MÓDULO

JAVASCRIPT/AJAX

UNIDADE

CONCEITOS BÁSICOS DE AJAX, JSON, JQUERY

Instituto Tecnológico

ÍNDICE

OBJETIVOS

Com esta unidade didática, pretende-se que desenvolva os seguintes objetivos de aprendizagem:

- Explicar o que é AJAX.
- Saber criar conexões de segundo plano com o servidor web.
- Aprender a trabalhar com JSON e XML a partir de AJAX.
- Conhecer em detalhe os principais eventos do jQuery.
- Saber criar conexões de segundo plano com o servidor web.
- Aprender a trabalhar com JSON e XML a partir de AJAX.

INTRODUÇÃO

A programação web tem uma desvantagem quando se trata de criar aplicações: o código é interpretado no cliente e criado no servidor, pelo que, tanto quanto sabe até agora, para fazer alguma alteração de dados é necessário usar o PHP, HTML, etc., sendo também necessário fazer uma nova chamada para o servidor, e atualizar a página para devolver os novos dados ao navegador.

Nesta unidade ficará a conhecer o uso do AJAX, uma técnica de desenvolvimento web em JavaScript, que permite deixar um canal de comunicação aberto com o servidor, para que não seja necessário atualizar a página web.

Será também abordado o jQuery e ao seu uso, que é tão propagado que frameworks como o Bootstrap incluem jQuery por defeito; é por isso que nesta unidade didática estudará, em profundidade, as principais características desta mesma biblioteca. Verá ainda como interagir com os outros elementos da web, como XML e JSON.

1. O QUE É O AJAX?

AJAX é a sigla para Asynchronous JavaScript and XML (JavaScript Assíncrono e XML), e é uma forma de desenvolvimento de JavaScript que permite criar um canal de comunicação com o servidor do computador cliente em segundo plano, possibilitando assim a recuperação de dados do servidor sem a necessidade de atualizar uma página.

O AJAX permite, ainda, analisar e trabalhar com arquivos XML (linguagem de marcação extensível), uma forma de definição de dados padrão.

Esta tecnologia já era conhecida há vários anos, mas o seu uso só foi padronizado em 2005, quando recebeu o nome AJAX e passou a ser suportada pela maioria dos navegadores.

1.1. VANTAGENS DE USAR AJAX

- É suportado pelos principais navegadores.
- A interatividade com o utilizador foi melhorada.
- Não precisa de nenhuma instalação como Flash ou Java.
- A velocidade de carregamento, já que não é necessário recarregar um *website* inteiro, mas somente o conteúdo.

1.2. DESVANTAGENS DO AJAX

- Pode causar problemas em navegadores antigos.
- Requer execução de JavaScript ativada no navegador.
- Para tirar o máximo proveito do seu poder, é necessário o conhecimento de uma linguagem web como PHP, ASP, JSP, ect.
- O elemento "voltar à página anterior" é perdido, pois, ao ser carregado em segundo plano, o objeto de histórico não guarda essas páginas.

2. USO DO OBJETO XMLHTTPREQUEST

O objeto XMLHttpRequest é o elemento-chave para criar um canal de comunicação assíncrona com o servidor. Para isso, a primeira coisa que precisa é criar uma instância do referido objeto, que neste caso é um Internet Explorer Active X e que os outros navegadores implementaram.

O código necessário para criar esta instância é o seguinte:

```
If (window.XMLHttpRequest){ // é mozilla, chrome, safari...
objRequest = newXMLHttpRequest();
} elseif (window.ActiveXObject) { // é internet explorer
objRequest = newActiveXObject(“Microsoft.XMLHTTP”)
}
```

2.1. PROPRIEDADES

As propriedades do objeto XMLHttpRequest são:

- **onreadystatechande:** propriedade que contém o nome da função que será executada quando o estado do objeto mudar.

- **readyState:** armazena o estado atual do objeto. Este valor pode ser:
  - 0, não inicializado.
  - 1, a carregar (iniciar).
  - 2, a carregar (já com o cabeçalho e status).
  - 3, interativo (com dados parciais).
  - 4, concluídas.
- **responseText:** armazena a stringd evolvida pelo servidor.
- **responseXML:** armazena a string devolvida pelo servidor com a diferença de que está em XML.

2.2. MÉTODOS

Os métodos do objeto XMLHttpRequest são:

- **open:** abre um canal assíncrono com o servidor. Recebe vários parâmetros como na lista seguinte:

```
open(método, url, [assíncrono], [utilizador], [chave])
```

  - **método**, é o tipo de solicitação HTTP, pode ser GET ou POST.
  - **URL** para o qual a solicitação HTTP é feita.
  - **assíncrono**, (opcional) é indicado false, se a solicitação for em modo síncrono em vez de assíncrono.
  - **utilizador**, (opcional) utilizador para a autenticação do URL.
  - **chave**, (opcional) chave para a autenticação do URL.
- **send:** faz a solicitação HTTP, ou seja, até que este método seja chamado, a conexão é iniciada. Ele recebe um parâmetro padrão com um valor null que indica os possíveis parâmetros associados ao envio da solicitação.

```
send(dados)
```

- **abort:** interrompe a solicitação atual.

```
abort()
```

- **getAllResponseHeaders:** devolve uma string com os cabeçalhos de resposta do servidor.
- **getResponseHeader:** recebe como parâmetro o nome de um cabeçalho e devolve uma string com o conteúdo desse cabeçalho.
- **setRequestHeader:** recebe dois parâmetros: o primeiro indica o nome de um cabeçalho e, o segundo, o valor que quer atribuir. O método *open* deve ser chamado primeiro.

Exemplo 01

De seguida verá como integrar um objeto numa função para poder fazer chamadas assíncronas ao servidor. Neste exemplo simples, irá carregar num elemento div o texto que é lido através de AJAX de dois HTML, que serão criados antes com um texto. Para o funcionamento correto deste exemplo deverá colocar as páginas num servidor web IIS ou Apache.

Ficheiros a carregar:

texto1.html

```
Olá sou um texto que irei carregar na div do exemplo ajax<br/>

Como pode ver no meu código-fonte, o texto que irei retornar

Está preparado para suportar elementos HTML, já que este Texto
Retornado pode ser um HTML completo, uma tabela…
```

texto2.html

```
Eu sou outro excerto de texto de outro arquivo HTML no exemplo e
eles também irão carregar de forma assíncrona do AJAX; vou escrever
uma tabela <br/><br/>

<table border = ”1”><tr><td> col1 </td><td> col 2
</td></tr></table><br/>

Tabela criada
```

O código:

```
<html>

<head>

<title>Exemplo básico de AJAX</title>

<meta charset="utf-8">

<script type="text/javascript">

function carregar(numero){

    var objHttp=null;

    if(window.XMLHttpRequest) {

          objHttp = newXMLHttpRequest();

    } elseif(window.ActiveXObject) {

          objHttp = newActiveXObject("Microsoft.XMLHTTP");

    }

    objHttp.open("GET", "texto" + numero +".html" ,true);

    objHttp.onreadystatechange = function() {

          if (objHttp.readyState==4) {

          document.getElementById('caixa').innerHTML =
objHttp.responseText;

          }

    }

    objHttp.send(null);

}

</script>

</head>

<body>

<a href="#" onclick="carregar('1')">Carregar HTML 1</a><br/><br/>
```

```
<a href="#" onclick="carregar('2')">Carregar HTML
2</a><br/><br/><br/>

<div style="width:400px;height:400px;border:1px solid #FC0"
id="caixa">

Aqui são os textos carregados de AJAX

</div>

</div>

</body>

</html>
```

Neste exemplo, são carregados do AJAX dois textos HTML previamente criados. A linguagem AJAX ganha potencial ao juntar-se a outras linguagens de programação, como PHP e ASP, executadas no servidor. Graças a isto, é possível extrair informações de um banco de dados e exibi-las a pedido do utilizador sem ter que atualizar a página web.

No caso de ler um XML e não um HTML ou texto, bastaria fazer o *open* para o ficheiro XML e na parte de leitura dos dados devolvidos fazer a chamada dos elementos filhos do XML.

Exemplo 02

Este exemplo permite ter na página HTML os dados descarregados mais recentes sobre o estado do tempo num XML. Este exemplo precisa de um servidor *web* IIS ou Apache.

Ficheiro com os dados, tempo.xml

```
<?xmlversion="1.0" encoding="UTF-8" ?>

<tempo>

<cidade>

<nome>Lisboa</nome>

<estado>Nublado</estado>
```

```
<maxima>10</maxima>
<minima>-1</minima>
</cidade>
<cidade>
<nome>Porto</nome>
<estado>Chuvoso</estado>
<maxima>7</maxima>
<minima>-3</minima>
</cidade>
<cidade>
<nome>Coimbra</nome>
<estado>Nublado</estado>
<maxima>12</maxima>
<Minima>5</minima>
</cidade>
<cidade>
<nome>Faro</nome>
<estado>Sol</estado>
<maxima>10</maxima>
<minima>3</minima>
</cidade>
</tempo>
```

Código que recupera os dados do tempo.xml

```
<html>
<head>
<meta charset="UTF-8">
<title>Exemplo básico de AJAX e XML</title>
<script>
functioncarregar() {
            // ler o ficheiro
            var xhttp = newXMLHttpRequest();
xhttp.onreadystatechange = function() {
if (this.readyState == 4 &&this.status == 200) {
                    mostrar(this);
                }
            };
xhttp.open("GET", "tempo.xml", true);
xhttp.send();
            // carregar usando variáveis
function mostrar(xml) {
                var nom, i, objHttp, frase;
objHttp = xml.responseXML;
                frase = "";
                nome = objHttp.getElementsByTagName('nome');
                estado = objHttp.getElementsByTagName('estado');
max = objHttp.getElementsByTagName('maxima');
                min = objHttp.getElementsByTagName('minima');
                for (i = 0 ; i <nome.length; i++) {
```

```
frase += "Cidade: <b>" + nome[i].childNodes[0].nodeValue +
"</b><br/>";

frase += "Estado do céu: " + estado[i].childNodes[0].nodeValue +
"<br>";

frase += "Temperatura Máxima: " + max[i].childNodes[0].nodeValue +
"<br>";

frase += "Temperatura Mínima: " + min[i].childNodes[0].nodeValue +
"<br/><br/><br/>";

                }

document.getElementById("caixa").innerHTML = frase;

            }

        }

</script>

</head>

<body>

<a href="#" onclick="carregar()">Carregar tempo a partir de XML</a>

<div style="width:400px;height:400px; position:relative;"
id="caixa">

    Aqui os textos serão carregados do AJAX

</div>

</div>

</body>

</html>
```

3. JSON

JSON é um formato de troca de dados simples e é sigla para "JavascriptObject-Notation". A simplicidade e o baixo peso do JSON generalizaram o seu uso para a troca de dados entre sites ou entre servidores e aplicações online. Em JavaScript é processado de forma simples utilizando o método eval().

Segue-se o mesmo ficheiro de dados criado com XML e com JSON para ver as diferenças entre ambos:

XML

```
<ficha>
<nome>Carlos</nome>
<dados>
<idade>30</idade>
<cidade>Lisboa</cidade>
<profissao>informático</profissao>
</dados>
</ficha>
<ficha>
<nome>João</nome>
```

```
<dados>

<idade>33</idade>

<cidade>Porto</cidade>

<profissao>professor</profissao>

</dados>

</ficha>

JSON

{

"ficha":[

{"nome": "Carlos",

"parametros": {"idade" : "30", "cidade": "Lisboa", "profissao":
"informático"}},

{"nome": "João",

"parametros": {"idade" : "33", "cidade": "Porto", "profissao":
"professor" }}

]

}
```

Com este sistema pode criar ficheiros que servem de banco de dados, sem a necessidade de utilizar SQL.Para pequenos projetos na web ou em muitos outros casos pode utilizar o JSON como recurso oferecido por outras entidades, como por exemplo, o Twitter, que fornece os tweets por conta online ou alguns websites de previsão do tempo que também usam JSON para publicar esses mesmos dados.

4. JSON E AJAX

Depois de aprender sobre o JSON e como criar ficheiros desse tipo, verá agora como pode aceder a esses dados. Como visto no ponto anterior, em JavaScript existe um método chamado eval() que interpreta o tipo de dados JSON, decompondo-os em quantos arrays necessários (de acordo com os dados).

Tendo em consideração oJSON anterior, do ficheiro de funcionários, segue-se um exemplo em que. ao clicar no botão "Arquivo de funcionários", carrega os dados do funcionário numa div.

Exemplo 03

Neste exemplo, irá criar um HTML estático chamado json.html, onde vão ser carregados os dados dos funcionários do ficheiro JSON (com o nome json.txt). Os dados JSON podem ser criados a partir de aplicações ou bancos de dados, podendo lê-los a partir de ficheiros HTML, ASP, URL da aplicação, entre outros.

Exemplo:

```
<html>

<head>

<title>Exemplo básico AJAX JSON</title>

<meta charset="utf-8">
```

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.
js"></script>

<script type="text/javascript">

function carregar(){

    $.ajax({

          url : 'json.txt',

          type : 'GET',

          success :function(data) {

                objeto_json = eval("("+data+")");

// ler o conteúdo

                var frase = "";

                for (i=0;i<objeto_json.ficha.length;i++){

          frase = frase + "Nome: <b>" + objeto_json.ficha[i].nome
+ "</b><br/>";

          frase = frase + "Idade: " +
objeto_json.ficha[i].parametros.idade + "<br/>";

          frase = frase + "Cidade: " +
objeto_json.ficha[i].parametros.cidade + "<br/>";

          frase = frase + "Profissão: " +
objeto_json.ficha[i].parametros.profissao + "<br/>";

          }

          $("#caixa").html(frase);

 },

error :function(xhr, status) {

alert('Ocorreu um erro.');

 }

});

}

</script>
```

```
</head>

<body>

<a href="#" onclick="carregar()">Arquivo de funcionários</a>

<div style="width:400px;height:400px; position:relative;"
id="caixa">

Vão ser aqui carregados os ficheiros apartir de AJAX

</div>

</body>

</html>
```

5. O QUE É O JQUERY?

O jQuery é uma biblioteca JavaScript pública e gratuita, criada para interagir com elementos HTML através de várias funções predefinidas que permitem criar efeitos visuais de forma mais rápida e fácil.

Por ser open source, possui atualizações e melhorias contínuas, e por ser tão utilizado existem inúmeras páginas de explicação e exemplos com todos os tipos de código.

5.1. INSTALAÇÃO E UTILIZAÇÃO

Na página oficial do jQuery, pode fazer o download da biblioteca, ou encontrar as chamadas para os registos CDN e carregá-los a partir do serviço jQuery online.

Para aceder a estes materiais, entre em https://jquery.com/download/ e faça o download da biblioteca ou use o CDN. Neste caso, será utilizado o CDN para evitar o download dos ficheiros e para que os exemplos e exercícios não precisem de ficheiro externo. Utilizar o CDN do Google:

- https://developers.google.com/speed/libraries/#jquery

Para instalar o jQuery no projeto web, terá de carregar o ficheiro como se estivesse a criar um script JavaScript:

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.
js"></script>
```

Assim que fizer o carregamento da biblioteca, ela estará acessível a partir das páginas que carregar.

Exemplo 10-01

O exemplo seguinte mostra como carregar a biblioteca e verificar se o carregamento está correto utilizando o método ready, que será visto mais detalhadamente no ponto “Eventos”.

Exemplo:

```
<html>
<head>
<title>jQuery</title>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.
js"></script>
</head>
<body>
<script>
 $(document).ready(function () {
    alert("jQuery carregado");
 });
</script>
</body>
</html>
```

5.2. SELETORES

A primeira coisa a aprender sobre jQuery é o sistema de chamadas. Para fazer as chamadas para a biblioteca, utiliza-se o cifrão ($) ou 'jQuery' (embora o mais padronizado seja o cifrão, por ser mais curto e rápido de usar).

Os seletores permitem trabalhar com os elementos da página, seja para ler os seus valores, atributos e textos, modificar os elementos ou geri-los.

5.2.1. SELETORES BÁSICOS

CÓDIGO SELETOR	DESCRIÇÃO
$(“*”)	Seleciona todos os elementos da página
$(“div”)	Seleciona os elementos da tag indicada (neste exemplo, as divs)
$(“.classeEx”)	Seleciona os elementos que tenham a classe indicada
$(“div.classeEx”)	Seleciona os elementos do tipo indicado (neste caso as divs) que possuem a classe indicada (neste caso “classeEx”)
$(“#nomeId”)	Seleciona o elemento com o atributo id com o valor especificado
$(“#nomeId.classeEx”)	Seleciona os elementos dentro do id indicado que possuem a classe indicada (neste caso os elementos com a classe “classeEx” que estão dentro do elemento com o id “nomeId”)
$(“div,a”)	Ao separar as seleções com vírgulas (,), os seletores são combinados, ou seja, são selecionados todos os elementos que tenham algum dos seletores indicados, neste caso seriam todas as divs e os “a”
$(“a[rel]”)	Seleção de atributos; permite selecionar os elementos que possuem o atributo indicado
$(“a[rel=’nofollow’]”)	A seleção de atributos e valores permite selecionar os elementos que possuem um valor específico para um determinado atributo. Se quiser selecionar todos aqueles que não contêm um valor indicado, basta mudar o '=' para '! ='
$(“a[href^=’http://”’)	Seleciona os elementos com um atributo cujo valor começa com o valor indicado, neste caso os links (a) cujo atributo href começa com “http: //”
$(“a[href$=’.pt”’)	Seleciona os elementos com um atributo cujo valor termine com o valor indicado
$(“a[href*=’google”’)	Seleciona os elementos com um atributo cujo valor contém a string especificada

5.2.2. SELETORES DE WIDGETS

CÓDIGO SELETOR	DESCRIÇÃO
$(":button")	Seleciona todos os elementos button
$(":checkbox")	Seleciona todas as checkboxes
$(":image")	Seleciona todas as imagens
$(":input")	Seleciona todos os inputs
$(":select")	Seleciona todos os selects
$("visible")	Seleciona todos os elementos visíveis
$("input:checked")	Seleciona todas os inputs que estão selecionados, ou seja, reúne as checkboxes marcadas
$(":contains('curso')")	Seleciona os elementos que contêm a string indicada como texto, por exemplo, $("div: contains('web')") selecionará todos os elementos div que têm a palavra "web" como texto (não como um atributo)
$(":empty")	Seleciona elementos vazios; por exemplo, $ ("td:empty") irá selecionar todas as células das tabelas que estiverem vazias

5.2.3. SELETORES DE HIERARQUIA

CÓDIGO SELETOR	DESCRIÇÃO
$("div h2")	Seleciona os elementos que descendem de outro, neste caso, o h2 que está dentro de uma div
$("div.classeEx> h2")	Seleciona os elementos que são filhos do elemento indicado; neste caso, selecionará os elementos da div que têm a classe "classeEx", e os elementos h2 que estão no próximo nível direto
$("tr:first-child")	Seleciona o primeiro filho de um elemento
$("tr:last:child")	Seleciona o último filho de um elemento
$("div:has(p)")	Seleciona os elementos que têm qualquer aparência do elemento marcado em seu conteúdo neste exemplo, as divs que contêm um elemento "p"

Todos os seletores, incluindo os menos utilizados, e os seus respetivos exemplos podem ser vistos em https://api.jquery.com/category/selectors/

5.3. MÉTODOS

Os métodos permitem manipular os elementos web em tempo real, conseguindo modificar, adicionar, apagar ou alterar as suas propriedades.

Segue-se uma lista dos métodos mais utilizados:

- **.addClass(nome da classe)**: permite adicionar uma classe ao elemento ou elementos selecionados com jQuery.
- **.after(código)**: adiciona o código especificado depois de todos os elementos selecionados.
- **.append(código)**: adiciona o código dentro dos elementos selecionados no final do código que eles já contêm.

- **.attr(nome do atributo)**: devolve o valor do atributo marcado do primeiro aparecimento do elemento selecionado. Se, por exemplo, selecionar todas as entradas e quiser remover o atributo de nome de cada uma, deverá utilizar o each (que será abordado mais tarde nesta unidade).
- **.before(código)**: insere o código especificado antes de todos os elementos selecionados.
- .**clone()**: permite fazer uma cópia exata do elemento selecionado.
- **.css(estilo)**: permite recolher o valor de um estilo do elemento, por exemplo, a cor de um texto.
- .**css(estilo, valor)**: é um dos mais utilizados e permite alterar os estilos de um elemento (cor, tamanho, forma, etc.), passando o estilo para alterar, por exemplo, color e o valor '#ff0000'.
- **.hasClass(classe)**: devolve "true", se o elemento selecionado tem a classe indicada, caso contrário devolve "false".
- **.html(valor)**: permite recolher ou modificar o conteúdo HTML de um elemento; se passar vazio, o valor irá recolher o código HTML do elemento, por exemplo, uma div. Se passar um valor, ele atualizará o código HTML do elemento com o valor enviado, eliminando todo o código HTML que o elemento possuía até ao momento. Não é válido alterar o valor de um input, por exemplo.
- **.remove()**: exclui um elemento do código.
- **.removeAttr(atributo)**: remove um atributo do(s) elemento(s) selecionado(s).
- **.removeClass(classe)**: remove uma classe do elemento selecionado.
- **.val(valor)**: permite recolher ou modificar o valor de um input; se o valor passar vazio recolhe o valor, se o preencher modifica o valor.

Exemplo 10-02

O exemplo a seguir mostra alguns métodos que afetam a página web e o seu uso prático.

Código de exemplo:

```
<html>

<head>

<title>jQuery - métodos</title>

<meta charset="UTF-8">

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.
js"></script>

    <style>

    .adClasse {color:#ff0000;}

    .caixa {width:150px; height:150px; position:relative;
float:left; border:1px solid #000;}

    .gClasse {background-color:green;}

    .vClasse {background-color:red;}

    </style>

</head>

<body>

<div class="caixa impar" id="caixa1">Caixa 1</div>

<div class="caixa par gClasse" id="caixa2">Caixa 2</div>

<div class="caixa impar" id="caixa3">Caixa 3</div>

<div class="caixa par" id="caixa4">Caixa 4</div>

<div class="caixa impar" id="caixa5">Caixa 5</div>

<br/><br/><br/><br/><br/><br/><br/><br/><br/>
```

```
texto 1: <input type="text" name="texto"/>

texto 2: <input type="text" name="texto2"/>

<br/><br/>

<a href="">REINICIAR EXEMPLO</a>

<br/><br/>

<a href="#" onclick="$('#caixa1').addClass('adClasse');return
false">Adicionar classe à caixa 1</a><br/><br/>

<a href="#" onclick="$('#caixa2').append('<br/><b>texto
adicionado</b>');return false">Adicionar texto à caixa
2</a><br/><br/>

<a href="#"
onclick="$('#caixa5').after($('#caixa2').clone());return
false">Clonar a caixa 2 e depois da caixa 5</a><br/><br/>

<a href="#" onclick="$('#caixa3').css('background-
color','#ff0000');return false">Mudar o css da caixa 3 colocando um
fundo vermelho</a><br/><br/>

<a href="#" onclick="$('#caixa4').remove();return false">Eliminar a
caixa 4</a><br/><br/>

<a href="#" onclick="$('#caixa2').removeClass('gClasse');return
false">Eliminar a classe que faz a caixa 2 verde</a><br/><br/>

<a href="#" onclick="$('input[name=\'texto\']').val('valor
inserido');return false">Escreva o valor de um input
(texto1)</a><br/><br/>

<a href="#"
onclick="if($('input[name=\'texto2\']').val()=='vermelho'){
$('.caixa').addClass('vClasse') };return false">Se escrever no
texto 2 "vermelho", atualiza a cor de todas as caixas para
vermelho</a><br/><br/>

</body>

</html>
```

5.4. EVENTOS

Os eventos JQuery permitem capturar momentos e eventos da página web, como por exemplo um click ou um carregamento. Segue-se uma lista com os principais:

- **blur()**: este evento será executado ao sair do elemento selecionado (foco perdido); principalmente utilizado em formulários.

```
$("input").blur(function(){

alert("saiu do input.");

});
```

- **change()**: será executado quando o valor do elemento selecionado for alterado. É usado principalmente em formulários, para realizar alguma ação ao modificar um campo, por exemplo, habilitar o botão para enviar o formulário apenas quando algum valor for alterado.

```
$("input").change(function(){

alert("o texto foi alterado.");

});
```

- **click()**: permite executar o evento click de um elemento do JavaScript, por exemplo, um botão num formulário.
- **click(função)**: permite adicionar ao elemento selecionado o evento click com uma função específica, por exemplo, um alerta de texto.

```
$("p").click(function(){

alert("foi clicado no elemento p");

});
```

- **dbclick()**: igual ao anterior (click), mas com o duplo clique do rato.
- **event.preventDefault()**: permite parar o evento típico de um elemento, por exemplo, evitar que ao clicar num link a URL seja lançada.

```
$("a").click(function(event){

 event.preventDefault();

});
```

- **focus()**: permite colocar o foco no elemento que pretender; normalmente é usado ao entrar numa página com um formulário para posicionar o cursor no primeiro elemento do formulário.
- **hover(função de entrada, função de saída)**: permite definir uma função ao entrar um elemento e ao sair do elemento, por exemplo, pode mudar a cor do texto ao passar o cursor sobre uma caixa e deixá-la como estava ao sair.

```
$("div").hover(function(){

$(this).css("color", "red");

}, function(){

$(this).css("color", "black");

}

);
```

- **keyup()**: é executado ao pressionar uma tecla do teclado enquanto o foco está no elemento; pode ser utilizado para pesquisar uma correspondência ao escrever dentro de um campo de texto (será visto no tópico "Ajax"). Existem ainda os eventos keydown e keypress.

  Também serve para iniciar o evento sem tocar no teclado, deixando o parâmetro vazio (semelhante ao click).
- **mouseup()**: é executado ao clicar com o rato sobre o elemento que está selecionado.

```
$("div").mouseup(function(){

alert('foi clicado numa div');

})
```

- **on(evento, função)**: é um manipulador de eventos utilizado para detetar eventos sem ter de chamar a função específica, por exemplo, para detetar o clique:

```
$("div").on("click", function(){

alert("foi clicado numa div");

});
```

- **resize(função)**: é executado quando o elemento selecionado muda de tamanho, por exemplo, é utilizado para detetar a mudança na resolução do ecrã:

```
$(window).resize(function(){

alert("o ecrã foi redimensionada");

});
```

- **scroll(função)**: é utilizado para detetar o scroll no elemento selecionado, por exemplo, ao percorrer um elemento que dá um aviso ou mostra uma imagem; é muito utilizado para mover elementos estáticos da página web durante o scroll.
- **submit()**: permite executar o evento de envio do elemento selecionado, podendo também adicionar uma função ao evento. Se passar a função como um parâmetro de submit(função), o uso mais comum é enviar um formulário a partir de um elemento que não está dentro do formulário ou que não seja o botão enviar (um botão com um link).
- **ready()**: é o evento mais utilizado. Esta função é uma imitação da função JavaScript genérica window.onload, mas enquanto a função JavaScript espera que todos os elementos da página sejam carregados, incluindo estilos e imagens, a função jQuery é executada assim que carrega a estrutura da página web e, portanto, as funções criadas com jQuery respondem de forma mais rápida do que em JavaScript. O seu uso mais comum é executar códigos após carregar toda a página web, por exemplo, chamadas para bibliotecas externas.

```
$(document).ready( function() { ...código... } );
```

Exemplo 10-03

Neste exemplo, verá os diferentes usos de eventos que foram explicados anteriormente.

Exemplo:

```
<html>

<head>

<title>jQuery - eventos</title>

<meta charset="utf-8">

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.
js"></script>

    <style>

          .caixa{width:150px;height:150px;border:1px solid #000}

    </style>

</head>

<body>

<div class="caixa impar" id="caixa">Caixa 1</div>

<br/><br/>

<form method="post" action="">

    texto 1: <input type="text" name="caixa1" id="caixa1"/>

    texto 2: <input type="text" name="caixa2" id="caixa2"
value="33"/>

    texto 3: <input type="text" name="caixa3" id="caixa3"/>

    <button onclick="alert('clique em mim, sou um botão do
formulário')" id="enviar">Enviar</button>

</form>

<br/><br/>
```

```
<a href="#" onclick="$('#enviar').click();return false">Simular
click no botão</a><br/><br/>

<script>

    $(document).ready( function() {

          $("#caixa1").blur(function(){

                alert('saiu do texto 1');

          });

          $("#caixa2").change(function(){

                alert('alterou a caixa 2');

          });

          $("#caixa").hover(function(){

                $(this).css("background-color", "red");

                }, function(){

                $(this).css("background-color", "white");

                }

          );

          $("#caixa3").keyup(function(event){

          alert('na caixa 3, pressionou:
'+String.fromCharCode(event.which));

          });

          $("#caixa").on("click", function(){

                alert("clicou na caixa");

          });

    } );

</script>

</body>

</html>
```

5.5. ITERAÇÃO DE ELEMENTOS

Com o jQuery é possível selecionar mais do que um elemento, graças aos seletores por nome, classe, tipo de elemento, etc. O jQuery tem uma função que permite percorrer os objetos selecionados (também pode ser utilizada para percorrer um array).

- each(objeto, função (key, valor))
- object.each(function () {})

É, por exemplo, muito utilizada para percorrer as checkboxes de um formulário, para saber quais são e se estão marcadas, antes de enviar o formulário, ou para dar respostas ao utilizador sem ter de enviar o formulário.

Exemplo 10-04

No exemplo seguinte, verá a sua utilização para percorrer os elementos selecionados. Para o exemplo, é utilizada uma função jQuery típica, designada is(), que será explicada após o exemplo.

Exemplo:

```
<html>

<head>

<meta charset="UTF-8">

<title>Jquery - each</title>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.
js"></script>

</head>

<body>

<br/><br/>

Selecione os seus desportos favoritos<br/><br/>
```

```
Futebol: <input type="checkbox" name="Futebol"
value="Futebol"/><br/>

Basquetebol: <input type="checkbox" name="Basquetebol"
value="Basquetebol"/><br/>

Andebol: <input type="checkbox" name="Andebol"
value="Andebol"/><br/>

Ténis: <input type="checkbox" name="Tenis" value="Ténis"/><br/>

Ciclismo: <input type="checkbox" name="Ciclismo"
value="Ciclismo"/><br/>

<br/><br/>

<a href="#" onclick="informacionChecks();return false">O que é que
está marcado?</a><br/><br/>

<script>

function informacionChecks(){

$("input:checkbox").each(function() {

 if($(this).is(':checked')){

    alert($(this).val());

}

 });

}

</script>

</body>

</html>
```

- **is()**: como a sua tradução indica, permite comparar um elemento para descobrir se é um seletor, um objeto ou um elemento específico, devolvendo true se for; no exemplo anterior, compara-se com um seletor de checked, que permite selecionar os itens marcados.
- **inArray(valor, array)**: esta é outra função que permite agilizar o trabalho com arrays em jQuery, e que permite saber se um elemento está contido num array sem ter de o percorrer com as funções each ou loopfor.

Exemplo 10-05

O exemplo seguinte mostra como pode, dado um array, pesquisar o elemento colocado num campo de texto, e como a função dirá se esse elemento está contido.

Código de exemplo:

```
<html>

<head>

<meta charset="UTF-8">

<title>jQuery - each</title>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.
js"></script>

<script>

    vector = ['camião','bicicleta','carro','mota'];

</script>

</head>

<body>

<br/><br/>O array é composto por:<br/>

<script>

$.each(vector,function(key,valor){

    document.write('<b>'+valor+'</b><br/>');

 });

</script>

<br/><br/><input type="text" id="buscar" value=""/>

<a href="#" onclick="buscarEnVector($('#buscar').val());return
false">Existe no vetor?</a><br/><br/>

<script>

function buscarEnVector(valor){
```

```
    if($.inArray(valor,vector) !== -1 ){
          alert('Existe!');
    }else{
          alert('Não existe!');
    }
}
</script>
</body>
</html>
```

5.6. EFEITOS

O jQuery permite adicionar efeitos visuais a uma página web de uma forma simples, através da criação de animações, graças ao CSS, e de alguns efeitos padrão.

Os efeitos mais utlizados estão predefinidos em jQuery e não precisa de os criar. São eles:

- **.show**: mostra o item selecionado, ou seja, exibe um item que estava oculto.
- **.hide**: oculta o item selecionado, ou seja, oculta um item que estava visível.
- **.fadeIn**: animação que mostra um elemento, alterando progressivamente a opacidade para 100%.
- **.fadeOut**: animação que oculta um elemento, alterando progressivamente a opacidade para 0%.
- **.slideDown:** mostra o item selecionado com um movimento de deslize vertical de cima para baixo; esse elemento deve estar escondido anteriormente.

- **.slideUp**: oculta o elemento selecionado com um movimento de deslize vertical por baixo; esse elemento deve estar visível anteriormente.

Exemplo 10-05

No exemplo seguinte, verá os usos das animações descritas anteriormente.

Exemplo:

```
<html>

<head>

<title> jQuery - animação </title>

<meta charset="UTF-8">

<script src =
"https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>

<style>

.caixa {width: 150px; height: 150px; position: relative; float:
left; border: 1px solid # 000}

</style>

</head>

<body>

<br/><br/>

<div class = "odd caixa" id = "caixa1"> Caixa 1 </div>

<div class = "even caixa" id = "caixa2"> Caixa 2 </div>

</br></br></br></br></br></br></br></br></br></br></br></br>

<a href=""> REINICIAR O EXEMPLO </a>

</br></br>

<a href="#" onclick="$('#caixa1').hide()"> Ocultar caixa 1
</a></br>

<a href="#" onclick="$('#caixa1').show()"> Mostrar caixa 1
</a></br>
```

```
<a href="#" onclick="$('#caixa2').fadeOut()"> Ocultar caixa 2 pela
opacidade </a></br>

<a href="#" onclick="$('#caixa2').fadeIn()"> Mostrar caixa 2 pela
opacidade </a></br>

<a href="#" onclick="$('#caixa1').slideUp()"> Ocultar caixa 1 com o
deslize vertical inferior </a></br>

<a href="#" onclick="$('#caixa1').slideDown()"> Mostrar o deslize
vertical superior da caixa 1 </a></br>

</body>

</html>
```

5.6.1. DURAÇÃO DOS EFEITOS

O jQuery permite modificar a duração dos efeitos para que sejam mais ou menos rápidos; para isso, passará o parâmetro de duração para a função:

- $(“div”).fadeIn(parâmetro)

  O parâmetro pode ser definido por defeito no jQuery:

  - slow: 600 milissegundos.
  - fast: 200 milissegundos.

  Ou pode passar diretamente o número em milissegundos que desejar.

Exemplo 10-06

Este exemplo é exatamente igual ao anterior, mas alterando os efeitos para terem durações diferentes.

Exemplo:

```
<html>

<head>

<title> jQuery - animação </title>

<meta charset="UTF-8">

<script src =
"https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>

<style>

.caixa {width: 150px; height: 150px; position: relative; float:
left; border: 1px solid # 000}

</style>

</head>

<body>

<br/><br/>

<div class = "odd caixa" id = "caixa1"> Caixa 1 </div>

<div class = "even caixa" id = "caixa2"> Caixa 2 </div>

</br></br></br></br></br></br></br></br></br></br></br></br>

<a href=""> REINICIAR O EXEMPLO </a>

</br></br>

<a href="#" onclick="$('#caixa1').hide('slow')"> Ocultar caixa 1
</a></br>

<a href="#" onclick="$('#caixa1').show('fast')"> Mostrar caixa 1
</a></br>
```

```
<a href="#" onclick="$('#caixa2').fadeOut(2000)"> Ocultar por caixa
de opacidade 2 </a></br>

<a href="#" onclick="$('#caixa2').fadeIn(100)"> Mostrar pela caixa
de opacidade 2 </a></br>

<a href="#" onclick="$('#caixa1').slideUp(500)"> Ocultar caixa 1
com o deslize vertical inferior </a></br>

<a href="#" onclick="$('#caixa1').slideDown(1500)"> Mostrar o
deslize vertical superior da caixa 1 </a></br>

</body>

</html>
```

5.6.2. VINCULAR FUNÇÕES AOS EFEITOS

O jQuery permite vincular uma função aos efeitos, ou seja, quando um efeito termina (por exemplo, um fadeIn), executa um código JavaScript para realizar alguma ação. Para isso, basta passara função que pretende executar como segundo parâmetro no efeito ou criar a função no próprio código:

```
$("div").fadeIn(300, function() {alert('olá!');})
```

5.6.3. EFEITOS PERSONALIZADOS

É possível criar os próprios efeitos graças ao CSS utilizando o seguinte método, que permite modificar os valores e propriedades CSS de um elemento com uma animação.

```
.animate(mudanças de CSS, duração, função)
```

Como nas funções anteriores, poderá, além das alterações, indicar uma duração e vincular uma função ao final da animação.

Exemplo 10-07

Neste exemplo, utilizando a função **animate**, é possível criar vários efeitos personalizados.

Exemplo:

```
<html>

<head>

<title>jQuery - animate</title>

<meta charset="UTF-8">

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.
js"></script>

<style>

    .caixa{width:150px;height:150px;position:relative;float:left;bo
rder:1px solid #000}

</style>

</head>

<body>

<br/><br/>

<div class="caixa impar" style="background-
color:#ff0000;color:#fff;" id="caixa1">Caixa 1</div>

<div class="caixa par " style="border:1px solid #d4d4d4"
id="caixa2">Caixa 2</div>

<br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>

<div class="" style="border:1px solid #d4d4d4" id="caixa3">

Caixa 3 esta é a caixa com mais texto Caixa 3 esta é a caixa com
mais texto Caixa 3 esta é a caixa com mais texto Caixa 3 esta é a
caixa com mais texto Caixa 3 esta é a caixa com mais texto Caixa 3
este é o caixa com mais texto abaixo de 3 esta é a caixa com mais
texto Caixa 3 esta é a caixa com mais texto Caixa 3 esta é a caixa
com mais texto

</div>
```

```
</br></br></br></br></br></br>

<a href="">REINICIAR EXEMPLO</a>

</br></br>

<a href="#" onclick="animacao1()">Animação 1 altera opacidade e
tamanho</a></br>

<a href="#" onclick="animacao2()">Animação 2 altera borda, margem
esquerda e tamanho</a></br>

<a href="#" onclick="animacao3()">Alteração do tamanho do texto da
animação 3</a></br>

<script>

function animacao1(){

    $('#caixa1').animate({opacity: '0.4',width : 500},1000);

 }

function animacao2(){

$('#caixa2').animate({borderWidth: '10px',marginLeft: '120px',width
: 500},1000);

 }

function animacao3(){

    $('#caixa3').animate({fontSize: '20px'},1000);

 }

</script>

</body>

</html>
```

5.7. AJAX COM JQUERY

O jQuery possui um método principal para trabalhar com chamadas Ajax, designado **$.ajax**. Este método permite fazer as chamadas de forma mais rápida e eficiente. É, basicamente, uma ferramenta para fazer chamadas AJAX de forma simplificada.

O método também permite um controlo total sobre o resultado da chamada AJAX, podendo definir de forma simples se pretende que algum evento ou código seja executado de acordo com o estado da chamada.

Opções de método:

- **async**: estabelece se a chamada é assíncrona. Por defeito, é "true", o que significa que não bloqueia a execução de outros códigos. Se for alterado para "false", o navegador aguardará o fim da chamada para continuar a carregar os outros scripts e dados.
- **complete**: indica uma função quando a chamada é completada, tenha ela falhado ou não.
- **data**: são as informações enviadas para a chamada, por exemplo *ação=guardar &nome=Joaquim*.
- **dataType**: é o tipo de informação a ser recebida. Se nada for indicado, o jQuery lê o resultado e, por defeito, trata-o como texto simples.
- **error**: estabelece uma função que será chamada se a função criar um erro na sua execução.
- **sucess**: define uma função que será chamada quando a função for concluída com sucesso e sem erros.
- **type**: o método de entrega de dados padrão é GET e pode ser alterado para POST.
- **url**: o caminho utilizado para fazer a chamada do Ajax.

Segue-se um exemplo:

```
$.ajax({
url : 'post.php', // o url para o pedido
 data : 'accao=guardar&nome=joaquim', // a informação para enviar
 type : 'GET', // especificar se será uma solicitação POST ou GET
 // código a ser executado se a solicitação for bem-sucedida;
success : function(data) {
 if(data=='OK'){
alert('Guardado OK'); // os dados foram salvos
 }else{
 alert('Error: '+data);// os dados não foram salvos
 }
 },
 // código a ser executado se a solicitação falhar;
 error : function(xhr, status) {
 alert('Desculpe, houve um problema');
 },
});
```

Observará que dentro do sucess é feita uma pergunta para saber o que é que ela devolve e que, se a solicitação não for feita com sucesso, mostra um alerta de erro. É necessário distinguir entre a falha da função que executaria o alerta, da falha de solicitação (“Desculpe, houve um problema”) criada a partir daquela, pois a chamada pode ter sido feita corretamente, mas ter falhado devido a um problema de PHP ou devido ao facto de que não pôde ser executado o código da URL de que foi chamada.

Exemplo 10-08

Este exemplo mostra o uso de jQuery Ajax para fazer uma chamada para uma página web e recolher dados (neste caso, um .txt). Este exemplo requer um servidor web.

Ficheirotexto.txt:

```
Eu sou um texto carregado com Ajax
```

Exemplo:

```
<html>

<head>

<title>Exemplo básico AJAX jQuery</title>

<meta charset="UTF-8">

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.
js"></script>

<script type="text/javascript">

function carregar(){

    $.ajax({

           url : 'texto.txt',

           type : 'GET',

           success : function(data) {

                  $('#caixa').html(data);

 },

 error : function(xhr, status) {

 alert('Desculpe, houve um problema');

 }

});
```

```
}

</script>

</head>

<body>

<a href="#" onclick="carregar()">Carregar texto</a>

<div style="width:400px;height:400px; position:relative;"
id="caixa">

Aqui será carregado o texto AJAX

</div>

</body>

</html>
```

O jQuery tem uma função específica que é uma sub-rotina derivada de $.ajax que permite o carregamento automático do conteúdo de um ficheiro num elemento. No exemplo anterior é carregado o conteúdo do ficheiro texto.txt numa camada do HTML graças a $.ajax. Para este tipo de cargas que não requerem um tratamento posterior dos dados obtidos, existe outra função:

- **$.load**: permite o carregamento direto do conteúdo, como poderá ver no exemplo seguinte.

Exemplo 10-09

Este exemplo é idêntico ao anterior, mas como não é necessário processar os dados, será utilizado o **$.load** em vez do **$.ajax**. Este exemplo precisa de um servidor web para funcionar corretamente.

Ficheiro texto.txt:

```
Eu sou um texto carregado com Ajax
```

Exemplo:

```
<html>

<head>

<title>Exemplo básico AJAX jQuery</title>

<meta charset="UTF-8">

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.
js"></script>

<script type="text/javascript">

function carregar(){

    $('#caixa').load('texto.txt');

 }

</script>

</head>

<body>

<a href="#" onclick="carregar()">Carregar texto</a>

<div style="width:400px;height:400px; position:relative;"
id="caixa">

Aqui será carregado o texto AJAX

</div>

</body>

</html>
```

Esta instrução é utilizada para carregar dados diretos. Para tratamento de dados ou carregamento de elementos, geralmente utiliza-se $.ajax, o que permite configurar o carregamento e processar os dados.

A função $.ajax é amplamente utilizada para verificar e enviar dados de formulário sem a necessidade de atualizar a página, o que acontece sem usar Ajax.

Para isso, o jQuery fornece uma instrução que permite recolher os dados do formulário para poder usá-los como envio de dados da instrução AJAX sem ter de dar forma e ler dado a dado, o que é muito útil em formulários que vão crescendo em número de campos, como um formulário que pede endereços de e-mail. A instrução é a seguinte:

- .serialize

Exemplo 10-10

Para ver como essa instrução funciona, segue-se um exemplo muito simples de recolha de dados de um formulário, para que observe como estes são recolhidos.

```
<html>

<head>

<title>Exemplo básico AJAX jQuery</title>

<meta charset="UTF-8">

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.
js"></script>

<script type="text/javascript">

function serie(){

    $('#dados').html($('#formulario').serialize());

}

</script>

</head>

<body>

<form method="post" action="" id="formulario">

    Nome <input type="text" name="nome" /><br/>

    Apelido <input type="text" name="apelido" /><br/>

    Telemóvel <input type="text" name="telemovel" /><br/>

    Email <input type="text" name="email" /><br/>
```

```
    Idade <input type="text" name="idade" /><br/>
</form>
<a href="#" onclick="serie()">Serializar formulario</a>
<br/><br/>
<div id="dados"></div>
</body>
</html>
```

CONCLUSÃO

O jQuery é uma ferramenta poderosa que não só torna a programação mais fácil, mas também permite uma velocidade de processamento de dados mais rápida e um controlo total sobre os elementos, eventos e métodos na página.

Graças ao $.ajax, que é um sistema de comunicação com o servidor do JavaScript ao computador do cliente, permite trazer dados em tempo real e sem a necessidade de atualizar a página web. Além disso, os dados podem ser obtidos de XML, JSON ou diretamente de texto ou HTML.

AUTOAVALIAÇÃO

1. **O que é o jQuery?**

   a) JavaScript versão 2.0.

   b) Empresa responsável pela definição do JavaScript.

   c) Uma biblioteca JavaScript.

   d) Uma função JavaScript.

2. **Quais são as principais bibliotecas JavaScript?**

   a) Bily, Tommy y Galant.

   b) PHP, ASP e CGI.

   c) jQuery, Dojo, YUI, Motools, Prototype e ExtJS.

   d) Click, Toggle, Bind e After.

3. **Para que serve a função add em jQuery?**

   a) Para adicionar.

   b) Para atualizar a versão jQuery.

   c) Para carregar um plug-in.

   d) Para adicionar elementos a uma seleção.

4. Qual é a função que permite adicionar uma nova classe a um elemento?

a) setClass.

b) addClass.

c) applyClass.

d) classify.

5. Qual é a função que permite adicionar um novo HTML a um documento?

a) add.

b) append.

c) incorporate.

d) include.

6. O que se deve fazer para usar as funcionalidades jQuery num documento?

a) Incluir a biblioteca no cabeçalho do código.

b) Incluir a biblioteca no documento de um repositório público.

c) Incluir a biblioteca no código, seja através de ficheiro local ou de um repositório público.

d) Utilizar o cifrão dentro das tags<jQuery>.

7. Qual é a diferença entre onLoad e $(document).ready?

a) onLoad é uma função jQuery enquanto o ready é uma função JavaScript padrão.

b) onLoad espera para carregar todas as imagens enquanto o ready é executado quando o HTML está pronto.

c) O ready existe, enquanto o onLoad não.

d) ready e onLoad são nomes diferentes para a mesma função.

8. O método after:

a) Adiciona HTML após os elementos selecionados.

b) Adiciona BODY após os elementos selecionados.

c) Adiciona HEAD após os elementos selecionados.

d) Adiciona FOOTER após os elementos selecionados.

9. A função add:

a) Permite adicionar mais atributos a um conjunto de elementos já selecionados.

b) Permite adicionar mais elementos a um conjunto de elementos já selecionados.

c) Permite adicionar mais configurações a um conjunto de elementos já selecionados.

d) Permite subtrair configurações de um conjunto de elementos selecionados.

10. O jQuery permite:

a) Um controlo avançado sobre a execução das funções nos elementos do formulário.

b) Um feedback simples sobre a execução das funções nos elementos do formulário.

c) Um controlo simples sobre a execução das funções nos elementos do formulário.

d) Um feedback avançado sobre a execução das funções nos elementos do formulário.

SOLUÇÕES

1.	a	**2.**	c	**3.**	d	**4.**	b	**5.**	b
6.	c	**7.**	b	**8.**	a	**9.**	b	**10.**	c

PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO

Para saber mais sobre o jQuery e as suas utilizações, visite os seguintes websites:

- https://api.jquery.com/
- https://www.w3schools.com/jquery/default.asp

BIBLIOGRAFIA

- Lindey, C. (2009). *jQuery Cookbook*. Sebastopol: O'Reilly Media.
- Sawyer McFarland, D. (2008). *JavaScript & jQuery*. Sebastopol: O'Reilly Media.