Tarea 8: Funciones JQuery
Materia: Programación Web
Alumno: Silva Martínez José Luis
Fecha: 06 marzo 2015

¿Qué es jQuery?

jQuery es una biblioteca de JavaScript rápida, pequeña y rica en funciones .

Hace las cosas como recorrido y manipulación de documentos HTML, gestión de eventos, animación y facilita Ajax con un API fácil de usar que funciona a través de una multitud de navegadores. Con una combinación de versatilidad y capacidad de ampliación, jQuery ha cambiado la forma en que millones de personas escriben JavaScript .

jQuery ha irrumpido con más fuerza como alternativa a Prototype. Su autor original es John Resig, aunque como sucede con todas las librerías exitosas, actualmente recibe contribuciones de decenas de programadores. jQuery también ha sido programada de forma muy eficiente y su versión comprimida apenas ocupa 20 KB.

jQuery consiste en un único fichero JavaScript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX. La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX. Para ello utiliza las funciones $() o jQuery().

Inicio de jQuery

Comúnmente antes de realizar cualquier acción en el documento con jQuery(), debemos percatarnos de que el documento esté listo. Para ello usamos $(document).ready();, de esta forma:

```
$(document).ready(function() {
  //Aquí van todas las acciones del documento.
});
```

Función $()

La forma de interactuar con la página es mediante la función $(), un alias de jQuery(), que recibe como parámetro una expresión CSS o el nombre de una etiqueta HTML y devuelve todos los nodos (elementos) que concuerden con la expresión. Esta expresión es denominada selector en la terminología de jQuery.

```
$("#tablaAlumnos"); // Devolverá el elemento con id="tablaAlumnos"
$(".activo");      // Devolverá una matriz de elementos con class="activo"
```

Una vez obtenidos los nodos, se les puede aplicar cualquiera de las funciones que facilita la biblioteca.

```
// Se elimina el estilo (con removeClass()) y se aplica uno nuevo (con addClass()) a todos los nodos con class="activo"
$(".activo").removeClass("activo").addClass("inactivo");
```

En gráficos

```
// Anima todos los componentes con class="activo"
$(".activo").slideToggle("slow");
```

FUNCIONES Jquery

Las funciones son la unidad básica de acción en jQuery. El principal punto de entrada de la mayoría de las aplicaciones de jQuery es un bloque de código que se ve así:

```
$(document).ready(function() {
   Realiza una acción
});
```

- $(document) es un objeto de jQuery. La parte de $() es en realidad una función disfrazada; convierte a document en un objeto de jQuery.
- .ready() es una especie de función; imagina que es como un ayudante que ejecuta el código que tiene al interior de sus paréntesis tan pronto como el documento de HTML está listo.
- function(){} es la acción que ejecutará .ready() tan pronto como se cargue el documento de HTML. (En el ejemplo anterior, la parte Realiza una acción es el lugar donde va dicha acción.)

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Resultado</title>
        <link rel='stylesheet' type='text/css' href='stylesheet
.css'/>
        <script type='text/javascript' src='script.js'></script>
    </head>
    <body>
        <div></div>
    </body>
</html>
```

stylesheet.css

```
div {
    height: 100px;
    width: 100px;
    background-color: #FA6900;
    border-radius: 5px;
}
```

script.js

```
$(document).ready(function() {
    $('div').hide();
});
```

Una función está compuesta de tres partes: la palabra clave function, los argumentos que la función tome (que van en medio de () y se separan por comas, si se trata de más de uno), y cualquier acción que la función deba ejecutar (que van en medio de {}). La forma general es:

```
function(argumento1, argumento2, etc.) {
   Realiza una acción
   Realiza otra acción
   ¡Realiza otra acción más!
}
```

Una de las cosas buenas de jQuery es que el argumento de una función puede ser casi cualquier cosa, ¡incluso puede ser otra función! Es por eso que.ready() puede incluir a function en medio de sus paréntesis; está tomando como argumento a una función.

La función básica de jQuery y una de las más útiles tiene el mismo nombre que en Prototype, ya que se trata de la "función dolar": $(). A diferencia de la función de Prototype, la de jQuery es mucho más que un simple atajo mejorado de la función document.getElementById().

La cadena de texto que se pasa como parámetro puede hacer uso de Xpath o de CSS para seleccionar los elementos. Además, separando expresiones con un carácter "," se puede seleccionar un número ilimitado de elementos.

```
// Selecciona todos los enlaces de la página

$('a')

// Selecciona el elemento cuyo id sea "primero"

$('#primero')

// Selecciona todos los h1 con class "titular"

$('h1.titular')

// Selecciona todo lo anterior

$('a, #primero, h1.titular')
```

Las posibilidades de la función $() van mucho más allá de estos ejemplos sencillos, ya que soporta casi todos los selectores definidos por CSS 3 (algo que dispondrán los navegadores dentro de varios años) y también permite utilizar XPath:

```
// Selecciona todos los párrafos de la página que tengan al menos un enlace
$('p[a]')
// Selecciona todos los radiobutton de los formularios de la página
$('input:radio')
// Selecciona todos los enlaces que contengan la palabra "Imprimir"
 $('a:contains("Imprimir")');
// Selecciona los div que no están ocultos
$('div:visible')
// Selecciona todos los elementos pares de una lista
$("ul#menuPrincipal li:even")
// Selecciona todos los elementos impares de una lista
$("ul#menuPrincipal li:odd")
// Selecciona los 5 primeros párrafos de la página
$("p:lt(5)")
```

10.3.2. Funciones para eventos

10.3.3. Funciones para efectos visuales

10.3.4. Funciones para AJAX

10.3.5. Funciones para CSS

10.3.6. Funciones para nodos DOM

10.3.7. Otras funciones útiles

10.3.2. Funciones para eventos

Una de las utilidades más interesantes de jQuery está relacionada con el evento `onload` de la página. Las aplicaciones web más complejas suelen utilizar un código similar al siguiente para iniciar la aplicación:

```
window.onload = function() {

  ...

};
```

Hasta que no se carga la página, el navegador no construye el árbol DOM, lo que significa que no se pueden utilizar funciones que seleccionen elementos de la página, ni se pueden añadir o eliminar elementos. El problema de `window.onload` es que el navegador espera a que la página se cargue completamente, incluyendo todas las imágenes y archivos externos que se hayan enlazado.

jQuery propone el siguiente código para ejecutar las instrucciones una vez que se ha cargado la página:

```
$(document).ready(function() {

  ...

});
```

La gran ventaja del método propuesto por jQuery es que la aplicación no espera a que se carguen todos los elementos de la página, sino que sólo espera a que se haya descargado el contenido HTML de la página, con lo que el árbol DOM ya está disponible para ser manipulado. De esta forma, las aplicaciones JavaScript desarrolladas con jQuery pueden iniciarse más rápidamente que las aplicaciones JavaScript tradicionales.

En realidad, `ready()` no es más que una de las muchas funciones que componen el módulo de los eventos. Todos los eventos comunes de JavaScript (`click`, `mousemove`, `keypress`, etc.) disponen de una función con el mismo nombre que el evento. Si se utiliza la función sin argumentos, se ejecuta el evento:

```
// Ejecuta el evento 'onclick' en todos los párrafos de la página
$('p').click();

// Ejecuta el evento 'mouseover' sobre un 'div' con id 'menu'
$('div#menu').mouseover();
```

No obstante, el uso más habitual de las funciones de cada evento es el de establecer la función manejadora que se va a ejecutar cuando se produzca el evento:

```
// Establece la función manejadora del evento 'onclick'
// a todos los párrafos de la página
$('p').click(function() {
  alert($(this).text());
});

// Establece la función manejadora del evento 'onblur'
// a los elementos de un formulario
$('#elFormulario :input').blur(function() {
  valida($(this));
});
```

Entre las utilidades definidas por jQuery para los eventos se encuentra la función `toggle()`, que permite ejecutar dos funciones de forma alterna cada vez que se pincha sobre un elemento:

```
$("p").toggle(function(){
  alert("Me acabas de activar");
},function(){
  alert("Me acabas de desactivar");
});
```

En el ejemplo anterior, la primera vez que se pincha sobre el elemento (y todas las veces impares), se ejecuta la primera función y la segunda vez que se pincha el elemento (y todas las veces pares) se ejecuta la segunda función.

10.3.3. Funciones para efectos visuales

Las aplicaciones web más avanzadas incluyen efectos visuales complejos para construir interacciones similares a las de las aplicaciones de escritorio. jQuery incluye en la propia librería varios de los efectos más comunes:

```
// Oculta todos los enlaces de la página
$('a').hide();


// Muestra todos los 'div' que estaban ocultos
$('div:hidden').show();


// Muestra los 'div' que estaba ocultos y oculta
// los 'div' que eran visibles
$('div').toggle();
```

Todas las funciones relacionadas con los efectos visuales permiten indicar dos parámetros opcionales: el primero es la duración del efecto y el segundo parámetro es la función que se ejecuta al finalizar el efecto visual.

Otros efectos visuales incluidos son los relacionados con el fundido o *"fading"* (`fadeIn()` muestra los elementos con un fundido suave, `fadeOut()` oculta los elementos con un fundido suave y `fadeTo()`establece la opacidad del elemento en el nivel indicado) y el despliegue de elementos (`slideDown()`hace aparecer un elemento desplegándolo en sentido descendente, `slideUp()` hace desaparecer un elemento desplegándolo en sentido ascendente, `slideToggle()` hace desaparecer el elemento si era visible y lo hace aparecer si no era visible).

10.3.4. Funciones para AJAX

Como sucede con Prototype, las funciones y utilidades relacionadas con AJAX son parte fundamental de jQuery. El método principal para realizar peticiones AJAX es `$.ajax()` (importante no olvidar el punto entre `$` y `ajax`). A partir de esta función básica, se han definido otras funciones relacionadas, de más alto nivel y especializadas en tareas concretas: `$.get()`, `$.post()`, `$.load()`, etc.

La sintaxis de `$.ajax()` es muy sencilla:

```
$.ajax(opciones);
```

Al contrario de lo que sucede con Prototype, la URL que se solicita también se incluye dentro del array asociativo de opciones. A continuación se muestra el mismo ejemplo básico que se utilizó en Prototype realizado con `$.ajax()`:

```
$.ajax({
  url: '/ruta/hasta/pagina.php',
  type: 'POST',
  async: true,
  data: 'parametro1=valor1&parametro2=valor2',
  success: procesaRespuesta,
  error: muestraError
});
```

La siguiente tabla muestra todas las opciones que se pueden definir para el método `$.ajax()`:

Opción	**Descripción**
`async`	Indica si la petición es asíncrona. Su valor por defecto es `true`, el habitual para las peticiones AJAX

Opción	Descripción
`beforeSend`	Permite indicar una función que modifique el objeto `XMLHttpRequest` antes de realizar la petición. El propio objeto `XMLHttpRequest` se pasa como único argumento de la función
`complete`	Permite establecer la función que se ejecuta cuando una petición se ha completado (y después de ejecutar, si se han establecido, las funciones de `success` o `error`). La función recibe el objeto `XMLHttpRequest` como primer parámetro y el resultado de la petición como segundo argumento
`contentType`	Indica el valor de la cabecera `Content-Type` utilizada para realizar la petición. Su valor por defecto es `application/x-www-form-urlencoded`
`data`	Información que se incluye en la petición. Se utiliza para enviar parámetros al servidor. Si es una cadena de texto, se envía tal cual, por lo que su formato debería ser `parametro1=valor1&parametro2=valor2`. También se puede indicar un array asociativo de pares clave/valor que se convierten automáticamente en una cadena tipo *query string*
`dataType`	El tipo de dato que se espera como respuesta. Si no se indica ningún valor, jQuery lo deduce a partir de las cabeceras de la respuesta. Los posibles valores son: `xml` (se devuelve un documento XML correspondiente al valor `responseXML`), `html` (devuelve directamente la respuesta del servidor mediante el valor `responseText`), `script` (se evalúa la respuesta como si fuera JavaScript y se devuelve el resultado) y `json` (se evalúa la respuesta como si fuera JSON y se devuelve el objeto JavaScript generado)
`error`	Indica la función que se ejecuta cuando se produce un error durante la petición. Esta función recibe el objeto `XMLHttpRequest` como primer parámetro, una cadena de texto indicando el error como segundo parámetro y un objeto con la

Opción	Descripción
	excepción producida como tercer parámetro
`ifModified`	Permite considerar como correcta la petición solamente si la respuesta recibida es diferente de la anterior respuesta. Por defecto su valor es `false`
`processData`	Indica si se transforman los datos de la opción `data` para convertirlos en una cadena de texto. Si se indica un valor de `false`, no se realiza esta transformación automática
`success`	Permite establecer la función que se ejecuta cuando una petición se ha completado de forma correcta. La función recibe como primer parámetro los datos recibidos del servidor, previamente formateados según se especifique en la opción `dataType`
`timeout`	Indica el tiempo máximo, en milisegundos, que la petición espera la respuesta del servidor antes de anular la petición
`type`	El tipo de petición que se realiza. Su valor por defecto es `GET`, aunque también se puede utilizar el método `POST`
`url`	La URL del servidor a la que se realiza la petición

Además de la función `$.ajax()` genérica, existen varias funciones relacionadas que son versiones simplificadas y especializadas de esa función. Así, las funciones `$.get()` y `$.post()` se utilizan para realizar de forma sencilla peticiones `GET` y `POST`:

```
// Petición GET simple
```

```
$.get('/ruta/hasta/pagina.php');


// Petición GET con envío de parámetros y función que
// procesa la respuesta
$.get('/ruta/hasta/pagina.php',
  { articulo: '34' },
  function(datos) {
    alert('Respuesta = '+datos);
  });
```

Las peticiones `POST` se realizan exactamente de la misma forma, por lo que sólo hay que cambiar`$.get()` por `$.post()`. La sintaxis de estas funciones son:

```
$.get(url, datos, funcionManejadora);
```

El primer parámerto (`url`) es el único obligatorio e indica la URL solicitada por la petición. Los otros dos parámetros son opcionales, siendo el segundo (`datos`) los parámetros que se envían junto con la petición y el tercero (`funcionManejadora`) el nombre o el código JavaScript de la función que se encarga de procesar la respuesta del servidor.

La función `$.get()` dispone a su vez de una versión especializada denominada `$.getIfModified()`, que también obtiene una respuesta del servidor mediante una petición `GET`, pero la respuesta sólo está disponible si es diferente de la última respuesta recibida.

jQuery también dispone de la función `$.load()`, que es idéntica a la función `Ajax.Updater()` de Prototype. La función `$.load()` inserta el contenido de la respuesta del servidor en el elemento de la página que se indica. La forma de indicar ese elemento es lo que diferencia a jQuery de Prototype:

```
<div id="info"></div>
```

```
// Con Prototype
new Ajax.Updater('info', '/ruta/hasta/pagina.php');

// Con jQuery
$('#info').load('/ruta/hasta/pagina.php');
```

Al igual que sucedía con la función `$.get()`, la función `$.load()` también dispone de una versión específica denominada `$.loadIfModified()` que carga la respuesta del servidor en el elemento sólo si esa respuesta es diferente a la última recibida.

Por último, jQuery también dispone de las funciones `$.getJSON()` y `$.getScript()` que cargan y evalúan/ejecutan respectivamente una respuesta de tipo JSON y una respuesta con código JavaScript.

10.3.5. Funciones para CSS

jQuery dispone de varias funciones para la manipulación de las propiedades CSS de los elementos. Todas las funciones se emplean junto con una selección de elementos realizada con la función `$()`.

Si la función obtiene el valor de las propiedades CSS, sólo se obtiene el valor de la propiedad CSS del primer elemento de la selección realizada. Sin embargo, si la función establece el valor de las propiedades CSS, se establecen para todos los elementos seleccionados.

```
// Obtiene el valor de una propiedad CSS
// En este caso, solo para el primer 'div' de la página
$('div').css('background');

// Establece el valor de una propiedad CSS
// En este caso, para todos los 'div' de la página
$('div').css('color', '#000000');
```

```
// Establece varias propiedades CSS
// En este caso, para todos los 'div' de la página
$('div').css({ padding: '3px', color: '#CC0000' });
```

Además de las funciones anteriores, CSS dispone de funciones específicas para obtener/establecer la altura y anchura de los elementos de la página:

```
// Obtiene la altura en píxel del primer 'div' de la página
$('div').height();

// Establece la altura en píxel de todos los 'div' de la página
$('div').height('150px');

// Obtiene la anchura en píxel del primer 'div' de la página
$('div').width();

// Establece la anchura en píxel de todos los 'div' de la página
$('div').width('300px');
```

10.3.6. Funciones para nodos DOM

La función `$()` permite seleccionar elementos (nodos DOM) de la página de forma muy sencilla. jQuery permite, además, seleccionar nodos relacionados con la selección realizada. Para seleccionar nodos relacionados, se utilizan funciones de filtrado y funciones de búsqueda.

Los filtros son funciones que modifican una selección realizada con la función `$()` y permiten limitar el número de nodos devueltos.

La función `contains()` limita los elementos seleccionados a aquellos que contengan en su interior el texto indicado como parámetro:

```
// Sólo obtiene los párrafos que contengan la palabra 'importante'
$('p').contains('importante');
```

La función `not()` elimina de la selección de elementos aquellos que cumplan con el selector indicado:

```
// Selecciona todos los enlaces de la página, salvo el que
// tiene una 'class' igual a 'especial'
$('a').not('.especial');
// La siguiente instrucción es equivalente a la anterior
$('a').not($('.especial'));
```

La función `filter()` es la inversa de `not()`, ya que elimina de la selección de elementos aquellos que no cumplan con la expresión indicada. Además de una expresión, también se puede indicar una función para filtrar la selección:

```
// Selecciona todas las listas de elementos de la página y quedate
// sólo con las que tengan una 'class' igual a 'menu'
$('ul').filter('.menu');
```

Una función especial relacionada con los filtros y buscadores es `end()`, que permite volver a la selección original de elementos después de realizar un filtrado de elementos. La documentación de jQuery incluye el siguiente ejemplo:

```
$('a')
  .filter('.pinchame')
  .click(function(){
    alert('Estás abandonando este sitio web');
  })
  .end()
  .filter('ocultame')
    .click(function(){
      $(this).hide();
```

```
        return false;
    })
.end();
```

El código anterior obtiene todos los enlaces de la página `$('a')` y aplica diferentes funciones manejadoras del evento `click` en función del tipo de enlace. Aunque se podrían incluir dos instrucciones diferentes para realizar cada filtrado, la función `end()` permite encadenar varias selecciones.

El primer filtrado (`$('a').filter('.pinchame')`) selecciona todos los elementos de la página cuyo atributo `class` sea igual a `pinchame`. Después, se asigna la función manejadora para el evento de pinchar con el ratón mediante la función `click()`.

A continuación, el código anterior realiza otro filtrado a partir de la selección original de enlaces. Para volver a la selección original, se utiliza la función `end()` antes de realizar un nuevo filtrado. De esta forma, la instrucción `.end().filter('ocultame')` es equivalente a realizar el filtrado directamente sobre la selección original `$('a').filter('.ocultame')`.

El segundo grupo de funciones para la manipulación de nodos DOM está formado por los buscadores, funciones que buscan/seleccionan nodos relacionados con la selección realizada. De esta forma, jQuery define la función `children()` para obtener todos los *nodos hijo* o descendientes del nodo actual,`parent()` para obtener el *nodo padre* o nodo ascendente del nodo actual (`parents()` obtiene todos los ascendentes del nodo hasta la raíz del árbol) y `siblings()` que obtiene todos los *nodos hermano* del nodo actual, es decir, todos los nodos que tienen el mismo *nodo padre* que el nodo actual.

La navegación entre *nodos hermano* se puede realizar con las funciones `next()` y `pev()` que avanzan o retroceden a través de la lista de *nodos hermano* del nodo actual.

Por último, jQuery también dispone de funciones para manipular fácilmente el contenido de los nodos DOM. Las funciones `append()` y `prepend()` añaden el

contenido indicado como parámetro al principio o al final respectivamente del contenido original del nodo.

Las funciones `after()` y `before()` añaden el contenido indicado como parámetro antes de cada uno de los elementos seleccionados. La función `wrap()` permite *"envolver"* un elemento con el contenido indicado (se añade parte del contenido por delante y el resto por detrás).

La función `empty()` vacía de contenido a un elemento, `remove()` elimina los elementos seleccionados del árbol DOM y `clone()` copia de forma exacta los nodos seleccionados.

10.3.7. Otras funciones útiles

jQuery detecta automáticamente el tipo de navegador en el que se está ejecutando y permite acceder a esta información a través del objeto `$.browser`:

```
$.browser.msie;    // 'true' para navegadores de la familia Internet Explore
r

$.browser.mozilla; // 'true' para navegadores de la familia Firefox

$.browser.opera;   // 'true' para navegadores de la familia Opera

$.browser.safari;  // 'true' para navegadores de la familia Safari
```

Recorrer arrays y objetos también es muy sencillo con jQuery, gracias a la función `$.each()`. El primer parámetro de la función es el objeto que se quiere recorrer y el segundo parámetro es el código de la función que lo recorre (a su vez, a esta función se le pasa como primer parámetro el índice del elemento y como segundo parámetro el valor del elemento):

```
// Recorrer arrays

var vocales = ['a', 'e', 'i', 'o', 'u'];


$.each( vocales, function(i, n){

  alert('Vocal número ' + i + " = " + n);
```

```
});

// Recorrer objetos
var producto = { id: '12DW2', precio: 12.34, cantidad: 5 };

$.each( producto, function(i, n){
  alert(i + ' : ' + n);
});
```