

jQuery

Apostila Básica

Escrito por: Daniel de Campos Souza

| | |
|--|------------------|
| <u>INTRODUÇÃO</u> | <u>3</u> |
| <u>CAPÍTULO 1 – POR ONDE INICIAR</u> | <u>3</u> |
| <u>CAPÍTULO 2 – MEU PRIMEIRO SCRIPT!</u> | <u>4</u> |
| EXERCÍCIOS..... | 5 |
| <u>CAPÍTULO 3 – ADICIONANDO CSS</u> | <u>6</u> |
| EXERCÍCIOS..... | 7 |
| <u>CAPÍTULO 4 – MODIFICANDO APENAS PARTE DO DOCUMENTO</u> | <u>7</u> |
| EXERCÍCIOS..... | 9 |
| <u>CAPÍTULO 5 – HIDE E SHOW</u> | <u>10</u> |
| EXERCÍCIOS..... | 10 |
| <u>CAPÍTULO 6 – FIND E EACH</u> | <u>10</u> |
| EXERCÍCIOS..... | 11 |
| <u>CAPÍTULO 7 – IF E ELSE</u> | <u>11</u> |
| EXERCÍCIO..... | 12 |

Introdução

jQuery não é uma nova linguagem de programação, antes, trata-se de uma compilação do JavaScript. Para utilizá-la, basta referenciar o arquivo `.js` do jQuery em sua página e você já pode usar esta linguagem com prazer.

Recomendo que todo o código produzido também esteja em um arquivo externo; isso diminuí o tamanho final de sua página HTML e também assegura que o código não ficará exposto, tornando as coisas mais fácies, caso você queira mudar um código que apareça em todas as suas páginas

Capítulo 1 – Por onde iniciar

Para se usar o jQuery é necessário um conhecimento básico em JavaScript e HTML. Portanto, esta apostila foi desenvolvida para aqueles leitores que possuem um conhecimento básico dessas linguagens. A partir daí, podemos começar, mas por onde?

Primeiro faça o download da versão mais recente da biblioteca jQuery (aproximadamente 96KB copie e cole a url <http://jqueryjs.googlecode.com/files/jquery-1.2.6.js> em seu navegador).

Crie um arquivo HTML, faça a referência à biblioteca jQuery (usando o mesmo comando para se fazer referência à um arquivo `.js` qualquer).

```
<script language="javascript" type="text/javascript" src="jquery.js" ></script>
```

Certifique-se de que a referência à biblioteca seja feita antes da referencia ao seu arquivo `.js`, caso contrário o navegador não irá interpretar corretamente os códigos desenvolvidos.

Agora crie um arquivo `.js` em branco, nele iremos desenvolver todas as lições.

Faça a referência a ele em sua página HTML.

Observações:

Sua biblioteca jQuery (assim como qualquer outro arquivo) pode ser salva em qualquer pasta com qualquer nome, desde que no momento da referência todos os parâmetros estejam corretos.

```
<script language="javascript" type="text/javascript" src="seu_arquivo.js" ></script>
```

Ótimo, agora você está pronto para entrar no mundo do jQuery!

Capítulo 2 – Meu primeiro script!

Primeiramente temos que ver como funcionará um script jQuery.

A sintaxe é a seguinte – pode ser digitada no documento `.js` que você criou ou entre as tags `<script>` `</script>` caso você opte por não usar um documento separado.

```
$(document).ready(function
() {
    //insira seu código
    aqui
});
```

Vamos analisar parte por parte deste código. Usa-se o *cifrão* (\$) para referir-se a modificação de um elemento (podendo ser desde uma tag até uma classe e ID CSS). Usamos os parênteses após o cifrão para identificar qual elemento será modificado. Neste caso, nos referimos ao documento como um todo, pois os códigos que serão introduzidos irão alterar o conteúdo do documento.

A função `.ready` informa ao navegador que os comandos devem ser executados quando o documento estiver pronto (navegável). Adicionamos o parentese após o `.ready` para informar ao navegador que os comandos enlaçados pelos parênteses deverão ser executados no momento em que o documento for navegável.

O uso de *function* enlaça todos os comandos que deverão ser executados. Use os colchetes e parênteses para fechar o enlace. Use ponto e vírgula (;) para separar as linhas de comando e evitar erros de sintaxe.

Esta linha inicial é de uso essencial, para não dizer obrigatório, no funcionamento do seu arquivo jQuery. Todos os comandos que usaremos no documento serão enlaçados por esta linha de comando.

Agora que entendemos como o navegador irá interpretar os comandos, vamos construir nosso primeiro script; o famoso “olá mundo”.

```
$(document).ready(function
() {
    alert(“olá mundo”);
});
```

Em suma, neste comando, estamos informando ao navegador que, assim que o documento estiver pronto, ele deve escrever na tela “olá mundo”. Esse texto será exibido em uma janela de alerta.

Ótimo! Agora vamos tentar o mesmo comando só que de uma forma um pouco diferente:

```
$(document).ready(function
() {
    $("a").click(function()
{
    alert("olá mundo");
    });
});
```

Como já foi passado o uso do cifrão informa que elemento será alterado. Nesse caso a tag (“a”).

Feito isto, construa um código HTML simples em seu arquivo HTML. Não se preocupe com formatação, apenas tenha certeza de ter feito as devidas referências aos documentos .js.

Crie o seguinte código em seu documento HTML:

```
<body>
  <p><a href="#">clique aqui</a></p>
</body>
```

Salve o documento e abra-o em seu navegador, clique no link e veja o resultado.

Agora vamos estudar o caso. Quando fizemos a referência à tag “a” e usamos o comando *.click*, informamos ao navegador que uma ação deveria ser executada; nesse caso o comando *alert* que exibirá o texto “olá mundo”.

Observações:

Nós não usamos aspas no elemento document pois o mesmo se refere à página como um todo, diferente da tag “a” que se refere apenas a ela e/ou seus elementos “filhos” (caso a tag possua algum).

Exercícios

- 1) Crie um comando que exiba uma mensagem, avisando ao usuário que ele clicou em um link.
- 2) Em uma página HTML o usuário deverá ser informado de que o link clicado está funcionando, escreva um comando que faça isso.

Capítulo 3 – adicionando CSS

Usando alguns comandos em jQuery, é possível alterar a aparência dos documentos, podemos adicionar uma classe que haja em nossos arquivos CSS, ou até mesmo adiciona o CSS de modo inline, ou seja como atributo *style* da tag.

Observação:
Atente ao correto uso das letras maiúsculas em todos os comandos, para evitar erros de sintaxe

Para isso, usaremos o comando `.addClass`.

Para testar esse comando, crie um arquivo css e referencie-o em sua página HTML, ou construa-o direto no arquivo HTML usando as tags `<style>` e `</style>`.

Heis os dados que seu CSS deve conter:

```
.teste {  
  border-width: 3px;  
  border-style: dashed;  
  border-color: red;  
}
```

Agora vamos fazer com que, ao clicarmos no link, seja possível adicionar a classe `“teste”` à tag `<p>`.

Exemplo

```
$(document).ready(function () {  
  $("a").click(function() {  
    $("p").addClass("teste");  
  });  
});
```

Agora, quando clicarmos no link, uma borda, de espessura media, pontilhada e vermelha, irá aparecer ao redor do texto.

Também podemos adicionar uma formatação CSS direto à tag, usando o comando `.css()`. Dentro dos parênteses, coloque a formatação desejada.

Exemplo

```
$(document).ready(function () {  
    $("a").click(function () {  
        $("p").css("border", "3px dashed red");  
    });  
});
```

Exercícios

- 1) Crie um comando que, quando o usuário clicar no link, o navegador adicione uma borda azul, com 4 pixels de espessura e que seja tracejada, esta formatação deve ser uma classe CSS do seu documento anexo ou que esteja entre as tags `<style></style>` do seu documento HTML.
- 2) Modifique este comando para que os atributos sejam adicionados diretamente à tag.

Capítulo 4 – Modificando apenas parte do documento

Nesse capítulo, iremos aprender como modificar elementos específicos na página. Até agora, tudo o que aprendemos modificava todas as tags nas páginas; porém, com uma modificação na sintaxe nós poderemos alterar apenas uma pequena parte do documento.

Os comandos novos que aprenderemos agora são `.addClass` e `.removeClass`, que como os próprios nomes dizem, servem para adicionar e remover uma classe CSS, respectivamente, da tag.

Lembra-se que, no começo desta apostila, foi dito que poderia-se usar o cifrão (\$) para indicar uma tag ou ID que seria alterado? Pois bem, chegou a hora de trabalhar com ID's.

Para iniciarmos, crie um documento CSS com uma classe chamada *“fundo”*. Seu conteúdo deve ser:

```
.fundo {  
    background-color: red;  
    color: white;  
}
```

Agora, em seu documento HTML, crie duas listas com marcadores, ou não classificadas (``), com, no mínimo, dois itens; dê a uma das listas o ID `lista_teste`.

Em seu documento `.js`, crie o seguinte comando:

```
$(document).ready(function () {
    $("#lista_teste").hover(function() {
        $(this).addClass("fundo");
    }, function() {
        $(this).removeClass("fundo");
    });
});
```

Com esse comando você irá modificar apenas os itens que contêm a ID `"lista_teste"`, no momento em que você passar o mouse sobre eles.

Agora, vamos aprofundar o código ainda mais. Ao invés de modificar a lista toda, que tal modificar apenas um único item dessa lista?

Para isso, no seu documento `.js`, escreva o seguinte comando:

```
$(document).ready(function () {
    $("#lista_teste li:last").hover(function() {
        $(this).addClass("fundo");
    }, function() {
        $(this).removeClass("fundo");
    });
});
```

Com isso, apenas o último item da lista irá ser modificado quando você passar o mouse por cima dele, caso o mouse seja passado no resto da lista, nenhuma alteração será feita.

Agora vamos entender o que nós escrevemos: Aprendemos um comando novo; o (`this`), ele faz a referência ao último elemento que foi alterado no script. Nesses casos, o ID `"lista_teste"` e o último item desta lista.

Agora vamos criar um botão no documento HTML. Dê a ele o ID `"botao1"`.

Atualize seu código `.js` da seguinte forma:

```
$(document).ready(function () {
    $("#botao1").click(function() {
        $("#lista_teste").addClass("fundo");
    });
});
```

Dessa forma, toda vez que o usuário clicar no botão que possua o ID `"botao1"`, a lista receberá a classe `"fundo"`. Também é possível modificar apenas o último elemento da lista; basta adicionar `"li:last"` após o ID `"lista_teste"`. O código ficará assim:


```
$(document).ready(function () {
    $("#botao1").click(function() {
        $("#lista_teste li:last").addClass("fundo");
    });
});
```

Agora vamos adicionar um botão para remover a classe *“fundo”*. Crie em seu documento HTML outro botão e atribua a ele o ID *“botão2”*, escreva o seguinte código em seu documento .js:

```
$(document).ready(function () {
    $("#botao1").click(function() {
        $("#lista_teste").addClass("fundo");
    });
    $("#botao2").click(function() {
        $("#lista_teste").removeClass("fundo");
    });
});
```

Para fazer isso apenas no último item da lista, basta adicionar *“li:last”* após o ID *“lista_teste”*, nas duas vezes em que ela é referenciada no código, ficando dessa forma:

```
$(document).ready(function () {
    $("#botao1").click(function() {
        $("#lista_teste li:last").addClass("fundo");
    });
    $("#botao2").click(function() {
        $("#lista_teste li:last").removeClass("fundo");
    });
});
```

Essas funções *.addClass* e *.removeClass* podem ser usadas para alterar qualquer tag, porém tome muito cuidado para não usar junto ao evento *.hover*, pois em alguns casos, como por exemplo textos, pode ser que o parágrafo adicione e remova a classe CSS toda a vez que o ponteiro do mouse oscilar entre o texto e o espaço em branco.

Exercícios

- 1) Construa um documento que contenha uma lista ordenada(), e, toda vez que o usuário passar o mouse sobre a lista, ela receba um fundo vermelho.
- 2) Crie um documento, com dois parágrafos, onde haja dois botões; um que adicione uma borda preta ao redor do parágrafo e outro que remova essa borda

Capítulo 5 – Hide e Show

Com jQuery, é possível criar um pouco mais de dinamismo ao site. Para isso, iremos usar os comandos `.hide` e `.show`. Como os próprios nomes indicam, esses comandos escondem e mostram o conteúdo de uma tag.

É possível determinar a velocidade com que o conteúdo será escondido / exibido; basta adicionar o parâmetro `"fast"` para rápido ou `"slow"` para devagar. É aconselhado o uso desses comandos junto a botões ou links.

Exemplo

```
$(document).ready(function () {  
    $("#botao1").click(function() {  
        $("p").hide("fast");  
    });  
    $("#botao2").click(function() {  
        $("p").show("slow");  
    });  
});
```

Este comando pode ser aplicado diretamente a uma *ID*, ao invés de tag, possibilitando um melhor uso desta função.

Exercícios

- 1) Crie um documento com dois parágrafos e dois botões, um irá esconder os parágrafos e o outro irá exibí-los.
- 2) Modifique o documento para que apenas um parágrafo seja escondido e exibido.

Capítulo 6 – Find e Each

Nós usamos o comando `.find` para pesquisar os elementos filhos das tags, ou a própria tag. Já o comando `.each` determina que, para cada elemento encontrado, o comando seguinte deve ser executado.

Exemplo

```
$(document).ready(function () {  
    $("#lista_teste").find("li").each(function() {  
        $(this).html($(this).html() + " olá");  
    });  
});
```

Esta linha de comando faz com que cada elemento definido seja alterado; alguns comandos como o `.addClass` já fazem isto internamente.

O comando `.html` busca a tag definida e faz a alteração.

Você pode usar o `.find` e `.each` para alterar qualquer tag, mas, no caso de tags que não possuam filhos (ex: `<p>`), você deve definir a busca dentro da tag `<body>`.

Exemplo

```
$(document).ready(function () {
    $("body").find("p").each(function() {
        $(this).html($(this).html() + " olá");
    });
});
```

Exercícios

- 1) Construa uma lista ordenada (``) e, para cada elemento, adicione ao final a palavra "item".
- 2) Crie dois parágrafos, e um comando que adicione, ao final de cada um, a frase: "fim do parágrafo".

Capítulo 7 – If e Else

Assim como no JavaScript e em qualquer outra linguagem programação, o uso do *if* e *else* nos ajuda a não só estender a funcionalidade, mas também as possibilidades de uso do seu código.

Podemos usar esse comando para criar novos eventos quando um outro tenha sido concluído, como por exemplo, para verificar se um formulário foi corretamente preenchido.

Exemplo:

Supondo que em um documento HTML haja um formulário, onde alguns de seus campos necessitem estar iguais, pode-se usar o seguinte código, tendo como base que são dois campos; um com *ID* "campo1" e o outro com *ID* "campo2", e o botão *submit* com *ID* "bot1":

```
$(document).ready(function() {
    $("#bot1").click(function() {
        if ($("#campo1").val() != $("#campo2").val()) {
            alert("favor preencher os campos corretamente");
        } else {
            alert("campos preenchidos corretamente");
        }
    });
});
```

Observação:
Os mesmos elementos de comparação do JavaScript são usados no jQuery, por isso se faz necessário um conhecimento básico desta linguagem para o uso do jQuery.

A sintaxe de verificação é a mesma que a do JavaScript. Para usar o *if* e o *else*, basta indicar quais os campos (ou *ID's*), e quais os dados devem ser verificados.

O comando `.val()` indica ao navegador que o conteúdo dos campos deve ser verificado. Note que para fazer a comparação do *ID's* é necessário o uso do cifrão (\$) para que o navegador não entenda isso como uma string.

Exercício

Crie um formulário onde o usuário deve preencher um campo igual ao outro, e, caso não sejam iguais, o navegador deve alertá-lo; caso sejam, deverá aparecer a mensagem de que tudo foi preenchido corretamente.

Parabéns, você acaba de concluir sua introdução ao jQuery! Estude profundamente e pesquise mais para dominar esta poderosa ferramenta.