarelo
  //VERSÃO MELHORADA, trata outros parâmetros também.
  var defaults = {
    'corDeFundo' : 'yellow'
  };
  var settings = $.extend( {}, defaults, options );
  this.css({ backgroundColor: settings.corDeFundo });
};
$(document).ready(function() {
  //Chamada do plug-in passando parâmetro
  $("input").meuPlugin({corDeFundo: "red"});
});

$(document).ready(function() {
  //Chamada do plug-in passando parâmetro
  $("input").meuPlugin({corDeFundo: "red"});
});
```

```
        //pinta com a cor definida no objeto "options"
        this.css({ backgroundColor: options.corDeFundo });
      };
      $(document).ready(function() {
        //Chamada do plug-in passando parâmetro
        $("input").meuPlugin({corDeFundo: "red"});
      });
```

Código-fonte 5.19 – Exemplo de criação de plug-in no jQuery (5)
Fonte: Elaborado pelo autor (2016)

O resultado final pode ser o mesmo, mas é uma solução mais curta, elegante e eficaz.

A última melhoria permite o cascateamento de métodos, tão utilizada em jQuery. Se quisermos esconder os elementos logo após a aplicação do *plug-in*, a linha será `$("input").meuPlugin().hide();`, certo? Pois bem, essa linha não funcionaria. Precisamos incluir um `return this;`, que permitirá que a linha funcione:

```
      JQUERY_MEUPLUGIN6.HTML
      <!DOCTYPE html>
      <html lang="pt-br">
        <head>
          <title>Exemplo de plug-in em JQuery</title>
          <meta charset="utf-8">
          <script src="jquery-2.min.js"></script>
          <script src="jquery_meuplugin6.js"></script>
        </head>
        <body>
          <form>
            <label for="nome">Nome: </label>
            <input type="text" id="nome"><br>
            <label for="idade">Idade: </label>
            <input type="number" id="idade">
          </form>
        </body>
      </html>

      JQUERY_MEUPLUGIN6.JS
      $.fn.meuPlugin = function(options) {
        //Tratamento de parâmetros
        var defaults = {  'corDeFundo' : 'yellow'  };
        var settings = $.extend( {}, defaults, options );
        this.css({ backgroundColor: settings.corDeFundo });
        //Resolve a questão do cascateamento de chamadas de
método.
        return this;
      };
      $(document).ready(function() {
```

```
        //Chamada do plug-in passando parâmetro
        //com métodos em cascata
        $("input").meuPlugin({corDeFundo: "red"}).hide();
    });
```

Código-fonte 5.20 – Exemplo de criação de plug-in no jQuery (6)
Fonte: Elaborado pelo autor (2016)

Veja o exemplo em funcionamento:

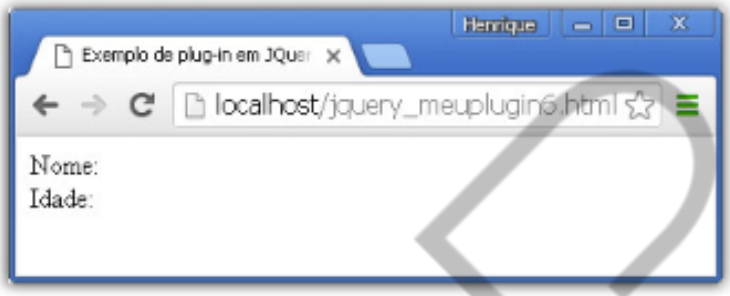

Figura 5.16 – Exemplo de criação de *plug-in* no jQuery (5)
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

É um exemplo simples, é verdade, mas possui todos os conceitos básicos necessários. Certamente você fará bom uso dos conceitos explicados aqui, criando *plug-ins* realmente úteis aos seus projetos.

5.10 JQueryUI: as vantagens do framework no frontend

A vida de um desenvolvedor de *frontend* HTML também não é fácil: existem inúmeros recursos e novas possibilidades de interação com o usuário. Criar animações e efeitos visuais utilizando JavaScript pode não ser uma tarefa muito divertida, especialmente quando o desenvolvedor não tem à disposição ferramentas que tragam produtividade a essa tarefa.

O jQueryUI é uma dessas ferramentas: uma biblioteca criada por cima do jQuery focada em interações com a interface de usuário, *widgets* e efeitos visuais. Com seu uso, o desenvolvedor pode conceber mais facilmente interfaces interativas e animadas que impressionem.

Essa poderosa ferramenta pode ser dividida em três módulos:

- **Efeitos visuais:** simples animações de fácil aplicação e parametrização, que permitem animar um elemento HTML com efeitos como explosão, deslizar, sumir (*fade*), entre outros.

- **Interações:** eventos que permitem maior interatividade com a interface, como a possibilidade de arrastar e soltar elementos, redimensioná-los ou ordená-los, por exemplo.
- ***Widgets*:** interfaces de usuário já construídas e totalmente customizáveis.

5.10.1 Baixe apenas o que você precisa

Por se tratar de muitos efeitos, interações, *widgets* e temas, a biblioteca é consideravelmente maior que o *framework* jQuery, além de precisar dele para funcionar. Por essa razão, é possível baixar os componentes de forma fragmentada, permitindo ao desenvolvedor baixar apenas os elementos que deseja; é uma abordagem extremamente recomendada, pois evita que a biblioteca sobrecarregue a interface, prejudicando a performance.

5.10.2 Efeitos visuais

São inúmeros os efeitos disponíveis no jQueryUI. No exemplo a seguir, eles foram aplicados a uma caixa de diálogo:

```
      JQUERYUI_EFEITOS.HTML
      <!DOCTYPE html>
      <html lang="pt-br">
        <head>
          <title>Exemplo de efeitos com jQuery UI</title>
          <meta charset="utf-8">
          <script src="jquery-2.min.js"></script>
          <script src="jquery-ui/jquery-ui.min.js"></script>
          <script src="jqueryui_efeitos.js"></script>
          <link rel="stylesheet" href="jquery-ui/jquery-
ui.theme.min.css">
          <link rel="stylesheet" href="jquery-ui/jquery-
ui.structure.min.css">
          <link rel="stylesheet" href="jqueryui_efeitos.css">
        </head>
        <body>
      <div class="toggler">
            <div id="efeito" class="ui-widget-content ui-
corner-all">
              <h3 class="ui-widget-header ui-corner-
```

```
all">Efeito</h3>
            <p>Vários exemplos de efeitos que podem ser
feitos com o jQuery UI. Experimente!</p>
          </div>
        </div>
        <input type="radio" name="efeito" value="blind">
Blind
        <input type="radio" name="efeito" value="bounce">
Bounce
        <input type="radio" name="efeito" value="clip">
Clip
        <input type="radio" name="efeito" value="drop">
Drop
        <input type="radio" name="efeito" value="explode">
Explode
        <input type="radio" name="efeito" value="fade">
Fade
        <br>
        <input type="radio" name="efeito" value="fold">
Fold
        <input type="radio" name="efeito"
value="highlight"> Highlight
        <input type="radio" name="efeito" value="puff">
Puff
        <input type="radio" name="efeito" value="pulsate">
Pulsate
        <input type="radio" name="efeito" value="scale">
Scale
        <input type="radio" name="efeito" value="shake">
Shake
        <br>
        <input type="radio" name="efeito" value="size">
Size
        <input type="radio" name="efeito" value="slide">
Slide
        <input type="radio" name="efeito" id="transfer"
value="transfer"> Transfer
      </body>
    </html>
```

JQUERYUI_EFEITOS.CSS

```
#efeito {
  width: 200px;
  height: 110px;
  padding: 0.4em;
  position: relative;
}
#efeito h3 {
  margin: 0;
  padding: 0.4em;
  text-align: center;
```

```
      }
      .toggler {
        width: 500px;
        height: 150px;
        position: relative;
      }
      .ui-effects-transfer {
        border: 2px dotted gray;
      }

      JQUERYUI_EFEITOS.JS
      $(document).ready(function() {
        function rodarEfeito(nmEfeito) {
          var options = {};
          switch(nmEfeito) {
            case "scale":
              options = { percent: 0 };
            break;
            case "size":
              options = { to: { width: 150, height: 30 } };
            break;
            case "transfer":
              options = { to: "#transfer", className: "ui-
effects-transfer" };
            break;
          }
          $( "#efeito" ).effect( nmEfeito, options, 500,
callback );
         }
          // método de efeito do jQueryUI
          $( "#efeito" ).effect( nmEfeito, options, 500,
callback );
        }

        // ao clica em qualquer radio, envia a escolha para
efeito
        $("input[type='radio']").click(function() {
          rodarEfeito($(this).val());
        });
        // função de callback para trazer a caixa de volta
        function callback() {
          setTimeout(function() {
            $( "#efeito" ).removeAttr( "style"
).hide().fadeIn();
          }, 1000 );
        };
      });
```

Código-fonte 5.21 – Exemplo de efeitos usando jQueryUI
Fonte: Elaborado pelo autor (2016)

Veja alguns dos exemplos funcionando:

Figura 5.17 – Exemplo de efeitos de Explode usando jQueryUI
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

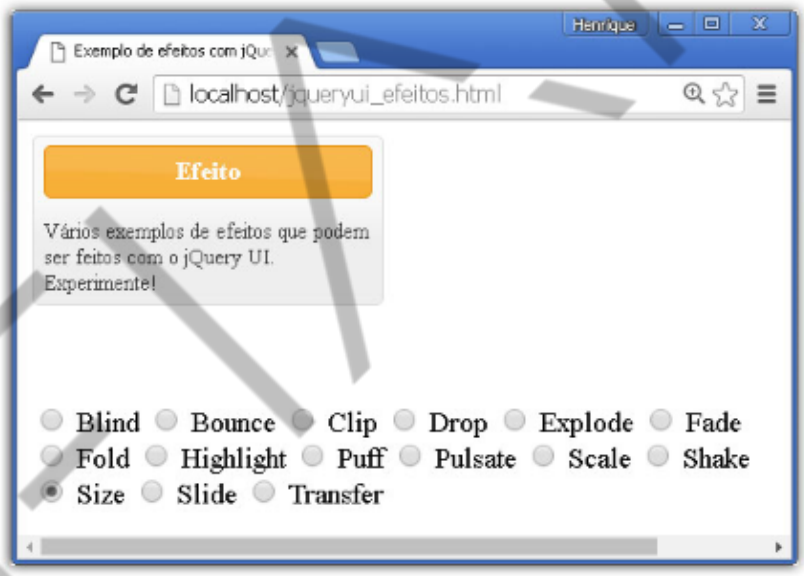

Figura 5.18 – Exemplo de efeitos de *size* usando jQueryUI
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

5.10.3 Interações

A biblioteca permite, sem muito esforço, criar interações com os elementos HTML, como arrastar e soltar, redimensionar, selecionar ou ordenar.

Veja no exemplo a seguir como é fácil tornar uma `<div>` redimensionável:

```
JQUERYUI_RESIZE.HTML
<!DOCTYPE html>
<html lang="pt-br">
```

```
        <head>
          <title>Exemplo de resize com jQuery UI</title>
          <meta charset="utf-8">
          <script src="jquery-2.min.js"></script>
          <script src="jquery-ui/jquery-ui.min.js"></script>
          <script src="jqueryui_efeitos.js"></script>
          <link    rel="stylesheet"    href="jquery-ui/jquery-
ui.theme.min.css">
      <link      rel="stylesheet"      href="jquery-ui/jquery-
ui.structure.min.css">
          <link rel="stylesheet" href="jqueryui_efeitos.css">
        </head>
        <body>
          <div id="quadrado"></div>
        </body>
      </html>

      JQUERYUI_RESIZE.CSS
      #quadrado {
        width: 120px;
        height: 100px;
        border: 2px solid black;
        background-color: #DDD;
      }

      JQUERYUI_RESIZE.JS
      $(document).ready(function() {
        $("#quadrado").resizable();
      });
```

Código-fonte 5.22 – Exemplo de redimensionamento usando jQueryUI
Fonte: Elaborado pelo autor (2016)

Fácil e prático:

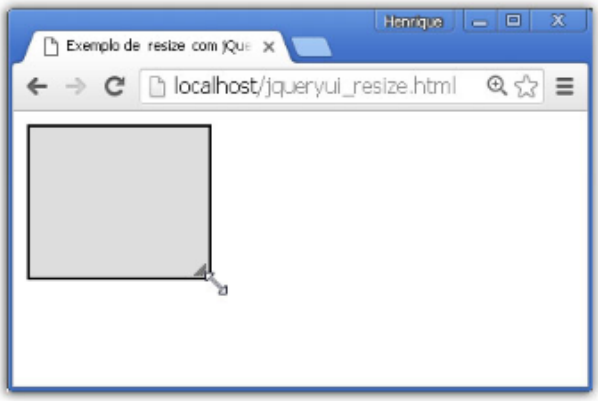

Figura 5.19 – Exemplo de redimensionamento usando jQueryUI
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

Graças ao CSS contido na biblioteca, as linhas diagonais no canto inferior direito são automaticamente adicionadas, representando a possibilidade de redimensionamento do elemento.

Várias interações podem ser aplicadas em um único elemento. A adição do método `draggable()` concede a possibilidade de arrastá-lo:

```
JQUERYUI_RESIZE.JS (MODIFICADO)
$(document).ready(function() {
  $("#quadrado").resizable().draggable();
});
```

Código-fonte 5.23 – Exemplo de redimensionamento e arrasto usando jQueryUI
Fonte: Elaborado pelo autor (2016)

Arrastável e dimensionável:

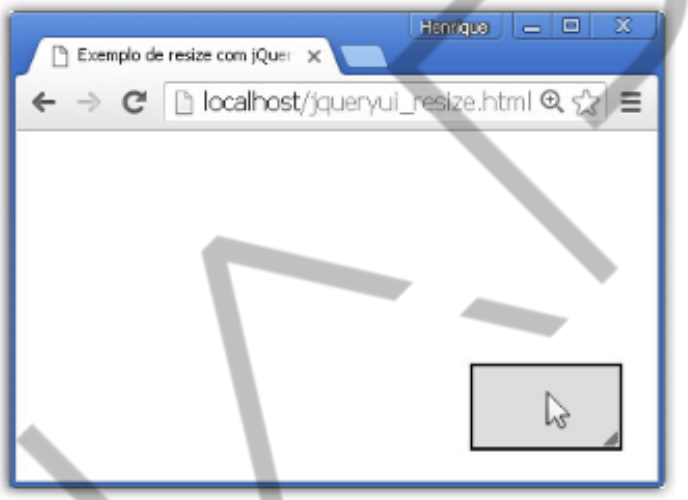

Figura 5.20 – Exemplo de redimensionamento e arrasto usando jQueryUI
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

A possibilidade de criar listas em que o usuário possa classificar os elementos é bem interessante. O exemplo a seguir demonstra como essa necessidade pode ser facilmente resolvida com o jQueryUI:

```
JQUERYUI_SORTABLE.HTML
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de classificação de elementos com
jQuery UI</title>
<meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="jquery-ui/jquery-ui.min.js"></script>
    <script src="jqueryui_sortable.js"></script>
    <link  rel="stylesheet"  href="jquery-ui/jquery-
ui.theme.min.css">
<link  rel="stylesheet"  href="jquery-ui/jquery-
ui.structure.min.css">
    <link                          rel="stylesheet"
href="jqueryui_sortable.css">
```

```
        </head>
        <body>
          <span>Melhores filmes de todos os tempos (do melhor
para os ... "menos melhores")</span>
          <ul id="melhores_filmes">
            <li>Menina de Ouro</li>
            <li>Poderoso Chefão</li>
            <li>A Lista de Schindler</li>
            <li>Um sonho de Liberdade</li>
          </ul>
        </body>
      </html>

      JQUERYUI_SORTABLE.CSS
      #melhores_filmes li {
        width: 200px;
        height: 20px;
        border: 2px solid #555;
        background-color: #888;
        color: #FFF;
        border-radius:3px;
        list-style-type: none;
        padding: 5px;
        margin: 2px;
      }
      #melhores_filmes li.placeholder {
          border: 2px none;
          background-color: #DDD;
      }

      JQUERYUI_RESIZE.JS
      $(document).ready(function() {
        $('#melhores_filmes').sortable({
          placeholder: 'placeholder'
        });
      });
```

Código-fonte 5.24 – Exemplo de classificação de elementos usando jQueryUI
Fonte: Elaborado pelo autor (2016)

Veja o exemplo funcionando:

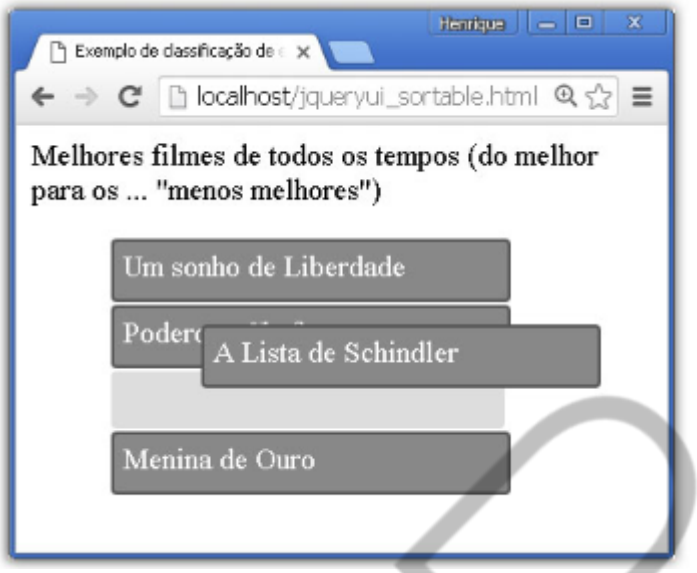

Figura 5.21 – Exemplo de classificação de elementos usando jQueryUI
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

5.10.4 Widgets

Várias são as ferramentas que trazem uma grande facilidade na navegação e na usabilidade das interfaces. Veremos a seguir alguns exemplos.

O primeiro deles é o acordeão: grandes quantidades de textos podem ser agrupadas e removidas da tela, mostrando apenas quando for conveniente, por meio desse mecanismo que lembra o instrumento musical.

A seguir, um exemplo de uso:

```
JQUERYUI_ACCORDION.HTML
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de widget do jQuery UI</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="jquery-ui/jquery-ui.min.js"></script>
    <script src="jqueryui_accordion.js"></script>
    <link    rel="stylesheet"    href="jquery-ui/jquery-
ui.theme.min.css">
    <link    rel="stylesheet"    href="jquery-ui/jquery-
ui.structure.min.css">
  </head>
  <body>
    <div id="accordion">
      <h3>Seção A</h3>
      <div>Primeira seção</div>
```

```
            <h3>Seção B</h3>
            <div>Segunda seção</div>
            <h3>Seção C</h3>
            <div>Terceira seção</div>
          </div>
        </body>
      </html>

      JQUERYUI_ACCORDION.JS
      $(document).ready(function() {
        $('#accordion').accordion();
      });
```

Código-fonte 5.25 – Exemplo de widget de acordeão usando jQueryUI
Fonte: Elaborado pelo autor (2016)

Veja o exemplo funcionando:

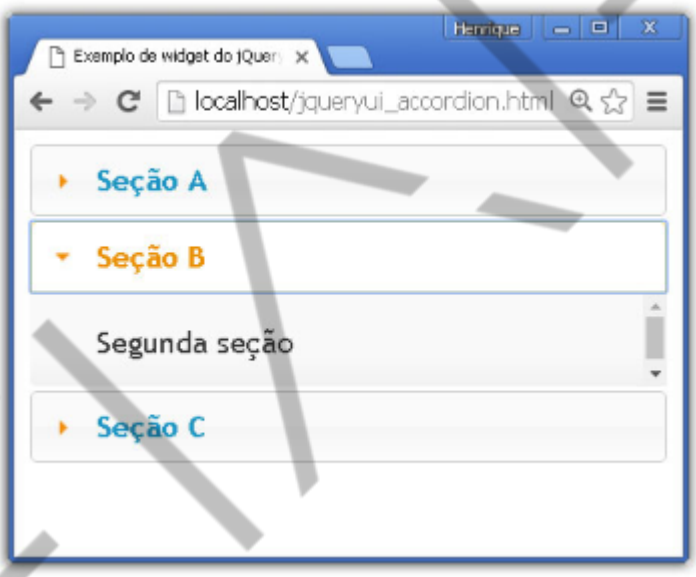

Figura 5.22 – Exemplo de *widget* de acordeão usando jQueryUI
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

A criação de menu também ficou mais prática:

```
      JQUERYUI_MENU.HTML
      <!DOCTYPE html>
      <html lang="pt-br">
        <head>
          <title>Exemplo de widget do jQuery UI</title>
          <meta charset="utf-8">
          <script src="jquery-2.min.js"></script>
          <script src="jquery-ui/jquery-ui.min.js"></script>
          <script src="jqueryui_menu.js"></script>
          <link    rel="stylesheet"    href="jquery-ui/jquery-
ui.theme.min.css">
          <link    rel="stylesheet"    href="jquery-ui/jquery-
ui.structure.min.css">
          <link rel="stylesheet" href="jqueryui_menu.css">
```

```
    </head>
    <body>
      <ul id="menu">
        <li>ABBA</li>
        <li class="ui-state-disabled">Alan Parsons</li>
        <li>AC/DC</li>
        <li>A-ha</li>
        <ul>
          <li>Hunting high and low</li>
          <li>Take on me</li>
          <li>Stay on these roads</li>
        </ul>
        <li>Audioslave</li>
      </ul>
    </body>
  </html>

  JQUERYUI_MENU.CSS
  .ui-menu { width: 180px; }

  JQUERYUI_MENU.JS
  $(document).ready(function() {
    $('#menu').menu();
  });
```

Código-fonte 5.26 – Exemplo de widget de menu usando jQueryUI
Fonte: Elaborado pelo autor (2016)

Simples, não é? Veja como funciona:

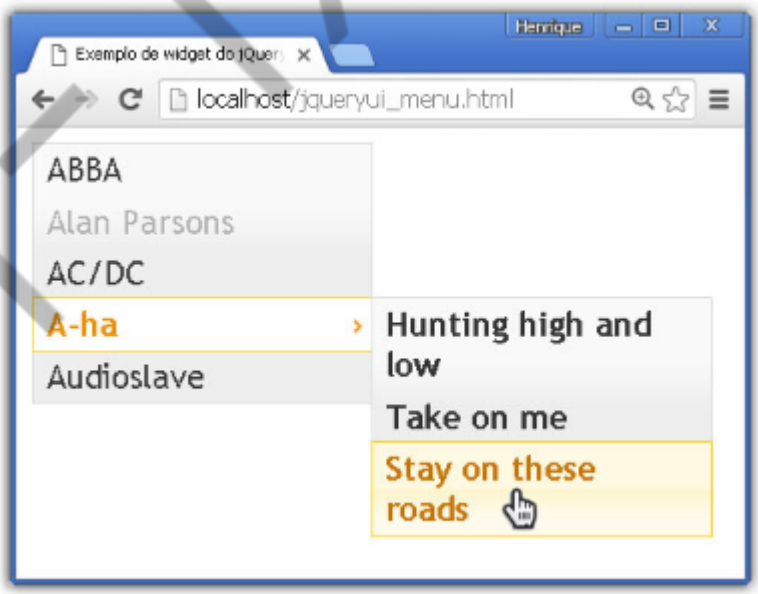

Figura 5.23 – Exemplo de *widget* de menu usando jQueryUI
Fonte: Elaborado pelo autor (2016), adaptado por FIAP (2017)

Vários *widgets*, como o *Datepicker*, já são suportados pelo HTML5 e tendem a se tornar desnecessários no futuro. No entanto, enquanto o suporte dos navegadores for deficitário, podem ser muito úteis:

JQUERYUI_DATEPICKER.HTML

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de widget do jQuery UI</title>
    <meta charset="utf-8">
    <script src="jquery-2.min.js"></script>
    <script src="jquery-ui/jquery-ui.min.js"></script>
    <script src="jqueryui_datepicker.js"></script>
    <link    rel="stylesheet"    href="jquery-ui/jquery-
ui.theme.min.css">
    <link    rel="stylesheet"    href="jquery-ui/jquery-
ui.structure.min.css">
  </head>
  <body>
    <label for="data">Data: </label>
    <input type="text" id="data">
  </body>
</html>
```

JQUERYUI_DATEPICKER.JS

```
$(document).ready(function() {
  $("#data").datepicker({
    dateFormat: 'dd/mm/yy',
    dayNames:
['Domingo','Segunda','Terça','Quarta','Quinta','Sexta','Sábado
'],
    dayNamesMin: ['D','S','T','Q','Q','S','S','D'],
    dayNamesShort:
['Dom','Seg','Ter','Qua','Qui','Sex','Sáb','Dom'],
    monthNames:
['Janeiro','Fevereiro','Março','Abril','Maio','Junho','Julho',
'Agosto','Setembro','Outubro','Novembro','Dezembro'],
    monthNamesShort:
['Jan','Fev','Mar','Abr','Mai','Jun','Jul','Ago','Set','Out','
Nov','Dez'],
    nextText: 'Próximo',
    prevText: 'Anterior'
  });
});
```

Código-fonte 5.27 – Exemplo de widget de datePicker usando jQueryUI
Fonte: Elaborado pelo autor (2016)

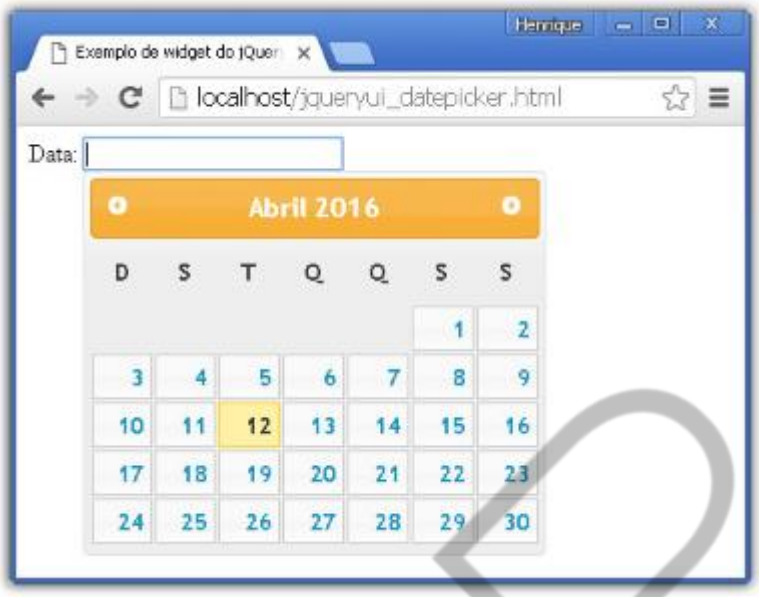

Figura 5.24 – Exemplo de *widget* de datePicker usando jQueryUI
Fonte: Elaborado pelo autor (2016)

CONCLUSÃO

Por tudo o que vimos aqui (e por muito mais, caso queira se aprofundar no assunto), é compreensível o fato de o jQuery estar presente em 70% dos *websites*: o *framework* torna o desenvolvimento em JavaScript muito mais fácil e produtivo.

Existem, é claro, outras excelentes opções, como o AngularJS, promovido pelo Google. Sendo assim, podemos concluir que o melhor *framework* é aquele que atende às suas necessidades.

REFERÊNCIAS

JQUERY. **jQuery API.** Disponível em: <http://api.jquery.com/>. Acesso em: 15 abr. 2016.

_____. **How to Create a Basic Plugin.** Disponível em: <https://learn.jquery.com/plugins/basic-plugin-creation/>. Acesso em: 15 abr. 2016.

JQUERYUI. **jQuery UI.** Disponível em: <http://www.jqueryui.com/>. Acesso em: 15 abr. 2016.

W3TECH.COM. **Usage of JavaScript libraries for websites.** Disponível em: <http://w3techs.com/technologies/overview/javascript_library/all>. Acesso em: 15 abr. 2016.