O Básico de jQuery

$(document).ready()

Você não pode manipular a página com segurança até o documento estar "pronto" (ready). O jQuery detecta o estado de prontidão para você; o código incluido dentro de `$(document).ready()` somente irá rodar depois que a página estiver pronta executar o código JavaScript.

Um bloco $(document).ready()

```
1  $(document).ready(function() {
2      console.log('pronto!');
3  });
```

Há um atalho para `$(document).ready()` que você verá algumas vezes; entretando, eu não recomendo usá-lo se você estiver escrevendo código que pessoas que não têm experiência com jQuery poderá ver.

Atalho para $(document).ready()

```
1  $(function() {
2      console.log('pronto!');
3  });
```

Você ainda pode passar uma função nomeada para `$(document).ready()` ao invés de passar uma função anônima.

Passando uma função nomeada ao invés de uma anônima

```
1  function readyFn()
2  {
3      // código para executar quando o documento estiver pronto
4  }
5  $(document).ready(readyFn);
```

Selecionando elementos

O conceito mais básico do jQuery é "selecionar alguns elementos e fazer alguma coisa com eles." O jQuery suporta a maioria dos seletores CSS3, assim como alguns seletores não-padrão. Para uma referência completa de seletores, visite http://api.jquery.com/category/selectors/.

A seguir, alguns exemplos de técnicas comuns de seleção.

Selecionando elementos por ID

```
1  $('#myId'); // lembre-se que os IDs devem ser únicos por página
```

Selecionando elementos pelo nome da classe

```
1  $('div.myClass');   // há um aumento de performance se você
2                      // especificar o tipo de elemento.
```

Selecionando elementos por atributo

```
1  $('input[name=first_name]'); // cuidado, isto pode ser muito lento
```

Selecionando elementos através da composição de seletores CSS

```
1  $('#contents ul.people li');
```

Pseudo-seletores

```
1  $('a.external:first');
2  $('tr:odd');
3  $('#myForm :input'); // selecione todos os elementos input num formulário
4  $('div:visible');
5  $('div:gt(2)');      // todos exceto as três primeiras divs
6  $('div:animated');   // todas as divs com animação
```

Quando você usa os pseudos-seletores `:visible` e `:hidden`, o jQuery testa a visibilidade atual do elemento, não os atributos `visibility` ou `display` do CSS - ou seja, ele olha se a altura e a largura física do elemento na página são ambas maiores que zero. No entanto, este teste não funciona com elementos `<tr>`; neste caso, o jQuery faz a veri cação da propriedade display do CSS, e considera um elemento como oculto se sua propriedade `display` for `none`. Elementos que não forem adicionados no DOM serão sempre considerados ocultos, mesmo que o CSS que os afetam torná-los visíveis. (Consulte a seção de Manipulação mais adiante neste capítulo para aprender como criar e adicionar elementos ao DOM).

Para referência, aqui está o código que jQuery usa para determinar se um elemento é visível ou oculto, com os comentários adicionados para maior

esclarecimento:

```
jQuery.expr.filters.hidden = function( elem )
{
    var width = elem.offsetWidth, height = elem.offsetHeight,
        skip = elem.nodeName.toLowerCase() === "tr";

    // o elemento tem 0 de altura e 0 de largura e
    // não é uma <tr>?
    return width === 0 && height === 0 && !skip ?

        // então deve estar escondido
        true :

        // mas se o elemento tiver largura e altura
        // e não for uma <tr>
        width > 0 && height > 0 && !skip ?

            // então deve estar visível
            false :

            // se chamos aqui, o elemento tem largura
            // e altura, mas também é uma <tr>,
            // então verifica a propriedade display para
            // decidir se ele esta escondido
            jQuery.curCSS(elem, "display") === "none";
};
jQuery.expr.filters.visible = function( elem )
{
    return !jQuery.expr.filters.hidden( elem );
};
```

Escolhendo seletores

Escolher bons seletores é uma forma de melhorar a performance do seu JavaScript. Uma pequena especi cidade - por exemplo, incluir um elemento como `div` quando selecionar elementos pelo nome da classe - pode ir por um longo caminho. Geralmente, sempre que você puder dar ao jQuery alguma dica sobre onde ele pode esperar encontrar o que você estiver procurando, você deve dar. Por outro lado, muita especi cidade pode não ser muito bom. Um seletor como `#myTable thead tr th.special` é um desperdício se um seletor como `#myTable th.special` lhe dará o que você precisa.

O jQuery oferece muitos seletores baseados em atributo, permitindo que você selecione elementos baseados no conteúdo de atributos arbitrários usando expressões regulares simpli cadas.

```
// encontre todos elementos <a>s em que o atributo
// rel termine com "thinger"
$("a[rel$='thinger']");
```

Se por um lado estes seletores podem ser bem úteis, eles também podem ser extremamente lerdos - Uma vez eu escrevi um seletor baseado em atributos que travou minha página por múltiplos segundos. Sempre que possível, faça suas seleções usando IDs, nomes de classe e nomes de tags.

Quer saber mais? Paul Irish tem uma excelente apresentação sobre melhorar a performance do JavaScript, com vários slides focando especi camente em performace de seletores.

Minha seleção contém algum elemento?

Uma vez que você fez uma seleção, você irá querer saber se há algo para trabalhar com ela. Você talvez se sinta tentado a fazer algo assim:

```
1  if ($('div.foo'))
2  {
3      ...
4  }
```

Isso não irá funcionar. Quando você faz uma seleção usando `$()`, um objeto é sempre retornado, e objetos sempre são tratados como `true`. Mesmo se sua seleção não tiver nenhum elemento, o código dentro do `if` vai executar do mesmo jeito.

Ao invés disso, você precisa testar a propriedade length da seleção, que diz a você quantos elementos foram selecionados. Se a resposta for 0, a propriedade length será interpretada como falso quando usada como um valor booleano.

Testando se uma seleção contém elementos.

```
1  if ($('div.foo').length)
2  {
3      ...
4  }
```

Salvando seleções

Toda vez que você faz uma seleção, um monte de código é executado, e o jQuery não faz caching de seleções para você. Se você fez uma seleção que você talvez precise fazer novamente, você deve salvar a seleção numa variável ao invés de fazer a seleção várias vezes.

Armazenando seleções em variáveis

```
1  var $divs = $('div');
```

> *No , o nome da variável começa com um sinal de dólar. Ao invés de outras linguagens, não há nada especial sobre o sinal de dólar em JavaScript -- é apenas outro caracter. Nós o usamos para indicar que a varíavel contém um objeto jQuery. Esta prática -- um tipo de Notação Húngara -- é meramente uma convenção, e não é obrigatória.*

Uma vez que você armazenou sua seleção, você pode chamar os métodos do jQuery na

variável que você armazenou, da mesma forma que você faria na seleção original.

> *Uma seleção somente obtém os elementos que estão na página quando você faz a seleção. Se você adicionar elementos na página depois, você terá que repetir a seleção ou então adicioná-la à seleção armazenada na variável. Seleções armazenadas não atualizam automaticamente quando o DOM muda.*

Re nando & Filtrando Seleções

Algumas vezes você tem uma seleção que contém mais do que você quer; neste caso, você talvez queira re nar sua seleção. O jQuery oferece vários métodos para você obter exatamente o que precisa.

Re nando seleções

```
1  $('div.foo').has('p');        // o elemento div.foo que contém <p&gt`s
2  $('h1').not('.bar');           // elementos h1 que não têm a classe bar
3  $('ul li').filter('.current'); // itens de listas não-ordenadas com a classe current
4  $('ul li').first();            // somente o primeiro item da lista não ordenada
5  $('ul li').eq(5);              // o sexto item da lista
```

Seletores relacionados à formulários

O jQuery oferece vários pseudo-seletores que lhe ajudam a encontrar elementos nos seus formulários; estes são especialmente úteis porque pode ser difícil distinguir entre elementos form baseados no seu estado ou tipo usando seletores CSS padrão.

:button
: Seleciona elementos do tipo `<button>` e elementos com `type="button"`

:checkbox
: Seleciona inputs com `type="checkbox"`

:checked
: Seleciona inputs selecionados

:disabled
: Seleciona elementos de formulário desabilitados

:enabled
: Seleciona elementos de formulário habilitados

: le
: Seleciona inputs com `type="file"`

:image
: Seleciona inputs com `type="image"`

:input

Seleciona `<input>`, `<textarea>`, e elementos `<select>`

:password
Seleciona inputs com `type="password"`

:radio
Seleciona inputs com `type="radio"`

:reset
Seleciona inputs com `type="reset"`

:selected
Seleciona inputs que estão selecionados

:submit
Seleciona inputs com `type="submit"`

:text
Seleciona inputs com `type="text"`

Usando pseudo-seletores relacionados à formulários

```
1  $('#myForm :input'); // obtém todos os elementos que aceitam entrada de dados
```

Trabalhando com seleções

Uma vez que você tem uma seleção, você pode chamar métodos nela. Métodos geralmente vêm em duas formas diferentes: getters e setters. Getters retornam uma propriedade do primeiro elemento selecionado; setters ajustam (setam) uma propriedade em todos os elementos selecionados

Encadeamento

Se você chamar um método numa seleção e este método retornar um objeto jQuery, você pode continuar a chamar métodos do jQuery sem precisar pausar com um ponto-e-vírgula.

Encadeamento

```
1  $('#content').find('h3').eq(2).html('o novo texto do terceiro h3!');
```

Se você estiver escrevendo uma cadeia que inclui vários passos, você (e a pessoa que virá depois de você) talvez ache seu código mais legível se você quebrar o código em várias linhas.

Formatando código encadeado

```
1  $('#content')
2      .find('h3')
3      .eq(2)
4      .html('novo texto do terceiro h3!');
```

Se você mudar sua seleção no meio de uma cadeia, o jQuery provê o método `$.fn.end` para você voltar para sua seleção original.

Restaurando sua seleção original usando `$.fn.end`

```
1  $('#content')
2      .find('h3')
3      .eq(2)
4          .html('novo texto para o terceiro h3!')
5      .end() // restaura a seleção para todos h3 em #context
6      .eq(0)
7          .html('novo texto para o primeiro h3!');
```

> *Encadeamento é um recurso extraordinariamente poderoso, e muitas bibliotecas adotaram-no desde que o jQuery o tornou popular. Entretando, deve ser usado com cuidado. Encadeamentos extensos podem deixar o código extremamente difícil de debugar ou modi car. Não há uma regra que diz o quão grande uma cadeia deve ser -- mas saiba que é fácil fazer bagunça*

Getters & Setters

O jQuery "sobrecarrega" seus métodos, então o método usado para setar um valor geralmente tem o mesmo nome do método usado para obter um valor. Quando um método é usado para setar um valor, ele é chamado de setter. Quando um método é usado para pegar (ou ler) um valor, ele é chamado de getter. Os setters afetam todos os elementos na seleção; getters obtêm o valor requisitado somente do primeiro elemento na seleção.

O método `$.fn.html` usado como setter

```
1  $('h1').html('olá mundo');
```

O método html usado como getter

```
1  $('h1').html();
```

Os setters retornam um objeto jQuery, permitindo que você continue chamando métodos jQuery na sua seleção; getters retornam o que eles foram pedidos para retornar, o que signi ca que você não pode continuar chamando métodos jQuery no valor retornado pelo getter.

CSS, Styling, & Dimensões

O jQuery possui uma forma bem prática para pegar e setar propriedades CSS dos elementos.

> *Propriedades CSS que normalmente incluem um hífen, precisam ser acessadas no estilo camel case em JavaScript. Por exemplo, a propriedade CSS `font-size` é expressada como `fontSize` em JavaScript.*

Pegando propriedades CSS

```
1  $('h1').css('fontSize'); // retorna uma string, como "19px"
```

Setando propriedades CSS

```
1  $('h1').css('fontSize', '100px'); // setando uma propriedade individual
2  $('h1').css({ 'fontSize' : '100px', 'color' : 'red' });
3  // setando múltiplas propriedades
```

Note o estilo do argumento que usamos na segunda linha -- é um objeto que contém múltiplas propriedades. Este é um jeito comum de passar múltiplos argumentos para uma função, e muitos setters do jQuery aceitam objetos para setar múltiplos valores de uma só vez.

Usando classes do CSS para estilos

Como um getter, o método `$.fn.css` é útil; Entretanto, ele geralmente deve ser evitado como um setter em código de produção, pois você não quer informação de apresentação no seu JavaScript. Ao invés disso, escreva regras CSS para classes que descrevam os vários estados visuais, e então mude a classe no elemento que você quer afetar.

Trabalhando com classes

```
1  var $h1 = $('h1');
2
3  $h1.addClass('big');
4  $h1.removeClass('big');
5  $h1.toggleClass('big');
6
7  if ($h1.hasClass('big'))
8  {
9      ...
10 }
```

Classes também podem ser úteis para armazenar informações de estado de um elemento, como indicar se um elemento está selecionado, por exemplo.

Dimensões

O jQuery oferece uma variedade de métodos para obter e modi car informações sobre dimensões e posições de um elemento.

O código do é somente uma introdução muito curta sobre as funcionalidades de dimensões do jQuery; para detalhes completos sobre os métodos de dimensão do jQuery, visite http://api.jquery.com/category/dimensions/.

Métodos básicos de dimensões

```
1  $('h1').width('50px');  // seta a largura de todos os elementos h1
2  $('h1').width();        // obtém a largura do primeiro h1
3
4  $('h1').height('50px'); // seta a altura de todos os elementos h1
5  $('h1').height();       // obtém a altura do primeiro h1
6
7  $('h1').position();     // retorna um objeto contendo informações
8                          // sobre a posição do primeiro h1 relativo
9                          // a seu pai
```

Atributos

Atributos de elementos podem conter informações úteis para sua aplicação, então é importante saber como setá-los e obtê-los.

O método `$.fn.attr` atua como getter e setter. Assim como o método `$.fn.css`, `$.fn.attr` atuando como setter, pode aceitar tanto uma chave e um valor ou um objeto contendo um ou mais pares chave/valor.

De nindo atributos

```
1  $('a').attr('href', 'todosMeusHrefsSaoOMesmoAgora.html');
2  $('a').attr({
3     'title' : 'todos os títulos são os mesmos também!',
4     'href' : 'algoNovo.html'
5  });
```

Agora, nós quebramos o objeto em múltiplas linhas. Lembre-se, espaços não importam em JavaScript, então você deve se sentir livre para usá-lo do jeito que quiser para fazer seu código car mais legível! Depois, você pode usar uma ferramenta de mini cação para remover espaços desnecessários para seu código de produção.

Obtendo atributos

```
$('a').attr('href');
// retorna o href do primeiro elemento <a> do documento
```

Travessia

Uma vez que você tem uma seleção do jQuery, você pode encontrar outros elementos usando sua seleção como ponto de início.

Para documentação completa dos métodos de travessia do jQuery, visite http://api.jquery.com/category/traversing/.

> *Seja cuidadoso com travessias de longas distâncias nos seus documentos -- travessias complexas torna imperativo que a estrutura do seu documento permaneça a mesma, uma di culdade à estabilidade, mesmo se você for o responsável por criar toda a aplicação desde o servidor até o cliente. Travessias de um ou dois passos são legais, mas geralmente você irá querer evitar travessias que levem você de um container para outro.*

Movendo pelo DOM usando métodos de travessia.

```
$('h1').next('p');
$('div:visible').parent();
$('input[name=first_name]').closest('form');
$('#myList').children();
$('li.selected').siblings();
```

Você também pode iterar sobre uma seleção usando `$.fn.each`. Este método itera sobre todos os elementos numa seleção e executa uma função para cada um. A função recebe como argumento um índice com o elemento atual e com o próprio elemento DOM. Dentro da função, o elemento DOM também está disponível como `this` por padrão.

Iterando sobre uma seleção

```
$('#myList li').each(function(idx, el) {
    console.log(
        'O elemento ' + idx +
        'tem o seguinte html: ' +
        $(el).html()
    );
});
```

Manipulando elementos

Uma vez que você fez uma seleção, a diversão começa. Você pode mudar, mover, remover e clonar elementos. Você ainda pode criar novos elementos através de uma sintaxe simples

Para uma referência completa dos métodos de manipulação do jQuery, visite http://api.jquery.com/category/manipulation/.

Obtendo e De nindo informações sobre elementos

Há um certo número de formas que você pode mudar um elemento existente. Dentre as tarefas mais comuns que você irá fazer é mudar o HTML interno ou o atributo de um elemento. O jQuery oferece métodos simples e cross-navegador para estes tipos de manipulação. Você ainda pode obter informações sobre elementos usando muitos dos mesmos métodos nas suas formas de getter. Nós veremos exemplos destes métodos durante esta seção, mas especi camente, aqui vão alguns poucos métodos que você pode usar para obter e de nir informação sobre elementos.

> *Mudar coisas em elementos é trivial, mas lembre-se que a mudança irá afetar todos os elementos na seleção, então se você quiser mudar um elemento, esteja certo de especi cá-lo em sua seleção antes de chamar o método setter*

> *Quando os métodos atuam como getters, eles geralmente só trabalham no primeiro elemento da seleção e eles não retornam um objeto jQuery, portanto você não pode encadear métodos adicionais a eles. Uma exceção notável é `$.fn.text`; como mencionado acima, ele obtém o texto para todos os elementos da seleção*

$.fn.html
: Obtém ou seta o conteúdo html.

$.fn.text
: Obtém ou seta os conteúdos de texto; HTML será removido.

$.fn.attr
: Obtém ou seta o valor do atributo fornecido.

$.fn.width
: Obtém ou seta a largura em pixels do primeiro elemento na seleção como um inteiro.

$.fn.height
: Obtém ou seta a altura em pixels do primeiro elemento na seleção.

$.fn.position
: Obtém um objeto com a informação de posição do primeiro elemento na seleção, relativo a seu primeiro ancestral (pai). Este é somente um getter.

$.fn.val
: Obtém ou seta o valor de elementos de formulários.

Mudando o HTML de um elemento.

```
1  $('#myDiv p:first') .html('Primeiro parágrafo <strong>novo</strong>!');
```

Movendo, copiando e removendo elementos.

Há uma variedade de formas de mover elementos pelo DOM; geralmente, há duas abordagens:

- Coloque o(s) elemento(s) selecionado(s) relativo à outro elemento
- Coloque um elemento relativo ao(s) elemento(s) selecionado(s)

Por exemplo, o jQuery fornece `$.fn.insertAfter` e `$.fn.after`. O método `$.fn.insertAfter` coloca o(s) elemento(s) selecionado(s) depois do elemento que você passou como argumento; o método `$.fn.after` coloca o elemento passado como argumento depois do elemento selecionado. Vários outros métodos seguem este padrão: `$.fn.insertBefore` e `$.fn.before`; `$.fn.appendTo` e `$.fn.append`; e `$.fn.prependTo` e `$.fn.prepend`.

O método que faz mais sentido para você dependerá de quais elementos você já selecionou e quais você precisará armazenar uma referência para os elementos que você está adicionando na página. Se você precisar armazenar uma referência, você sempre irá querer fazer pela primeira forma -- colocando os elementos selecionados relativos à outro elemento -- de forma que ele retorne o(s) elemento(s) que você está colocando. Neste caso, os métodos `$.fn.insertAfter`, `$.fn.insertBefore`, `$.fn.appendTo`, e `$.fn.prependTo` serão suas ferramentas para escolha.

Movendo elementos usando outras formas

```
1  // faz o primeiro item da lista se tornar o último
2  var $li = $('#myList li:first').appendTo('#myList');
3
4  // outra forma de resolver o mesmo problema
5  $('#myList').append($('#myList li:first'));
6
7  // perceba que não tem como acessar o item
8  // da lista que movemos, pois ele retorna
9  // a própria lista
```

Clonando elementos

Quando você usa métodos como $.fn.appendTo, você está movendo o elemento; porém, algumas vezes você irá querer fazer uma cópia do elemento. Neste caso, você precisará usar $.fn.clone primeiro.

Exemplo: Fazendo uma cópia de um elemento

```
1  // copia o primeiro item da lista para o fim
2  $('#myList li:first').clone().appendTo('#myList');
```

> *Se você precisa copiar dados e eventos relacionados, esteja certo de passar `true` como um argumento para `$.fn.clone`.*

Removendo elementos

Há duas formas de remover elementos da página: `$.fn.remove` e `$.fn.detach`. Você irá usar `$.fn.remove` quando você quiser remover a seleção permanentemente da página; enquanto o método retorna os elementos removidos, estes elementos não terão seus eventos e dados associados a ele se você retorná-los à página.

Se você precisa que os dados e eventos persistam, você irá usar `$.fn.detach`. Da mesma forma que `$.fn.remove`, ele retorna uma seleção, mas também mantém os dados e os eventos associados com a seleção para que você possa restaurar a seleção para a página no futuro.

> *O método `$.fn.detach` é extremamente útil se você estiver fazendo uma manipulação pesada à um elemento. Neste caso, é bom aplicar um `$.fn.detach` no elemento da página, trabalhar no seu próprio código, e então restaurá-lo à página quando você terminar. Isto evita que você faça "toques ao DOM" caros enquanto mantém os dados e eventos do elemento.*

Se você quer deixar um elemento na página mas simplesmente quer remover seu conteúdo, você pode usar `$.fn.empty` para retirar o HTML interno do elemento.

Criando novos elementos

O jQuery oferece uma forma elegante e trivial para criar novos elementos usando o mesmo método `$()` que você usava para seleções.

Criando novos elementos

```
1  $('<p>Este é um novo parágrafo</p>');
2  $('<li class="new">novo item de lista</li>');
```

Criando um novo elemento com um objeto atributo

```
1  $('<a/>', {
```

```
      html : 'Este é um link <strong>new</strong>',
     'class' : 'new',
     href : 'foo.html'
});
```

Perceba que no objeto de atributos nós incluimos como segundo argumento a propriedade class entre aspas, enquanto as propriedades html e href não. Geralmente, nomes de propriedades não precisam estar entre aspas a não ser que elas sejam palavras reservadas (como a class neste caso)

Quando você cria um novo elemento, ele não é adicionado imediatamente à página. Há várias formas de adicionar um elemento à página uma vez que ele esteja criado.

Inserindo um novo elemento na página

```
var $myNewElement = $('<p>Novo elemento</p>');
$myNewElement.appendTo('#content');
$myNewElement.insertAfter('ul:last');          // isto irá remover p de #content!
$('ul').last().after( $myNewElement.clone() ); // clona o p, portanto temos 2 agora
```

Estritamente falando, você não precisa armazenar o elemento criado numa variável -- você pode simplesmente chamar o método para adicioná-lo diretamente depois do `$()`. Entretanto, a maior parte do tempo você precisará de uma referência ao elemento que você adicionou para que você não o selecione depois.

Você ainda pode criar um elemento ao mesmo tempo que você o adiciona à página, mas note que neste caso você não obtém a referência do novo objeto criado.

Criando e adicionando um elemento à página ao mesmo tempo

```
$('ul').append('<li>item de lista</li>');
```

> *A sintaxe para adicionar novos elementos à página é tão fácil que é tentador esquecer que há um enorme custo de performance por adicionar ao DOM repetidas vezes. Se você está adicionando muitos elementos ao mesmo container, você irá concatenar todo html numa única string e então adicionar a string ao container ao invés de ir adicionando um elemento de cada vez. Você pode usar um array para colocar todas os pedaços juntos e então aplicar um `join` nele em uma única string para adicionar ao container.*

```
var myItems = [], $myList = $('#myList');
for ( var i=0; i<100; i++ )
{
    myItems.push('<li>item ' + i + '</li>');
}
$myList.append(myItems.join(''));
```

Manipulando atributos

Os recursos de manipulação de atributos do jQuery são muitos. Mudanças básicas são simples, mas o método $.fn.attr também permite manipulações mais complexas.

Manipulando um único atributo

```
1  $('#myDiv a:first').attr('href', 'novoDestino.html');
```

Manipulando múltiplos atributos

```
1  $('#myDiv a:first').attr(
2      {
3          href : 'novoDestino.html',
4          rel  : 'super-special'
5      }
6  );
```

Usando uma função para determinar um novo valor de atributo

```
1   $('#myDiv a:first').attr({
2       rel : 'super-special',
3       href : function()
4       {
5           return '/new/' + $(this).attr('href');
6       }
7   });
8
9   $('#myDiv a:first').attr('href', function()
10  {
11      return '/new/' + $(this).attr('href');
12  });
```

Exercícios

Selecionando

Abra o arquivo `/exercises/index.html` no seu navegador. Use o arquivo `/exercises/js/sandbox.js` ou trabalhe no Firebug para fazer o seguinte:

1. Selecione todos os elementos DIV que têm a classe "module".
2. Escreva três seletores que você pode usar para obter o terceiro item na lista não-ordenada #myList. Qual é o melhor para se usar? Por quê?
3. Selecione o label para o input de busca usando um seletor de atributo.
4. Encontre quantos elementos na página estão escondidos (dica: .length).
5. Encontre quantos elementos na página têm um atributo alt.
6. Selecione todas as linhas ímpares no corpo de uma tabela.

Atravessamento

Abra o arquivo `/exercises/index.html` no seu navegador. Use o arquivo `/exercises/js/sandbox.js` ou trabalhe no Firebug para fazer o seguinte:

1. Selecione todos os elementos de imagem na página; logue o atributo alt de cada imagem.
2. Selecione a caixa de texto de busca, entÃ£o atravesse para o form e adicione uma classe nele.
3. Selecione o item de lista dentro de #myList que possui uma classe "current" e remova esta classe dele; adicione uma classe "current" no próximo item de lista.
4. Selecione o elemento select dentro de #specials; atravesse para o botÃ£o de submit.
5. Selecione o primeiro item de lista no elemento #slideshow; adicione a classe "current" a ele e entÃ£o adicione a classe "disabled" para seus elementos sibling.

Manipulando

Abra o arquivo `/exercises/index.html` no seu navegador. Utilize o arquivo `/exercises/js/sandbox.js` ou use o Firebug para realizar o seguinte:

1. Adicione cinco novos itens de lista no m da lista não ordenada #myList. Dica:

```
1  for (var i = 0; i<5; i++) { ... }
```

2. Remova os itens ímpares
3. Adicione outro h2 e outro parágrafo à última div.module
4. Adicione outra opção para o elemento select; dê ao option o valor "Quarta-Feira".
5. Adicione uma nova div.module à página depois da última; coloque uma cópia de uma das imagens existentes dentro dela.

Impresso por: **André Gustavo dos Santos Burin**
Matrícula: **6182**