JQuery

Para simplificar, podríamos decir que jQuery es un framework Javascript, es un producto que sirve como base para la programación avanzada de aplicaciones, que aporta una serie de funciones o códigos para realizar tareas habituales. Por decirlo de otra manera, framework son unas librerías de código que contienen procesos o rutinas ya listos para usar. Los programadores utilizan los frameworks para no tener que desarrollar ellos mismos las tareas más básicas, puesto que en el propio framework ya hay implementaciones que están probadas, funcionan y no se necesitan volver a programar.

Por ejemplo, en el caso que nos ocupa, jQuery es un framework para el lenguaje Javascript, luego será un producto que nos simplificará la vida para programar en este lenguaje. Como probablemente sabremos, cuando un desarrollador tiene que utilizar Javascript, generalmente tiene que preocuparse por hacer scripts compatibles con varios navegadores y para ello tiene que incorporar mucho código que lo único que hace es detectar el browser del usuario, para hacer una u otra cosa dependiendo de si es Internet Explorer, Firefox, Opera, etc. jQuery es donde más nos puede ayudar, puesto que implementa una serie de clases (de programación orientada a objetos) que nos permiten programar sin preocuparnos del navegador con el que nos está visitando el usuario, ya que funcionan de exacta forma en todas las plataformas más habituales.

Es importante comentar que jQuery no es el único framework que existe en el mercado. Como es normal, cada uno de los frameworks tiene sus ventajas e inconvenientes, pero jQuery es un producto con una aceptación por parte de los programadores muy buena y un grado de penetración en el mercado muy amplio. Además, es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del framework. Otra cosa muy interesante es la dilatada comunidad de creadores de plugins o componentes, lo que hace fácil encontrar soluciones ya creadas en jQuery para implementar asuntos como interfaces de usuario, galerías, votaciones, efectos diversos, etc.

UTILIZAR JQUERY

Para poder trabajar con JQuery disponemos de varios métodos, descargar la última versión del framework o utilizar un CDN existente en internet.

Para descargar JQuery desde la pagina web tenemos dos posibilidades, una que es la adecuada para páginas web en producción, puesto que está minimizada y ocupa menos espacio, con lo que la carga de nuestro sitio será más rápida. La otra posibilidad es descargar la versión que está con el código sin comprimir, con lo que ocupa más espacio, pero se podrá leer la implementación de las funciones del framework, que puede ser interesante en etapa de desarrollo, porque podremos bucear en el código de jQuery por si tenemos que entender algún asunto del trabajo con el framework.

Una vez descargado lo colocaremos en un lugar conveniente para que podaos utilizarlo en el código de la pagina html.

```
8
9     <script src="jquery-3.1.1.js"></script>
10
```

La otra manera es la de utilizar un **CDN** de internet. CDN significa *Content Delivery Network*, que se traduciría como red de entrega de contenido y no es otra cosa que un servicio que nos permite incluir las librerías de código de jQuery desde los servidores de algunas importantes empresas.

```
8
9     <script src="http://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
10
```

En la mayoría de los casos, usar un servidor CDN es una buena idea, por diversos motivos, entre los que podríamos resumir:

- **Mayor velocidad**: Los servicios CDN están ofrecidos por grandes empresas, con replicación de servidores y diversas localizaciones de entrega a lo largo del mundo. Posiblemente, cualquiera de los otros proveedores CDN, pueda enviar el script jQuery más rápido que tu propio servidor.

- **Cacheado probable**: Es muy probable que la persona que te visita ya haya cacheado el script jQuery, tras la visita a otra página web que esté usando también el CDN de alguna de estas empresas.

Como todo en la vida, también podemos encontrar algunos inconvenientes:

- Necesitamos estar conectados a Internet para acceder al CDN.

- Tenemos menor control: No puedes tener total control sobre lo que estás trayéndote como script. Claro que el script estará correcto, pero no podrás modificarlo si lo necesitas, ya que está en otro servidor.

Cuando hacemos ciertas acciones complejas con Javascript tenemos que estar seguros que la página haya terminado de cargar y esté lista para recibir comandos Javascript que utilicen la estructura del documento con el objetivo de cambiar cosas, como crear elementos, quitarlos, cambiar sus propiedades, etc.

La sentencia *window.onload*, se ejecutará cuando el navegador haya descargado completamente TODOS los elementos de la página, lo que incluye imágenes, iframes, banners, etc.

Pero en realidad no hace falta esperar todo ese tiempo de carga de los elementos de la página para poder ejecutar sentencias Javascript que alteren el DOM de la página. Para ello, jQuery incluye una manera de hacer acciones justo cuando ya está lista la página, aunque haya elementos que no hayan sido cargados del todo. Esto se hace con la siguiente sentencia.

```
18  <body>
19  <div id="micapa">HOLA MUNDO!!</div>
20  <div> Contenido de la pagina</div>
21  <script src="http://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
22  <script>
23  $(document).ready(function(){
24     $("#micapa").addClass('otro');
25  });
26  </script>
27  </body>
28  </html>
```

Existe una forma abreviada *para $(document).ready()* la cual podrá encontrar algunas veces, sin embargo, es recomendable no utilizarla en caso que este escribiendo código para gente que no conoce jQuery.

```
18  <body>
19  <div id="micapa">HOLA MUNDO!!</div>
20  <div> Contenido de la pagina</div>
21  <script src="http://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
22  <script>
23  $(function(){
24     $("#micapa").addClass('otro');
25  });
26  </script>
27  </body>
28  </html>
```

Además es posible pasarle a *$(document).ready()* una función nombrada en lugar de una anónima.

```
18  <body>
19  <div id="micapa">HOLA MUNDO!!</div>
20  <div> Contenido de la pagina</div>
21  <script src="http://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
22  <script>
23  function funcionR(){
24     $("#micapa").addClass('otro');
25  }
26  $(document).ready(funcionR)
27  </script>
28  </body>
29  </html>
```

SELECTORES JQUERY

Como la propia palabra indica, los selectores son un mecanismo, disponible en jQuery, para seleccionar determinados elementos de la página. Los selectores, al menos los más básicos, son parecidos, o iguales, a los que se utilizan en CSS para seleccionar los elementos a los que se desean aplicar ciertos estilos. Estos son los selectores mas básicos.

```
/* Seleccionar etiquetas del DOM */
$("elemento")
/* Seleccionar id del DOM */
$("#idelemento")
/* Seleccionar clase del DOM */
$(".claseelemento")
```

```
/* seleccionar las etiquetas "p" */
$("p")

/* seleccionar los elementos con id="inicio" */
$("#inicio")

/* seleccionar los elementos con clase="principal" */
$(".principal")

/* seleccionar los elementos con id="inicio" con clase="principal" y que son etiqueta "p" */
$("p#inicio.principal")

/* seleccionar las etiquetas "a" con clase="principal" dentro de una etiqueta "p" */
$("p a.principal")

/* seleccionar los elementos con id="inicio" o clase ="principal" */
$("#inicio,.principal")

/* seleccionar todos los elementos */
$("*")

/* seleccionar todas las etiquetas "p", "a" y "div" */
$("p,a,div")
```

```
/*En este ejemplo controlamos los enlaces (etiqueta “a”)
que están dentro de un párrafo (etiqueta “p”). Es decir,
 que los enlaces (elemento ‘a’) sean hijos directos de ‘p’.
 Si hubiese otro elemento/etiqueta como un “div” o “span”
 que contiene al elemento ‘a’ no se controlaría.*/
$("p > a")

/*En este otro ejemplo estaríamos controlando los enlaces
que están dentro de un párrafo que a su vez están en un
elemento ‘li’ de una lista con clase = “miclase”.*/
$("ul.miclase li > p > a")

/*En el código inferior controlamos los enlaces (elemento ‘a’)
que son precedidos inmediatamente por el elemento ‘div’, es
decir que ‘a’ debe ir justo después de ‘div’ sin que ningún
 otro elemento se interponga entre ellos.*/
$("div + a")

/*Con el código inferior seleccionamos los enlaces (elemento ‘a’)
 prececidos por cualquier hermano suyo en un ‘div’.*/
$("div ~ a")
```

SELECTORES JQUERY POR ATRIBUTOS

Veamos cómo podemos seleccionar elementos mediante las propiedades de sus atributos. Comencemos controlando el tipo de atributo.

```
/* seleccionar los elementos con el atributo href */
$("[href]")
/* seleccionar los elementos con el atributo src */
$("[src]")
/* seleccionar los elementos con el atributo class */
$("[class]")
/* seleccionar los elementos con el atributo id */
$("[id]")

/* seleccionar elementos con atributo href=index.html */
$("[href='index.html']")
/* seleccionar elementos con atributo href distinto a index.html */
$("[href!='index.html']")
/* seleccionar elementos con atributo src que finalicen con .png */
$("[src$='.png']")
/* seleccionar elementos con atributo src que comiencen con 'imagen.jpg' o con atributo src ='imagen.jpg' */
$("[src|='imagen.jpg']")
/* seleccionar elementos con atributo title que comiencen con 'alergia' */
$("[title^='alergia']")
/* seleccionar elementos con atributo title que contengan la palabra específica 'alergia' */
$("[title~='alergia']")
/* seleccionar elementos con atributo title que contengan la palabra 'alergia' */
$("[title*='alergia']")
```

SELECTORES JQUERY POSICIONALES

Estos selectores están basados en las relaciones posicionales entre elementos (como veíamos antes en ejemplo de la estructura del DOM). Como antes, los vamos a ver a través de ejemplos.

```
/* seleccionar el primer enlace en la web, elemento 'a' */
$("a:first")
/* seleccionar el último enlace en la web, elemento 'a' */
$("a:last")
/* seleccionar la fila par de una tabla, elemento tr */
$("tr:even")
/* seleccionar la fila impar de una tabla, elemento tr */
$("tr:odd")

/* seleccionar (elementos 'a') que sean el primer elemento hijo
de su elemento padre */
$("a:first-child")
/* seleccionar (elementos 'a') que sean el último elemento hijo
de su elemento padre */
$("a:last-child")
/* seleccionar (elementos 'a') que sean el primer elemento hijo
de tipo 'a' de su elemento padre */
$("a:first-of-type")
/* seleccionar (elementos 'a') que sean el último elemento hijo
de tipo 'a' de su elemento padre */
$("a:first-of-type")
/* seleccionar (elementos 'a') que sean el quinto elemento hijo
de su elemento padre */
$("a:nth-child(5)")
```

```
/* seleccionar (elementos 'a') que sean el segundo elemento hijo
de su elemento padre comenzando por el final */
$("a:nth-last-child(2)")
/* seleccionar (elementos 'a') que sean el segundo elemento hijo
de tipo 'a' de su elemento padre */
$("a:nth-of-type(2)")
/* seleccionar (elementos 'a') que sean el cuarto elemento hijo
de tipo 'a' de su elemento padre comenzando por el final */
$("a:nth-last-of-type(4)")
/* seleccionar (elementos 'a') que sean el único elemento hijo
de su elemento padre */
$("a:only-child")
/* seleccionar (elementos 'a') que sean el único elemento hijo
de tipo 'a' de su elemento padre */
$("a:only-of-type")
```

SELECTORES JQUERY FORMULARIOS

Cuando trabajemos con formularios jQuery nos ofrece una serie de selectores propios que nos permiten seleccionar de manera sencilla el elemento preciso. Vamos a verlos con ejemplos:

```
/* seleccionar todos los inputs */
$(":input")
/* seleccionar todos los elementos <input> con un tipo de atributo
igual al nombre del selector*/
$(":text")
$(":password")
$(":radio")
$(":checkbox")
$(":submit")
$(":reset")
$(":button")
$(":image")
$(":file")
/* seleccionar todos los input habilitados */
$(":enabled")
/* seleccionar todos los input deshabilitados */
$(":disabled")
/* seleccionar todos los input seleccionados (elementos option) */
$(":selected")
/* seleccionar todos los input checkeados (radio button) */
$(":checked")
```

FILTROS PARA SELECTORES

Hay ocasiones en las que necesitamos acceder de manera más directa a un elemento. En este caso tenemos a nuestra disposición los métodos transversales, que nos permiten ir hacia arriba, hacia abajo y por todo el DOM sin problemas. En este artículo vamos a ver uno de ellos, los *filtros.*

Los filtros son aquellos que concretan una selección de elementos ya realizada aplicando una selección adicional que limita la selección anterior (por ejemplo, hemos

seleccionado previamente todos los elementos *div* y ahora queremos sólo el cuarto *div* de esa selección).

```
// el elemento div.foo contiene elementos <p>
$('div.foo').has('p');

// el elemento h1 no posse la clase 'bar'
$('h1').not('.bar');

// un item de una lista desordenada
// que posse la clase 'current'
$('ul li').filter('.current');

// el primer item de una lista desordenada
$('ul li').first();

// el sexto item de una lista desordenada
$('ul li').eq(5);
```

ENCADENAMIENTOS

Si en una selección se realiza una llamada a un método, y éste devuelve un objeto jQuery, es posible seguir un "encadenado" de métodos en el objeto. Si se está escribiendo un encadenamiento de métodos que incluyen muchos pasos, es posible escribirlos línea por línea, haciendo que el código luzca más agradable para leer.

```
$('#content').find('h3').eq(2).html('nuevo texto para el tercer elemento h3');


$('#content')
        .find('h3')
        .eq(2)
        .html('nuevo texto para el tercer elemento h3');
```

Si desea volver a la selección original en el medio del encadenado, jQuery ofrece el método *$.fn.end* para poder hacerlo.

```
$('#content')
  .find('h3')
  .eq(2)
    .html('nuevo texto para el tercer elemento h3')
  .end() // reestablece la selección a todos los elementos h3 en #content
  .eq(0)
    .html('nuevo texto para el primer elemento h3');
```

OBTENEDORES Y ESTABLECEDORES

jQuery "sobrecarga" sus métodos, en otras palabras, el método para establecer un valor posee el mismo nombre que el método para obtener un valor. Cuando un método es utilizado para establecer un valor, es llamado método establecedor (en inglés *setter*). En cambio, cuando un método es utilizado para obtener (o leer) un valor, es llamado obtenedor (en inglés *getter*).

```
/* Obtenemos el valor de la etiqueta <h1> */
$('h1').html();

/* Establecemos 'hello world' como texto de
la etiqueta <h1> */
$('h1').html('hello world');
```

Los métodos establecedores devuelven un objeto jQuery, permitiendo continuar con la llamada de más métodos en la misma selección, mientras que los métodos obtenedores devuelven el valor por el cual se consultó, pero no permiten seguir llamando a más métodos en dicho valor.

PROPIEDADES CSS y DIMENSIONES

Las propiedades CSS que incluyen como separador un guión del medio, en JavaScript deben ser transformadas a su estilo *CamelCase*. Esta regla no es aplicada cuando se pasa el nombre de la propiedad CSS al método *$.fn.css*, en este caso, los dos formatos (en *CamelCase* o con el guión del medio) funcionarán.

```
// devuelve una cadena de caracteres como "19px"
$('h1').css('fontSize');

// Este metedo de escritura tambien funcionara.
$('h1').css('font-size');

// establece una propiedad individual CSS
$('h1').css('fontSize', '100px');

// establece múltiples propiedades CSS
$('h1').css({
  'fontSize' : '100px',
    'color' : 'red'
  });

/* establece múltiples propiedades CSS con
valores relativos*/
$('h1').css({
        'fontSize' : '+=15px',
        'paddingTop' : '+=20px'
    });
```

A partir de la versión 1.6 de la biblioteca, utilizando *$.fn.css* también es posible establecer valores relativos en las propiedades CSS de un elemento determinado