Apostila jQuery

Conteúdo

Por onde começar com a jQuery?

Este guia é uma introdução à biblioteca **jQuery**. Ter um conhecimento sobre javascript e modelo de objeto de documento (**DOM**) são requisitos necessários. Ele começa do básico e tenta explicar os detalhes quando necessário. Ele abordará um exemplo simples alô mundo, seletores e eventos básicos, **AJAX**, efeitos e o desenvolvimento de plugins.

Este guia não contém exemplos "prontos para clicar". A intenção de prover apenas os "códigos para copiar" é um convite para que você mesmo faça os testes. Copie um exemplo, veja o que ele faz, e modifique-o.

Configuração

Para começar, nós precisamos de uma cópia da biblioteca jQuery. Ainda que a versão mais recente possa ser encontrada neste link: **http://docs.jquery.com/Downloading_jQuery**, este guia fornece um pacote básico que você poderá baixar.

Kit para começar com o jQuery: Faça o download deste arquivo e extraia o seu conteúdo. Abra o starterkit.html e o ***custom.js*** com o seu editor preferido e o ***starterkit.html*** com o navegador.

***Caso já tenha este arquivo em seu computador não há necessidade de baixá-lo.**

Agora nós temos tudo o que precisamos para iniciar o notório exemplo do **"Alô mundo".**

Links interessantes para este capítulo:

- Kit para começar
- Downloads do jQuery

Olá jQuery

Como quase tudo o que fazemos quando estamos utilizando o **jQuery** lê ou manipula um modelo de objeto de documento (DOM), precisamos nos certificar que começamos adicionado eventos etc tão logo o DOM esteja pronto.

Para fazer isso, nós registramos o evento ***ready (pronto)*** para o documento.

```
$(document).ready(function() {
	// faça alguma coisa quando o DOM
estiver pronto
});
```

Colocar um ***alert*** nesta função não faz muito sentido, pois o ***alert*** não requer que o DOM esteja carregado. Então vamos tentar algo mais sofisticado: Mostrar um ***alert*** quando clicarmos o link.

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

Dessa forma o *alert* será exibido assim que você clicar no link.

Vamos dar uma olhada no que estamos fazendo: $("a") é um seletor do jQuery, neste caso, ele seleciona todos os elementos a. O $ por si só é um *alias* para a "classe" jQuery, por outro lado o $() constrói um novo objeto jQuery. A função click() que chamamos depois é um método do objeto jQuery.

Ele liga o evento clique a todos os elementos selecionados (neste caso, um único elemento a) e executa a função fornecida quando o evento ocorre. Isto é similar ao seguinte código:

Link

A diferença é bem óbvia: Nós não precisamos escrever onclick para todo elemento. Nós temos uma separação clara de estrutura (HTML) e comportamento (JS), assim como separamos estrutura e formatação utilizando CSS.

Com isso em mente, exploramos seletores e eventos um pouco mais a fundo.

Links interessantes para este capítulo:

- Base do jQuery
- Expressões do jQuery
- Eventos Básicos do jQuery

Me encontre: Usando seletores e eventos

O jQuery provê duas maneiras de selecionar elementos. A primeira utiliza uma combinação de seletores CSS e XPath passados como uma string para o construtor do jQuery (ex. $("div > ul a")). A segunda utiliza vários métodos do objeto jQuery. Ambas podem ser combinadas.

Para testar alguns desses seletores, vamos selecionar e modificar a primeira lista ordenada no nosso kit para começar.

Primeiramente queremos selecionar a própria lista. A lista tem um ID "listaOrdenada". No javascript clássico, você pode selecioná-la usando document.getElementById("listaOrdenada"). Com o jQuery, nós fazemos isso assim:

```
$(document).ready(function() {
	$
("#listaOrdenada").addClass("vermelho");
});
```

O kit para começar fornece uma folha de estilos com a classe "vermelho" que simplesmente adiciona um fundo vermelho. No entanto, quando você recarrega a página no seu navegador, você verá que a primeira lista tem o fundo vermelho. A segunda lista permanece inalterada.

Agora vamos adicionar mais classes para os elementos filhos desta lista.

```
$(document).ready(function() {
	$("#listaOrdenada >
li").addClass("azul");
});
```

Isto seleciona todos os lis filhos do elemento com id listaOrdenada e adiciona a classe "azul".

Agora alguma coisa mais sofisticada: Nós queremos adicionar e remover a classe quando o usuário passar o mouse sobre o elemento li, mas somente no último elemento da lista.

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

Existem diversos outros seletores similires à sintaxe CSS e XPath. Mais exemplos e a lista de todas as expressões disponíveis podem ser encontrados neste link: http://docs.jquery.com/DOM/Traversing/Selectors.

Para todo evento *onxxx* disponível, como onclick, onchange, onsubmit, existem um equivalente no jQuery. Alguns outros eventos, como ready e hover, são métodos convenientes para algumas tarefas.

Você pode encontrar uma lista completa com todos os eventos suportados no http://www.visualjquery.com na seção de *Events*.

Com estes seletores e eventos você já pode fazer muita coisa, mas tem muito mais.

```
$(document).ready(function() {
	$
("#listaOrdenada").find("li").each(functi
on(i) {
		$(this).html( $(this).html() + "
BAM! " + i );
	});
   });
```

O find() permite que você faça uma pesquisa mais a fundo nos descendentes dos elementos já selecionados, apesar de $("#listaOrdenada).find("li") ser praticamente o mesmo que $ ("#listaOrdenada li"). O each() faz a iteração sobre cada elemento e permite um processamento mais profundo. A maioria dos métodos, como o addClass(), utiliza o each() internamente. Neste exemplo, o html() é utilizado para recuperar o texto de cada elemento li, adicionar algum texto a ele e definí-lo como o texto do elemento.

Uma outra tarefa que você frequentemente terá que lidar é chamar métodos em elementos DOM que não são suportados pelo jQuery. Pense em um formulário que você gostaria de resetar depois que enviou com sucesso via AJAX.

```
$(document).ready(function() {
	// use isto para resetar um único
formulário
	$("#reset").click(function() {
		$("#form")[0].reset();
	});
});
```

Este código seleciona o elemento com o ID "form" e chama o reset() no primeiro elemento selecionado. Caso exista mais de um form, você pode fazer dessa maneira:

```
$(document).ready(function() {
	// use isto para resetar todos os formulários
de uma só vez
	$("#reset").click(function() {
		$("form").each(function() {
			this.reset();
		});
	});
});
```

Isto deve selecionar todos os formulários no seu documento, fazer a iteração sobre eles e chamar o reset() para cada um.

Outro problema que você pode encontrar é não selecionar alguns elementos. O jQuery provê o filter() e o not() para isto. Enquanto o filter() reduz a seleção para os elementos que atendem à expressão de filtro, o not() faz exatamente o contrário, removendo todos os elementos que atendem a expressão. Imagine uma lista desordenada onde você quer selecionar todos os elementos li que não possuam um filho ul.

```
$(document).ready(function() {
	$("li").not("[ul]").css("border", "1px
solid black");
});
```

Isto seleciona todos os elementos li e remove todos os elementos da seleção que possuam um elemento ul como filho. Sendo assim todos os elementos li ficarão com uma borda, com exceção daqueles que possuam um filho ul. A sintaxe [expressão] é vinda do XPath e pode ser utilizada para filtrar elementos e atributos filhos. Talvez você queira selecionar todas as âncoras que possuam o attributo name:

```
$(document).ready(function() {
	$
("a[@name]").background("#eee");
});
```

Isto adiciona uma cor de fundo para todos os elementos âncora com o atributo name.

Mais comum que selecionar as âncoras pelo nome, você pode precisar selecionar as âncoras pelo atributo "href". Isto pode ser um problema uma vez que os navegadores se comportam inconsistentemente quando retornam o que eles pensam que o valor do "href" é. Para selecionar apenas uma parte do value, podemos utilizar o seletor contém "*=" ao invés do igual ("="):

```
$(document).ready(function() {
	$("a[@href*=/content/gallery]").click(function() {
		// faça alguma coisa com todos os links que apontam para algum lugar
em /content/gallery
	});
});
```

Até agora, usamos todos os seletores para selecionar os filhos ou filtrar a seleção atual. Existem situações onde você irá precisar selecionar os anteriores ou próximos elementos, conhecidos como *siblings* (filhos do mesmo pai). Pense em uma página de um FAQ, onde todas as respostas estão escondidas em um primeiro momento e são exibidas quando a questão é clicada. O código do jQuery para isso:

```
$(document).ready(function() {
	$
('#faq').find('dd').hide().end().find('dt').click(function
() {
        var resposta = $(this).next();
        if (resposta.is(':visible')) {
            resposta.slideUp();
        } else {
            resposta.slideDown();
        }
    });
});
```

Aqui estamos usando um pouco de encadeamento para reduzir o tamanho do código e ganhar performance, uma vez que '#faq' é selecionada apenas uma única vez. Usando o end(), o

primeiro find() é desfeito, assim podemos começar a procurar com o próximo find() no nosso elemento #faq, ao invés de procurar no filho dd.

Com o acionamento do evento click, a função passada ao método click(), utilizamos $(this).next() para encontrar o próximo *sibling* a partir do dt atual. Isto nos permite selecionar rapidamente a resposta seguinte à questão clicada.

Além dos *siblings*, você também pode selecionar os elementos pais (também conhecidos como ancestrais para os mais familiarizados com o XPath). Talvez você pode querer realçar o parágrafo que é o pai do link que o usuário passar o mouse. Tente isso:

```
$(document).ready(function() {
	$("a").hover(function() {
	   $
(this).parents("p").addClass("realcar");
	}, function() {
	   $
(this).parents("p").removeClass("realca
r");
	});
   });
```

Para todos os elementos âncoras que o usuário passar o mouse, o parágrafo pai é procurado e a classe "realcar" é adicionada e removida.

Vamos voltar um passo atrás de continuarmos: o jQuery faz com que o código fique menor e tornando-o mais fácil de ler e dar manutenção. O código seguinte é um atalho para a notação $(document).ready(callback):

```
$(function() {
	// código a ser executado quando DOM
está pronto
});

Aplicado ao exemplo do Alô Mundo!, ficaria
assim:

$(function() {
	$("a").click(function() {
		alert("Alô Mundo!");
	});
});
```

Agora com o básico em mãos, queremos explorar outros aspectos, começando com o AJAX.

Links interessantes para este capítulo:

- Documentação da API do jQuery
- Visual jQuery - Uma documentação categorizada e navegável da API do jQuery
- Expressões do jQuery: CSS
- Expressões do jQuery: XPath
- Expressões do jQuery: Customizadas
- Eventos especiais no jQuery
- jQuery DOM Traversing

Me avalie: Usando AJAX

Página 7 – Apostila jQuery

Nesta parte nós escreveremos uma pequena aplicação AJAX que permite que o usuário avalie alguma coisa, assim como é feito no youtube.com.

Precisamos de um pouco de código do servidor para isso. Meu exemplo utiliza um arquivo php que lê o parâmetro "avaliacao" e retorna o número de avaliações e a média de avaliação. Dê uma olhada no rate.php para o código server-side.

Nós não queremos que este exemplo funcione sem AJAX, mesmo que isto seja possível, então nós geramos a marcação necessária com o jQuery e a adicionamos à div com o ID "avaliacao".

```
$(document).ready(function() {
	// gera a marcação
	var ratingMarkup = ["Por favor avalie: "];
	for(var i=1; i <= 5; i++) {
		ratingMarkup[ratingMarkup.length] = "<a href='#'>" + i
+ "</a> ";
	}
	// adiciona a marcação ao container e aplica o acionador de click
às âncoras
	$
("#avaliacao").append( ratingMarkup.join('') ).container.find("a").click(fun
ction(e) {
		e.preventDefault();
		// envia a requisição
		$.post("rate.php", {avaliacao: $(this).html()},
function(xml) {
			// formata o resultado
			var resultado = [
				"Obrigado por avaliar, média atual: ",
				$("media", xml).text(),
				", número de votos: ",
				$("contador", xml).text()
			];
			// saída do resultado
			$("#avaliacao").html(resultado.join(''));
		} );
	});
});
```

Este pedaço de código gera cinco elementos âncoras adicionando-os ao elemento container com o id "avaliacao". Depois disso, para cada âncora no container, um observador do click é adicionado. Quando a âncora é clicada, uma requisição post é enviada para o rate.php com o conteúdo da âncora como parâmetro. O resultado retornado como um XML é então adicionado ao container, substituindo as âncoras.

Se você não possui um servidor web com o PHP instalado em mãos, você pode dar uma olhada no exemplo on-line.

Para ver um exemplo muito bom de um sistema de avaliação que funciona até mesmo sem javascript, visite softonic.de e clique em "Kurz bewerten!"

Mais documentação sobre os métodos AJAX do jQuery pode ser encontrada aqui ou no Visual jQuery na categoria AJAX.

Um problema muito comum encontrado quando se carrega um conteúdo por AJAX é o seguinte: Quando se adiciona controladores de evento ao seu documento que deveriam ser aplicados também ao conteúdo carregado, você deve aplicar estes controladores depois que o conteúdo é carregado. Para prevenir a código duplicado, você pode delegar a uma função. Exemplo:

```
// Vamos usar o atalho
$(function() {
	var addClickHandlers = function() {
		$
("a.cliqueParaCarregarConteudo").click(function() {
			$("#target").load(this.href,
addClickHandlers);
		});
	};
	addClickHandlers();
});
```

Agora a função addClickHandlers é aplicada uma vez quando o DOM está pronto e depois toda vez que o usuário clicar um link com a classe cliqueParaCarregarConteudo e o conteúdo tiver terminado de ser carregado.

Por favor observe que a função "addClickHandlers" é definida como uma variável local, ao invés de uma função global (como function addClickHandlers() {...}). Procure seguir esta prática para prevenir conflito com outras variáveis ou funções globais.

Com AJAX podemos explorar muita coisa "Web 2.0". Mas até agora estamos devendo efeitos bacanas.

Links interessantes para este capítulo:

- Módulo AJAX do jQuery
- API do jQuery: Contém descrição e exemplos para adicionar e todos os outros métodos do jQuery
- ThickBox: Um plugin para o jQuery que o utiliza para aprimorar o famoso lightbox

Me anime: Usando Efeitos

Animações simples com o jQuery podem ser feitas com o show() e o hide()

```
$(document).ready(function() {
	$("a").toggle(function() {
		$
(".algumacoisa").hide('slow');
	}, function() {
		$
(".algumacoisa").show('fast');
	});
});
```

Você pode criar qualquer combinação das animações com o animate() como por exemplo um *slide* com fade:

```
$(document).ready(function() {
	$("a").toggle(function() {
		$
(".algumacoisa").animate({
			height:
'hide',
			opacity:
'hide'
		}, 'slow');
	}, function() {
```

```
                $
("algumacoisa").animate({
                        height:
'show',
                        opacity:
'show'
                }, 'slow');
        });
});
```

Efeitos muito mais sofisticados podem ser feitos com a coleção de plugins Interface (http://interface.eyecon.ro/). No site você encontrará demonstrações e a documentação.

Além da coleção Interface, que está no topo da lista de plugins do jQuery, existem muitos outros. O próximo capítulo lhe mostrará como usar o plugin *tablesorter*.

Links interessantes para este capítulo:

- Módulo de efeitos do jQuery
- Plugin Interface

Me ordene: Usando o plugin tablesorter

O plugin tablesorter permite que você ordene as tabelas no lado do cliente. Você inclui o jQuery e o pluguin e informa ao pluguin quais tabelas deseja ordernar.

Para experimentar este exemplo, adicione esta linha ao starterkit.html (abaixo da inclusão do jquery):

<script src="lib/jquery.tablesorter.js" type="text/javascript"></script>

Depois de incluir o plugin, você poderá chamá-lo assim:

```
$
(document).ready(function()
{
        $
("#large").tableSorter();
});
```

Tente clicar nos cabeçalhos da tabela e veja se ela fica ordenada em ordem ascendente no primeiro clique e descendente no segundo.

A tabela deve utilizar algum realce nas linhas, podemos adicioná-los passando algumas opções:

```
$(document).ready(function() {
        $("#large").tableSorter({
                stripingRowClass: ['odd','even'],     // Nome da classe a ser utilizada para a divisão
das linhas como um array.
                stripRowsOnStartUp: true              // Divide as linhas quando o tableSorter iniciar.
        });
});
```

Existem mais exemplos e documentação sobre as opções disponíveis no site do tablesorter (www.motherrussia.polyester.se/jquery-plugins/tablesorter/).

A maioria dos plugins pode ser utilizada como este: Inclua o arquivo do plugin e chame o método do plugin em alguns elementos, passando os parâmetros opcionais para customizar o plugin.

Uma lista atualizada dos plugins disponíveis pode ser encontrada no site do jQuery (www.jquery.com/plugins).

Quando você utilizar o jQuery com mais freqüência, você perceberá que será melhor empacotar o seu próprio código como um plugin, seja para o reuso para você mesmo ou para a sua empresa, ou para compartilhá-lo com a comunidade. O próximo capítulo dá algumas dicas sobre como estruturar um plugin.

Links interessantes para este capítulo:

- Plugins para o jQuery
- Tablesorter Plugin

Plugue-me: Escrevendo seus próprios plugins

Escrever seus próprios plugins para o jQuery é muito fácil. Se você seguir as regras abaixo, será fácil para outros integrarem o seu plugin também.

1. *Encontre um nome para o seu plugin, vamos chamar o nosso exemplo "foobar".*
2. *Crie um arquivo chamado jquery.[nomedoseuplugin].js, ex. jquery.foobar.js*
3. *Crie um ou mais métodos no plugin extendendo o objeto jQuery, ex.:*

```
jQuery.fn.foobar = function() {
  // faça alguma coisa
};
```

6. *Opcional: Crie um objeto com funções de ajuda, ex.:*

```
jQuery.fooBar = {
  height: 5,
  calculateBar = function() { ... },
  checkDependencies = function() { ... }
};
```

Você poderá então chamar essa função de ajuda pelo seu plugin:

```
jQuery.fn.foobar = function() {
  // faça alguma coisa
  jQuery.foobar.checkDependencies(value);
  // faça alguma outra coisa
};
```

```
11.Opcional: Crie configurações padrões que possam ser alteradas pelo usuário, ex.:
12.jQuery.fn.foobar = function(options) {
13.             var settings = {
14.             value: 5,
15.             name: "pete",
16.             bar: 655
17.   };
18.   if(options) {
19.             jQuery.extend(settings, options);
20.   }
    };
```

Você pode então chamar o plugin sem opções, usando as configurações padrões:

$("...").foobar();

Ou com algumas opções:

```
$
("...").foobar({
        value:
        123,
        bar: 9
});
```

Se você lançar o seu plugin, você deverá prover alguns exemplos e uma documentação. Existem diversos plugins disponíveis como ótimas referências.

Agora você já deve ter uma idéia básica de como escrever um plugin. Vamos utilizar este conhecimento e escrever nosso próprio plugin.

Uma coisa que muita gente, tentando manipular formulários com o jQuery, pede, é marcar e desmarcar radio buttons ou checkboxes. Eles acabam com um código como este:

```
$
("input[@type='checkbox']").each(fu
nction() {
          this.checked = true;
          // ou, para desmarcar
          this.checked = false;
          // ou, para inverter
          this.checked = !
this.checked;
});
```

Sempre que você possuir um *each* no seu código, você pode querer reescrever isto com um plugin, quase sem alterações:

```
$.fn.check = function() {
          return
this.each(function() {
                    this.checked =
true;
          });
};
```

Este plugin agora pode ser utilizado:

$("input[@type='checkbox']").check();

Agova você pode escrever plugins para as duas funções uncheck() e toggleCheck() também. Mas ao contrário nós estendemos nosso plugin para aceitar algumas opções.

```
$.fn.check = function(modo) {
	var modo = modo || 'on'; // se modo não está definido, use 'on' como padrão
	return this.each(function() {
		switch(modo) {
		case 'on':
			this.checked = true;
			break;
		case 'off':
			this.checked = false;
			break;
		case 'toggle':
			this.checked = !this.checked;
			break;
		}
	});
};
```

Definindo um valor padrão para as opções, permite que o usuário omita a opção ou passe "on", "off", e "toggle", ex.:

$("input[@type='checkbox']").check();
$("input[@type='checkbox']").check('on');
$("input[@type='checkbox']").check('off');
$("input[@type='checkbox']").check('toggle');

Com mais de uma configuração opcional, este método torna-se complicado, pois o usuário deverá passar valores nulos se quiser omitir o primeiro parâmetro e usar somente o segundo.

O uso do *tablesorter* no último capítulo demonstra o uso de um objeto literal também resolve este problema. O usuário omite todos os parâmetros ou passa um objeto com um par chave/valor para cada configuração que deseja sobrescrever.

Como um exercício, você poderia tentar reescrever o código do quarto capítulo como um plugin. O esqueleto do plugin deve ser similar a isto:

```
$.fn.rateMe = function(options) {
	var container = this; // ao invés de selecionar um container estático com  $("#rating"), nós agora utilizamos o contexto do jQuery

	var settings = {
		url: "rate.php"
		// coloque mais padrões aqui
		// lembre-se de colocar uma vírgula (",") depois de cada par, mas não depois do último!
	};

	if(options) { // verifica se as opções estão presentes antes de estender as configurações
		$.extend(settings, options);
	}

	// ...
```

```
        // resto do código
        // ...

        return this; // se possível, retorne "this" para não quebrar a corrente
});
```

Próximos passos

Se você tem interesse em desenvolver mais em javascript, você deve dar uma olhada numa extensão do Firefox chamada FireBug. Ela provê um console (ótimo para substituir os alerts), um debugger e outras coisas úteis para o seu desenvolvimento diário em javascript.

Se você tem problemas que não consegue resolver, idéias que gostaria de compartilhar ou apenas necessidade de expressar sua opinião sobre o jQuery, sinta-se à vontade para postar na lista de discussão do jQuery (www.jquery.com/discuss).