Xestionando o conxunto de elementos seleccionables

Sumario

- 1 Determinando o tamaño do conxunto
- 2 Obtención de elementos do conxunto
- 3 **Sobre o metodo get()** Este método creouse como retro-compatibilidade con Javascript para acceder a tódolos elementos.É util si necesitamos operar nos obxectos DOM en lugar de empregar funcións de jQuery.Polo tanto se facemos utilidade do método **get() ou a indexación por []** , teremos que usar a continuación **métodos e propiedades de Javascript**, xa que senón obteremos un erro de propiedade ou método incorrecto.
- 4 Ampliando ou reducindo o conxunto
    - 4.1 Engadir máis elementos ó conxunto
    - 4.2 Seleccións exclusivas
    - 4.3 Filtrado de seleccións
- 5 **Método filter()** Cando ó método filter() lle pasamos unha función, o que fai é chamar esa función para cada elemento da selección previa, e sacará da selección ese elemento sempre e cando a función devolva como valor false.
    - 5.1 Subconxuntos na selección
- 6 Obtención de conxuntos empregando relacións
- 7 Máis formas de obtención de subconxuntos
    - 7.1 O método find()
    - 7.2 O método is()
- 8 Recorrendo un conxunto con each()
- 9 Encadeamentos en jQuery

Determinando o tamaño do conxunto

Unha vez que temos o conxunto de elementos seleccionados (por un selector ou creados como novos elementos, ou unha combinación de ambos), estamos preparados para manipular todos eses elementos empregando os comandos de jQuery.

Nesta sección exploraremos as diferentes formas de redefinir, extender ou dividir o conxunto de elementos para poder operar con eles.

O conxunto de elementos seleccionados en jQuery funciona moitas veces como un array. Como todos sabemos os array inclúen a propiedade length que se pode empregar en JavaScript.

En jQuery dispomos do método **size()** o cal nos proporciona a mesma información que length en JavaScript.

Vexamos o seguinte exemplo:

```
$('#someDiv').html('Hai  '+$('a').size()+' enlace(s) nesta paxina.');
```

O selector $('a') selecciona todos os enlaces do documento e o método size() devolverá o número de elementos seleccionados.

Obtención de elementos do conxunto

jQuery permítenos tratar o conxunto de elementos como un array de JavaScript. Podemos empregar o indexado de arrays para obter calqueira elemento no conxunto pola súa posición na lista.

Por exemplo, para obter o primeiro elemento no conxunto de tódalas imaxes que teñan atributo alt podemos escribir:

```
$('img[alt]')[0]
```

Si preferimos empregar un método distinto da indexación, jQuery proporciona o método **get()** para facer esa tarefa.

 Sobre o metodo get()

Este método creouse como retro-compatibilidade con Javascript para acceder a tódolos elementos.

É util si necesitamos operar nos obxectos DOM en lugar de empregar funcións de jQuery.
Polo tanto se facemos utilidade do método **get() ou a indexación por []** , teremos que usar a continuación **métodos e propiedades de Javascript**, xa que senón obteremos un erro de propiedade ou método incorrecto.

Por exemplo:

```
O fragmento:

$('img[alt]').get(0)

é equivalente ó exemplo anterior.

Outro exemplo:

$(this).get(0) é equivalente a usar $(this)[0]
```

O método get() tamén pode ser usado para obter un array de JavaScript. Exemplo:

```
var allLabeledButtons = $('label+button').get();
```

Esta instrucción selecciona todos os elementos button da páxina que están inmediatamente precedidos por elementos label

Tamén podemos facer a operación inversa para atopar o índice dun elemento particular nun conxunto. Por exemplo si queremos obter o indice da primeira imaxe que teña como ID Santiago dentro do conxunto de tódalas imaxes do documento, poderemos obter este valor co método **index()**:

```
var n = $('img').index($('img#Santiago')[0]);
```

Ampliando ou reducindo o conxunto

Unha vez que xa temos definido o conxunto de elementos, poderíamos aumenta-lo ou reduci-lo a un subconxunto do mesmo. jQuery aporta unha larga colección de métodos para xestionar estes elementos.

Engadir máis elementos ó conxunto

Ás veces podemos estar na situación de querer engadir máis elementos a un conxunto que xa temos definido. As posibilidades de jQuery de encadeado de comandos permítenos facer esta tarefa nunha soa liña.

Vexamos o seguinte exemplo:

```
$('img[alt],img[title]')
```

Seleccionará tódalas imaxes que teñan tanto un atributo alt, como tódalas imaxes que teñan un atributo title.

Para face-lo mesmo empregando o método **add()** faríamos:

```
$('img[alt]').add('img[title]')
```

Empregando o método add(), permítenos relacionar un conxunto de selectores previos permitindo crear unha unión dos conxuntos que satisfará ambos grupos de selectores.

Para desfacer esta unión feita con **add()** disporemos do método **end()** que veremos máis adiante.

Outro exemplo con add():

```
$('img[alt]').addClass('bordesinxelo').add('img[title]').addClass('transparencia')
```

Nesta instrucción estamos seleccionando tódalas imaxes que teñan un atributo alt,e a continuación aplicámoslle a clase 'bordesinxelo'. A ese conxunto de imáxenes engadirémoslle mediante **add()**, todas as imaxes que teñan un atributo title e a ese conxunto resultante aplicámoslle a clase 'transparencia'.

Podemos empregar o método **add()** para engadir novos elementos pasando o código HTML. Por exemplo:

```
$('p').add('<div>Hola mundo!</div>')
```

A tódolos párrafos do documento engádeselle un div co contido Hola mundo!.

Seleccións exclusivas

Cando queremos seleccionar un conxunto que si cumpla unhas certas características, pero que non cumpla outras.

Por exemplo:

```
$("img[title]").not("[title*='cachorro']")
```

Nesta instrucción estamos seleccionando todas aquelas imaxes que teñen un atributo title, pero que non conteñan a palabra 'cachorro' nese atributo.

Os selectores que podemos pasar ó metodo **not()** están limitados a expresións de filtrado. Si pasáramos un selector máis explícito img[title*=cachorro] non obteríamos o resultado previsto, xa que a selección de elementos non está soportada por **not()**.

Filtrado de seleccións

Hai momentos nos que é moi complexo filtrar nuha selección empregando os selectores normales. Para elo dispomos en jQuery do método **filter()**.

 Método filter()

Cando ó método filter() lle pasamos unha función, o que fai é chamar esa función para cada elemento da selección previa, e sacará da selección ese elemento sempre e cando a función devolva como valor false.

Por exemplo, si queremos seleccionar todas as celdas dunha táboa que conteñan valores numéricos. Empregando os selectores de jQuery é imposible de facer. Para elo nestes casos é moi interesante o uso de **filter()** do seguinte xeito:

```
$('td').filter(function(){return this.innerHTML.match(/^\d+$/)})
```

A función usa unha expresión regular para determinar si o elemento cumple o patrón especificado (unha secuencia de un ou máis díxitos), devolvendo false si non o cumple. Para cada elemento cuia función devolve false, será eliminado do conxunto inicial $('td').

Máis información sobre métodos de filtrado

Subconxuntos na selección

Ás veces podemos desexar obter un subconxunto da selección inicial, baseados na posición dos elementos dentro do conxunto. Para facer isto jQuery proporciona un método chamado **slice(comezo,final)**.

O parámetro **comezo** indica a posición do primeiro elemento a ser incluído (comenzando na posición 0). O parámetro **final** indica a posición do elemento que NON vai a estar incluído no subconxunto, ou dito doutro xeito a posición do derradeiro elemento + 1. Si se omite o parámetro final extenderase ata o derradeiro elemento.

Exemplos:

```
$('*').slice(2,3);
```

Devolverá o terceiro elemento (recordar que comenza na posición 0).

Recordar que isto é distinto a $('*').get(2) que devolve o terceiro elemento no conxunto, non un subconxunto.

Polo tanto, unha instrucción como:

```
$('*').slice(0,4)
```

seleccionará todos os elementos da páxina e creará un subconxunto contendo os 4 primeiros elementos.

Si escribimos:

```
$('*').slice(6);
```

seleccionará todos os elementos da páxina comenzando no 7º elemento (posición 6) ata o final.

Obtención de conxuntos empregando relacións

jQuery permítenos obter novos conxuntos partindo dun conxunto existente, baseándonos nas relacións xerárquicas que existen entre eles.

Na táboa seguinte amósanse os métodos e as súas descripcións (en inglés). Todos estes métodos aceptan unha expresión selector de xeito opcional, que filtraría o conxunto orixinal. Si non se lle pasa dito selector serán seleccionados todos os elementos.

Método	Descripcion
children()	Devolve un conxunto de elementos consistente nos fillos únicos do conxunto ó que se lle aplica o método.
contents()	Devolve un conxunto cos contidos dos elementos, que pode incluir nodos de texto. (Moitas veces úsase para obter o contido de elementos de tipo <iframe>.
next()	Devolve un conxunto consistente no irmán seguinte tomando como referencia o conxunto ou elemento ó que se lle aplica o método.
nextAll()	Devolve un conxunto consistente en todos os fillos seguintes tomando como referencia o conxunto ó que se lle aplica o método.
parent()	Devolve un conxunto consistente no pai directo do conxunto ó que se lle aplica o método.
parents()	Devolve un conxunto consistente en todos os pais do conxunto ó que se lle aplica o método. Isto inclúe os pais directos e así todos hacia arriba ata chegar á raiz do documento.
prev()	Devolve un conxunto consistente no irmán anterior tomando como referencia o conxunto ou elemento ó que se lle aplica o método.
prevAll()	Devolve un conxunto consistente en todos os irmán anteriores tomando como referencia o conxunto ou elemento ó que se lle aplica o método.
siblings()	Devolve un conxunto consistente en todos os irmáns tomando como referencia o conxunto ou elemento ó que se lle aplica o método.

Exemplo de uso do método prev():

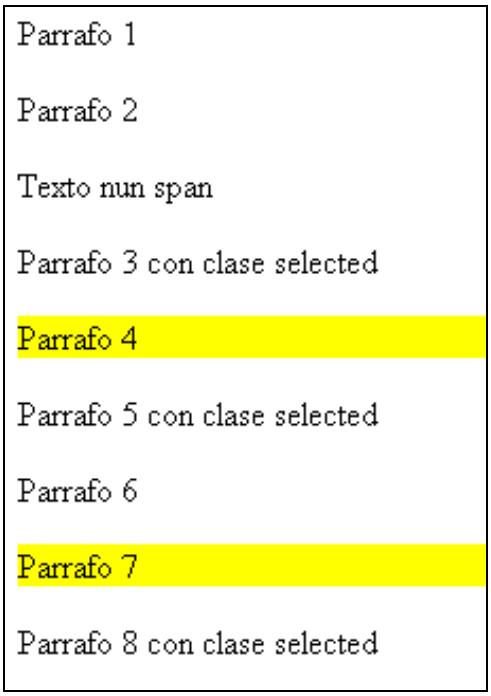
Seleccionaría todos os párrafos con clase selected, e tomando como referencia eses párrafos collería o irman anterior, sempre e cando sexan párrafos -> prev("p") e aplicaralles como color de fondo o amarelo.

Si non pomos un filtro en prev() collerá o irmán anterior.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Parrafos con jQuery</title>
</head>
<body>
  <p>Parrafo 1</p>
  <p>Parrafo 2</p>

  <span>Texto nun span</span>
  <p class="selected">Parrafo 3 con clase selected</p>

  <p>Parrafo 4</p>
  <p class="selected">Parrafo 5 con clase selected</p>

  <p>Parrafo 6</p>
  <p>Parrafo 7</p>
  <p class="selected">Parrafo 8 con clase selected</p>

  <script src="http://code.jquery.com/jquery-2.1.3.min.js"></script>
  <script>
    $(document).ready(function(){
      $("p.selected").prev("p").css("background", "yellow");
```

```
    });
  </script>
</body>
</html>
```

Daría como resultado:



Máis información e exemplos

Máis formas de obtención de subconxuntos

O método find()

Este método permítenos facer unha búsqueda dentro dun conxunto e devolver un novo subconxunto que conteña todos os elementos que cumplen o selector que se lle pasa como parámetro.

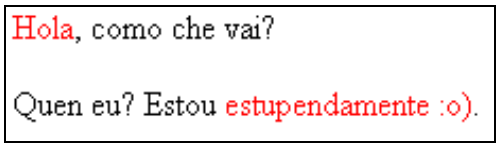
Exemplo:

```
$('div').find('p strong')
```

Devolverá de entre tódolos div do documento aqueles que conteñan unha marca strong dentro dun párrafo

Exemplo de uso método find():

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Mas jQuery</title>
</head>
<body>
  <p><span>Ola</span>, como che vai?</p>
  <p>Quen eu? Estou <span>estupendamente :o)</span>.</p>

  <script src="http://code.jquery.com/jquery-latest.js"></script>
  <script>
    $(document).ready(function(){
      $("p").find("span").css('color','red');
    });
  </script>
</body>
</html>
```

Da como resultado: 

O método is()

Este método permítenos comprobar si no conxunto temos algún elemento que cumple o selector que se lle pasa como parámetro. O método **is()** devolverá true si polo menos un elemento coincide co selector, devolverá false noutro caso.

Exemplo:

```
var tenImaxes = $('*').is('img');
```

Si temos algún elemento img no documento a variable tenImaxes terá o valor true, false noutro caso.

Recorrendo un conxunto con each()

Mediante o método **each(función())** poderemos executar unha función dentro do contexto de cada elemento do conxunto. Isto quere dicir que cada vez que se execute a función que programemos no each() teremos acceso a palabra clave **this** ou **$(this)** que fará referencia ó elemento actual do conxunto que estamos evaluando.

De xeito adicional cada vez que a función é executada si definimos un parámetro nesa función, ese parámetro indicará a posición actual do elemento que estamos evaluando, dentro do conxunto, comenzando pola posición 0. Si devolvemos false dentro da función párase o bucle de execución do each, si devolvemos true equivale a 'continue', é dicir salta ó seguinte elemento.

Exemplo:

```
$("img").each (function(numimaxen)
{
     alert("O ancho da imaxen "+numimaxen+ "é de "+$(this).width()+" pixels.");
    // Ou tamén se podería por:
     alert("O ancho da imaxen "+numimaxen+ "é de "+this.width+" pixels.");
});
```

Encadeamentos en jQuery

Gracias o encadeamento podemos escribir nunha soa liña un montón de comandos para realizar tarefas complexas. As ventaxas de facer isto radica sobre todo na eficiencia, xa que os conxuntos ou subconxuntos non teñen que ser recalculados para poder aplicarlle novos métodos.

Dependendo dos métodos aplicados poden xerarse múltiples subconxuntos. Por exemplo o método **clone()** crea copias dos elementos seleccionados nun paso previo. Si facemos a copia dos elementos a partir dese momento deixamos de acceder ó conxunto orixinal.

Vexamos o seguinte exemplo:

```
$('img').clone().appendTo('#zona');
```

Aquí temos dous conxuntos: o orixinal que ten tódalas imaxes do documento e un segundo conxunto consistente nas copias desas imaxes. O método **clone()** devolveu este segundo conxunto e sobre este conxunto aplicámoslle o método appendTo.

Pero que pasaría si consecuentemente queremos aplicar outro comando, por exemplo engadir unha clase o conxunto orixinal (despois da instrucción de clonado). Neste caso non poderíamos facelo, xa que afectaría ó conxunto clonado non ó conxunto orixinal de imaxes.

Para elo jQuery dispón do comando **end()**.

Este método cando é empregado nunha cadea jQuery, devolverá ó conxunto previo, polo que poderemos seguir aplicando máis operacións na cadea. Vexamos o exemplo:

```
$('img').clone().appendTo('#zona').end().addClass('bordes');
```

O método appendTo() devolverá despois da súa execución os clones das imaxes, pero chamando o método **end()** regresamos ó conxunto previo (as imaxes orixinais), ás que se lles aplicará o método addClass. Si non usamos o método end() o comando addClass() operaría sobre o conxunto de imaxes clonadas.

--Veiga (discusión) 13:54 26 ene 2015 (CET)