Sistemas Distribuídos e Tolerância a Falhas

jQuery

Ivan Pires – m3797
Gilberto Melfe – m4088

jQuery – O que é?

- Biblioteca javascript, open-source
- Criada por John Resig, no ano de 2006
- Disponível para download em: http://jquery.com/

- O seu objectivo é simplificar o desenvolvimento da parte das aplicações web que corre no lado do cliente

jQuery - Para que serve?

- Adicionar efeitos visuais/animações
- Manipular o DOM (criar/eliminar/reposicionar elementos)
- Aceder a informações fornecidas pelo servidor, sem necessidade de um “refresh” da página (AJAX)

A biblioteca jQuery permite adicionar interactividade e dinamismo a uma página web, com um relativo grau de simplicidade.

DOM

- Representação de um documento numa estrutura tipo árvore
- Interface de programação (API) que permite aceder/modificar a estrutura de um documento, programaticamente

language-neutral interface to an in-memory representation of an XML document

interface that allows programs and scripts to dynamically access and update the content and structure of documents

(document object model) A specification that defines the structure of a document as a collection of objects and how the document can be accessed and manipulated. A document object model is an API (application programming interface), which describes how programs actually interpret the structure and contents of a document.

The Document Object Model is a family-tree structure of sorts. HTML, like other markup languages, uses this model to describe the relationships of things on a page.

jQuery – “Selector” Function

- A função $() permite seleccionar elementos do DOM
- Usa os selectores da linguagem CSS (e alguns próprios)

- $('.poem-stanza') – vai seleccionar elementos com a classe ‘poem-stanza’
- $(‘#myID’) – seleciona o elemento com id, myID
- $(‘p > span’) – selecciona os elementos … filhos de elementos …

A operação fundamental ao trabalhar com jQuery é seleccionar partes do documento (DOM) sobre o qual vamos trabalhar, recorrendo à função $().

A função (método) $() tem várias funcionalidades, a primeira das quais, de seleccionar elementos do DOM e retorná-los na forma de um objecto jQuery. A maior parte dos métodos da biblioteca jQuery são invocados sobre um desses objectos e devolvem-no após realizarem alguma acção sobre ele. Esta característica permite tirar partido duma característica da biblioteca a que se chama “chaining” (demonstrada mais à frente). Para esta funcionalidade a função aceita como parâmetro uma expressão de selecção retirada da sintaxe da linguagem (de descrição de apresentação) CSS.

Por exemplo:

$(‘p’) - selecciona todos os elementos …

$(‘a[href]’) - selecciona os elementos … (“a”, com atributo href)

$(div.error) - (em HTML) selecciona os elementos … (“div”, que “são” da classe “error”, entre outras que podem ter)

$(‘#selected-plays li:not(.horizontal)’) - elementos …, descendentes de elementos com id=“selected-plays” que não sejam de classe Horizontal

$('a[href^=http][href*=henry]') - elementos … (com um atributo href que começe por “http” e contenha “henry” no valor desse atributo)

$('tr:nth-child(even)') – linhas de tabelas em posição par

Alguns selectores próprios (que usam a sintaxe das pseudo-classes CSS) são:

$(‘div.horizontal:eq(1)’) – selecciona os segundos elementos

jQuery – “Selector” Function (cont.)

- Todos os elementos seleccionados são encapsulados por um objecto jQuery

“Outro” método de seleccionar elementos é usar o método .filter() por exemplo:

```
$('tr').filter(':odd').addClass('alt');
```

Este método pode ter como parâmetro uma função que testa os elementos do conjunto inicial e decide mantê-los ou não na selecção.

Por exemplo, para adicionar a classe “external” o todos os links para sítios externos:

```
$('a').filter(function() {
        return this.hostname && this.hostname !=
location.hostname;
}).addClass('external');
```

jQuery – “Implicit iteration”

- Se usarmos a função $(‘p’) para seleccionar parágrafo(s) esta retorna um objecto jQuery que encapsula os elementos DOM “correspondentes”
- Ao aplicar-mos um método sobre esse objecto este itera automaticamente sobre todos os elementos DOM
- $('.poem-stanza').addClass('highlight');

Outra característica importante do funcionamento da biblioteca é a de realizar “implicit iteration”.
A biblioteca jQuery trabalha sempre sobre “conjuntos de objectos”; se uma selecção realizada com a função $() resultar na selecção de vários elementos do documento, um método aplicado ao objecto jQuery devolvido vai trabalhar sobre todos esse elementos, automaticamente, eliminando a necessidade de uma iteração explícita, para cada chamada a determinado método, no nosso próprio código.

jQuery – “Chaining”

- Um método é executado sobre um objecto e na maioria dos casos retorna esse mesmo objecto
- Desta forma podemos encadear invocações

- Por exemplo:

```
$('body').removeClass().addClass('large');
```

A biblioteca jQuery emprega um padrão de programação chamado “chaining” para a maioria dos seus métodos. Cada um desses métodos opera sobre um objecto, e devolve como resultado outro (“o mesmo”) objecto pronto para a chamada a uma nova operação sobre ele.
O que isto significa é que multiplas acções/modificações/efeitos podem ser encadeados e “despachados” numa única linha de código (e sem recorrer a variáveis temporárias).

Um exemplo deste uso pode ser (separado por linhas apenas para melhor compreensão):

```
$('td:contains(Henry)')       // seleccionar as células de tabelas, contendo o texto "Henry"
.parent()       // seleccionar os pais (antecessores directos) (um para cada, supostamente)
.find('td:eq(1)')                 // seleccionar os segundos filhos (as segundas células)
.addClass('highlight')          // adicionar a classe "highlight”
.end()          // retornar à selecção dos pais
.find('td:eq(2)')                 // seleccionar os terceiros filhos (as terceiras células)
.addClass('highlight');         // adicionar a classe "highlight”
```

???

```
$('td:contains(Henry)').nextAll().andSelf().addClass('highlight');
ou
$('td:contains(Henry)').parent().children().addClass('highlight');
```

jQuery – Quando executar?

- Se quisermos correr código assim que o DOM esteja pronto a ser manipulado, usamos o código:

  $(document).ready(/* função a executar */);

Geralmente queremos que o código javascript seja executado quando ocorre um determinado evento, ou seja, que esteja associado a determinado evento.

Podemos querer, por exemplo, correr determinado pedaço de código assim que o DOM correspondente ao documento tenha sido "criado". Vamos então associar a esse acontecimento o nosso código da seguinte maneira:

$(document).ready(/* função a executar */);

Nota: cada chamada ao método acrescenta o comportamento (especificado pela função parâmetro) a uma fila de comportamentos

jQuery - Eventos

- Outro método de escolher quando um pedaço de código é executado, é associar esse código a um evento (por usa vez associado a um elemento do DOM), da seguinte forma:

```
$('#switcher-narrow').bind('click', function() {
  $('body').addClass('narrow').removeClass('large'); });
```

```
$('#switcher-narrow').bind('click', function() {
        $('body').addClass('narrow').removeClass('large');
});
```

Múltiplas chamadas do método .bind() adicionam comportamento ao já existente (os vários comportamentos são levados a cabo por esta ordem de adição).

Quando um "event handler" é executado "this" é uma referência ao elemento DOM (objecto) ao qual aquele foi atribuido. Podemos usar esta referência dentro de $() para criar um objecto jQuery que encapsula aquele elemento DOM e executar sobre ele métodos da biblioteca, como se o tivéssemos seleccionado com $("selector CSS").

Compound Events:

toogle – duas funções como parâmetros, primeiro executa uma, quando se clica, depois a outra no segundo click (alternância)

hover - duas funções como parâmetros, primeiro executa uma quando o rato se posiciona sobre o elemento, depois a segunda quando o rato sai de cima do elemento

Para remover um event handler:

```
$('#switcher').unbind('click')
```

jQuery - Efeitos

- .css() – consultar/especificar propriedades CSS
- .hide() e .show(), .fadeIn() e .fadeOut(), .toggle() e .slideToggle() – mostrar/ocultar elementos
- .animate() – para criarmos os nossos próprios efeitos

O método .css() pertmite obter, e especificar o valor de uma (ou várias) propriedades css (pode receber como parâmetro um objecto javascript para esse efeito).

Os métodos .hide() e .show() interagem com a propriedade css, "display", para mostrarem ou ocultarem elementos.
Os métodos .fadeIn() e .fadeOut() têm funções semelhantes aos anteriores.

.toggle() actua como show e hide alternados
.slideToggle() mostra ou oculta os elementos com um slide

Com o método .animate() podemos criar animações à medida.
Usa-se da seguinte forma:

```
.animate( { property1: 'value1', property2: 'value2'}, speed, easing, function() {
                alert('The animation is finished.');
                }
);
```

Por exemplo:

```
$('p:eq(1)').animate({opacity: 'toggle'}, 'slow');
$('p:eq(1)').animate({ opacity: 'toggle', height: 'toggle' }, 'slow');
```

Para encadear efeitos podemos recorrer ao "chaining".

Para encadear "efeitos" não-encadeáveis (método .css(), por exemplo), sobre um único conjunto de elementos, usamos os médotos .queue()/.dequeue() como mostrado abaixo:

```
$switcher.fadeTo('fast',0.5).animate({
                'left': paraWidth – switcherWidth }, 'slow')
                .fadeTo('slow',1.0).slideUp('slow')
                .queue(function() {
                                $switcher.css('backgroundColor', '#f00').dequeue();
                })
                .slideDown('slow');
```

Importante: se não usarmos o método .dequeue() as animações seguintes (o .slideDown() no caso) não executariam!

Para encadear "efeitos" sobre um conjuntos de elementos diferentes (efeitos que executariam virtualmente simultaneamente) temos que passar ao método que realiza o efeito uma callback. Por exemplo:

```
$(this).next().slideDown('slow',function() {
                                $thirdPara.slideUp('slow');
                });
```

Nota: Não queremos usar a referência "this" dentro da callback, porque esta mudou (não é a mesma que era fora da função callback)

jQuery – Manipulação do DOM

- Os métodos da biblioteca podem criar/eliminar/transformar elementos e seus atributos.
- A função $() pode receber HTML como parâmetro e criar elementos DOM a partir dele, que depois são inseridos na estrutura com métodos de inserção.

.attr() e .removeAttr() para manipular atributos

```
$(document).ready(function() {
        $('div.chapter a').attr({'rel': 'external'});
});
```

.each() – iterador explícito, usado quando queremos aplicar código específico a cada elemento do conjunto seleccionado

```
$('div.chapter a').each(function(index) {
        $(this).attr({ 'rel': 'external',
                'id': 'wikilink-' + index
                });
         });
```

Para inserir elementos antes ou depois de outros elementos usamos:

.insertBefore(), .insertAfter() – selecção de elementos, após os quais, ou antes dos quais vamos inserir, é dada como parâmetro dos métodos, que são chained à criação de elementos;

com estes métodos podemos encadear mais acções sobre o elemento criado;

.before(), .after() - selecção de elementos, após os quais, ou antes dos quais vamos inserir, provém da cadeia de métodos

Para inserir elementos dentro de outros elementos usamos:

.prependTo() – insere no início

.appendTo() – insere no fim

Nota: “Keep in mind that even in cases of implicit iteration, the order of insertion is predefined, starting at the top of the DOM tree and working its way down.”

jQuery - AJAX

- É um meio de descarregar informação do servidor para o cliente (browser, que corre a aplicação web) sem necessidade de efectuar um “refresh” de toda a página.
- Para descarregar um fragmento de html usamos o método .load(url), em que o url é o endereço respectivo.

O método load() vai carregar o conteúdo do ficheiro, presente no servidor, e cujo endereço é especificado como seu parâmetro, para dentro do elemento seleccionado pela função $(). Por exemplo:

```
$('#dictionary').load('a.html');
```

A este caso, de um ficheiro html (um pedaço de html, não um documento html completo), é costume chamar-se AHAH (Asynchronous HTTP and HTML).

Nota: as chamadas AJAX são por omissão realizadas de modo Assíncrono.

jQuery – AJAX (cont.)

- Para descarregar informação representada no formato JSON usamos o método $.getJSON(url, callback), em que o url é o endereço respectivo, e a callback é uma função, chamada quando o pedido for completado, que recebe como parâmetro o objecto javascript retornado do pedido e o vai processar com vista a incluir a informação na página.

Podemos também usar o método $.getJSON que retira do servidor um ficheiro, que contém objectos javascript expressos na representação JSON (pares chave-valor entre {} e arrays entre []), e o processa, fornecendo esses objectos ao código que o invocou. Nota: este método é definido sobre O objecto jQuery (ou $) global e único (porquê?, ver pág 121 do livro Learning Jquery). Outro parâmetro é uma função (callback) que vai processar os objectos javascript obtidos (recebe-os como parâmetro) e gerar o HTML a ser inserido na página.

jQuery – AJAX (cont.)

- Para descarregar informação representada no formato XML usamos o método $.get (url, callback), em que o url é o endereço respectivo, e a callback é uma função, chamada quando o pedido for completado, que recebe como parâmetro o XML (possivelmente o DOM) retornado do pedido e o vai processar com vista a incluir a informação na página.

Para dados (enviados pelo servidor) expressos em XML usamos a função $.get() que tem dois parâmetros, o URL do ficheiro no servidor e uma callback (função) que vai processar os dados em XML. A callback pode receber os dados em plain text, ou se o servidor especificar o MIME type para o ficheiro como sendo XML, receber a árvore DOM correspodente ao XML enviado pelo servidor.
A biblioteca jQuery já disponibiliza métodos de navegação na árvore DOM (como por exemplo, .find(), que permite encontrar descendentes de uma selecção feita com a função $()) que podemos utilizar para encontrar os dados necessários e posteriormente processá-los e integrá-los na página a construir.

jQuery – AJAX (cont.)

- Quando recebemos XML podemos usar as capacidades de navegação no DOM fornecidas pela biblioteca (métodos .find() .filter(), etc., com as novas tags; no exemplo a apresentar vamos procurar os elementos “entry” e “definition”)

jQuery – Ajax (cont.)

- Além de descarregar informação também podemos enviar informação ao servidor (GET ou POST).
- Cada um dos métodos já utilizados pode receber como parâmetro um objecto javascript que vai ser transformado em parte da “query string” do pedido.

Referências

- http://api.jquery.com/
- Learning jQuery 1.3 – Packt Publishing
- jQuery – A biblioteca do programador Javascript