Tema 6: Librería de JavaScript: JQuery

Autor: Roberto Marín

Asignatura: Desarrollo Web Entorno Cliente (DAW2)

Año: 2017/2018

Índice:

6.1. Primer paso con JQuery

Vamos a emplear dos formas de usar JQuery.

6.1.1 En nuestro servidor

Simplemente descargaremos desde el sitio oficial y lo incluiremos como cualquier otro archivo .js

Nos encontraremos con 2 versiones de JQuery, por un lado la de producción que está comprimida y optimizada pero es difícil de leer y por otro la completa que es legible y nos permite estudiar el código. http://jquery.com/download/ Una vez descargada tan solo tendremos que incluirla con un código similar a este:

```
<head>
<script src="jquery-3.2.1.min.js"></script>
</head>
```

6.1.2 Usando CDN (Content Delivery Network)

Podremos hacer uso de un CDN, los mas usuales son los de Google y Microsoft.

- ☑ https://developers.google.com/speed/libraries/#jquery
- ☑ https://docs.microsoft.com/en-us/aspnet/ajax/cdn/overview#jQuery_Releases_on_the_CDN_0

Si elegimos Google tan solo tendremos que añadir (por ejemplo):

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
```

A nuestro documento HTML.

6.2 Compatibilidad con otras librerías

Como veremos JQuery hace uso del símbolo: `$` como alias. Para JS el símbolo `$` no es especial pero por alguna razón (que desconozco) ha sido elegido por varias librerías de JS. Por tanto, si nuestro proyecto va a hacer uso de varias librerías tendremos que usar el modo *no-conflict* de JQuery.

Veamos un ejemplo sencillo de uso de JQuery: 6.1.html

Si ahora cargamos la librería prototypejs el resultado será un error.

6.2.html

En la consola veremos:

`TypeError: $(...).ready is not a function`
https://cursosgs.es/dwec/wp-content/uploads/2017/11/Ejemplo2.html

Para solucionarlo haremos uso del modo no-conflict de JQuery.
En nuestro head habremos cargado las 2 librerías:

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<script
src="https://ajax.googleapis.com/ajax/libs/prototype/1.7.3.0/prototype.js"></script>
```

Para poder usar JQuery en la primera línea de nuestro `script` introduciremos algo como:

```
<script>
var aliasJQuery=jQuery.noConflict();
//podemos poner el alias que mas nos guste. A partir de aqui lo usamos asi:
aliasJQuery(document).ready(function(){
    aliasJQuery('#boton1').click(function(){
        aliasJQuery('#p1').html("Prueba de jQuery en modo no-conflict");
        });
    });
//normalmente usaremos un alias corto del tipo: 'jq'.
</script>
```

Ver 6.3.html

6.2.1 ¿Cómo mantener $?

En ocasiones nos va a interesar mantener el uso de $, normalmente porque aprovechamos código anterior. Para ello usaremos una función anónima autoejecutable así:

```
<script>
(function($) {
   // nuestro código, usando $ sin problemas.
})(jQuery);
</script>
```

Ver 6.4.html

6.3 Uso de selectores I

Enlace: http://api.jquery.com/category/selectors

Una de las ventajas de JQuery es que aglutina los métodos selectores del DOM. Así que podemos olvidarnos de *getElementById*, *getElementByClassName* y similares.

JS Vanilla tiene un equivalente completamente funcional con los métodos: `document.querySelector()` y `document.querySelectorAll()`, pero no vamos a pararnos aquí a discutir sobre si tiene sentido o no usar jQuery.

En su lugar vamos a usar $(<denominación del elemento>) bien por id, clase o atributo. Aquí veremos los más usuales y algunos ejemplos.

6.3.1 Selección básica de elementos

- ☑ Por id:

```
var $foo=$('#idfoo');
//equivalente a:
var i=document.getElementById('idfoo');
```

- ☑ Por clase:

```
var $bar=$('.clasebar');
//equivalente a:
var i=document.getElementsByClassName('clasebar');
```

- ☑ Por tag:

```
var $foobar=$('p');
//equivalente a:
```

```
var i=document.getElementsByTagName('p');
```

Por convenio se suele comenzar con `$` las variables que resultan de un selector jQuery.

En este punto habréis encontrado una diferencia importante entre los métodos de JS puro y los de JQuery. Pues si bien la selección de elementos por id es similar, cuando buscamos por clase o tag el comportamiento no es el mismo. Así si hacemos un *log* de `$bar` o `$foobar` nos mostrará el contenido del primer párrafo independientemente de cuantos haya con la misma clase. Veamos un ejemplo:

Para un documento html con el fragmento siguiente:

```
    <p>Párrafo sin clase ni id.</p>
    <p id="idfoo">Párrafo perteneciente a idfoo</p>
    <p class="clasebar">Párrafo 1 perteneciente a la clase: clasebar.</p>
    <p class="clasebar">Párrafo 2 perteneciente a la clase: clasebar.</p>
    <p class="clasebar">Párrafo 3 perteneciente a la clase: clasebar.</p>
```

Si (con JQuery cargado conveniente) ejecutamos JS

```
    var $bar=$('.clasebar');
    console.log($bar.html()); //Devuelve 'Párrafo 1 perteneciente a la clase:
clasebar.'
```

Nos mostrará el contenido del primer párrafo que encuentre con la clase.

Si, pensando en lo conocido, vemos a $bar como un vector, podremos hacer:

```
    console.log($bar[1].innerHTML); //Devuelve 'Párrafo 2 perteneciente a la clase:
clasebar.'
    console.log($bar[1].html()); //Da error
```

Sin embargo como vemos en el retorno del código anterior usar la sintaxis de array hace que no podamos usar el método `.html()`, que pertenece a JQuery, debido a que no lo reconoce para el objeto.
Por tanto si bien podemos recorrer con un for *clásico* la selección, JQuery nos ofrece un método propio: el método `.each()` que hemos usado en el siguiente ejemplo:

Veremos el metodo .each() en profundidad más adelante. No se pretende ahora que se entienda todo, tan sólo mostrar como conseguir efectos análogos a los de métodos conocidos.

Ejercicio 6.5:

Dada la Página Web 6.5.html, muestra por consola las siguientes operaciones:

a) Párrafo perteneciente a idfoo

b) Párrafo 1 perteneciente a la clase: clasebar.

c) Párrafo sin clase ni id.

d) Párrafo 2 perteneciente a la clase: clasebar.

e) Empleando el método each como si fuera un for: $bar.each(function(index,value)

En índex se recibe el índice del array y en value se recibe el objeto y mostrar:

El id es: p1

Párrafo 1 perteneciente a la clase: clasebar.

El id es: p2

Párrafo 2 perteneciente a la clase: clasebar.

El id es: p3

Párrafo 3 perteneciente a la clase: clasebar.

El id es: p4

Párrafo 4 perteneciente a la clase: clasebar.

El id es: p5

Párrafo 5 perteneciente a la clase: clasebar.

6.3.2 document.ready

En el ejemplo 6.5.html se usa la estructura:

```
$(document).ready(function(){
    //bloque de código
});
```

- ☑ Esta es la forma que tiene JQuery para prevenir que se ejecute el código antes de que el documento HTML esté correctamente cargado. Lo usaremos prácticamente en todos los ejemplos. Es el equivalente aproximado a:

```
window.onload=function(){
   // bloque de código
};
```

- ☑ Además de la forma `$(document).ready`, podemos usar la forma resumida:

```
$(function() {
   // bloque de código
});
```

- ☑ pasarle una función nominal:

```
function funcionInicio(){
   // bloque de código
}
$(document).ready(funcionInicio);
```

6.4 Uso de selectores II

En el apartado anterior hemos visto las selecciones más básicas con JQuery de elementos en HTML. Vamos a ver ahora ejemplos de selecciones más avanzadas que van a permitirnos filtrar los elementos seleccionados.

Deberemos tener en cuenta para lo no explicado aquí la documentación de referencia: http://api.jquery.com/category/selectors

JQuery usa los selectores CSS para realizar sus selecciones, por tanto podremos realizar todo tipo de combinaciones. Sin embargo hay que tener en cuenta que en ocasiones esto puede ocasionar que el proceso sea muy largo.

6.4.1 Selección en base a CSS

- ☑ Veamos un ejemplo: seleccionamos los elementos `<p>` de clase `.clase1`.
  `$('p.clase1');`

- ☑ Selecciona los elementos de tipo `<p>` dentro del `<div>` con `id="contenido"`
  `$('div#contenido p')`

- ☑ Selección de elementos por su atributo
  `$('input[name=first_name]');`

- ☑ Selección de elementos en forma de selector CSS
  `$('#contents ul.people li');`

6.4.2 Pseudo-selectores

- ☑ Selecciona el primer elemento <a> con la clase 'external'
  `$('a.external:first');`

- ☑ Selecciona todos los elementos <tr> impares de una tabla
  `$('tr:odd');`

- ☑ Selecciona todos los elementos del tipo input dentro del formulario #myForm
  `$('#myForm: input');`

- ☑ Selecciona todos los divs visibles `$('div:visible');`

- ☑ Selecciona todos los divs excepto los tres primeros `$('div:gt(2)');`

- ☑ Selecciona todos los divs actualmente animados `$('div:animated');`

Ejercicio 6.6condicionales.html

Compara dos números y saca por pantalla si: el primero es mayor, menor o igual al segundo. Para ello:

- ☑ Mediante JQuery selecciona los elementos por medio de su id

- ☑ Consigue el valor de cada uno de ellos empleando el método .val()

- ☑ Sin emplear funciones, comprueba si son mayores, menores o iguales.

- ☑ Saca por pantalla el resultado.

Ejercicio 6.7funciones.html:

Concatena dos números de dos diferentes inputs. Para ello

- ☑ Mediante JQuery, selecciona cada uno de los inputs

- ☑ Con una funciona llamada Calcula, pásale cada uno de los dos valores y concaténalos con el método .append()

- ☑ Mostrar el resultado por pantalla.

6.5 Efectos

En esta entrada vamos a tratar los efectos incorporados a la librería jQuery. En la próxima entrada veremos como realizar nuestros propios efectos. Podemos ver todos los efectos en la documentación oficial:

https://api.jquery.com/category/effects

Los efectos incorporados más habituales son los siguientes:

- ☑ .hide(), .show() y .toggle()

Esconden y muestran, respectivamente, el elemento seleccionado. Para ello asignan/desasignan la propiedad display del elemento a none.

```
$('#d1').hide();

$('#d1').hide(1000); //modificamos la duración del efecto a 1000 milisegundos.

$('#d1').show();

$('#d1').show('fast'); //modificamos la duración del efecto al valor de 'fast', que es por defecto

200 milisegundos.

$('#d1').toggle('slow); //haremos aparecer y desaparecer la selección con velocidad lenta.
```

- ☑ .fadeIn(), .fadeOut() y .fadeTo()

Modifican de manera animada la opacidad del elemento seleccionado. .fadeIn() le asigna un valor de 100% y .fadeOut() de 0%. Con .fadeTo() vamos a poder asignar una opacidad concreta.

```
$('#d2').fadeIn(); //fundido estandar a opacidad 100%

$('#d2').fadeOut('slow'); //se desvanece con velocidad 'slow'

$('#d2').fadeTo(400,0.3); //duración 400ms y opacidad final establecida en el 30%
```

Nótese que después de aplicar .fadeTo() se cambia el efecto de .fadeIn().

- ☑ .slideUp(), .slideDown() y .slideToggle()

Muestran, ocultan o alternan, respectivamente, el elemento de la selección.

.slideUp lo oculta con un desplazamiento hacia arriba;

.slideDown() muestra un elemento oculto con un desplazamiento hacia abajo.

.slideToggle() actuará como los anteriores; mostrando si está oculto y ocultando si está visible.

```
$('#d3').slideUp('slow'); // oculta hacia arriba con velocidad lenta

$('#d3').slideDown('fast'); // muestra hacia abajo con velocidad rápida

$('#d3').slideToggle(1500);  // alterna mostrado  o no con el efecto anterior con un tiempo de 1500 ms.
```

Ejercicio 6.8enunciado.html:

Dado el archivo 6.8selectoresenunciado.html, crea un script que seleccione elementos de HTML dentro de un formulario, y ejecute un evento de parpadeo sobre los mismos. Para ello:

- ☑ Mediante JQuery selecciona el elemento de HTML
- ☑ Mediante los métodos .fadeOut y .fadeIn, simule el parpadeo (ocultar/mostrar) de los elementos correspondientes.

Ejercicio 6.9Evento clickhide.html:

Dado varios elementos de HTML, eliminarlos de la pantalla pulsando sobre los mismos. Para ello:

- ☑ Selecciona los 3 elementos mediante JQuery
- ☑ Utilizando el método .hide(), ir borrando cada uno de los elementos.

6.6 Estilos, clases, dimensiones y atributos

- ☑ Estilos

Además de acceder (seleccionar) a los elementos de una web, JQuery nos va a permitir acceder y modificar su estilo. Esto vamos a poder hacerlo mediante cambios en el CSS, en sus dimensiones o en sus atributos.

JQuery tiene un ***método que nos va a permitir obtener el valor de una propiedad css de un elemento y también establecerlo***. Su sintaxis es:

$(selección).css(propiedad);

Por ejemplo:

$('p#parrafo1').css('color'); //devuelve el color de ese párrafo

Si la propiedad es del tipo 'background-color' en CSS, sabemos que en JavaScript tendríamos que usar backgroundColor. En JQuery podremos usar ambas notaciones.

- ☑ Clases

Para manejar las clases con JQuery disponemos de los métodos:

Enlace: *https://api.jquery.com/category/manipulation/class-attribute*

- addClass(): añade una clase (o clases **separadas por un espacio**) al elemento seleccionado: $('p').addClass('rojo');
- removeClass(): elimina una clase (o clases) al elemento seleccionado:

$('p').removeClass('rojo');

- hasClass(): devuelve true si el elemento seleccionado tiene asignada esa clase:

if ($('p').hasClass('rojo')){...};

- toogleClass(): alterna **añadir/quitar la clase (o clases**) al elemento seleccionado:

$('p').toogleClass('rojo');

☑ Dimensiones

Podremos obtener/establecer determinadas dimensiones con sus respectivos métodos:

Enlace: http://api.jquery.com/category/dimensions

- .height()
- .width()
- .innerHeight()
- .innerWidth()
- .outerHeight()
- .outerWidth()
- **.position() //** devuelve el valor **(top, left)** desde los márgenes del elemento (incluye margin como parte del elemento) y se posiciona respecto a su elemento padre
- **.offset() //** devuelve **el valor(top, left)** sin contar los margin del elemento y se posiciona respecto al documento.

Por ejemplo:

var $capa1 = $('#capa1');

a) var altura = $capa1.height() //devuelve la altura del elemento

b) $capa1.height('200px') //establece en 200px la altura del elemento

☑ Atributos

JQuery permite establecer/obtener valores de los atributos de elementos HTML con método:

- attr().

Este método admite múltiples atributos.

Ejemplos:

a) console.log($('#enlace').attr('href')); //devuelve el valor de href para #enlace

b) $('#enlace').attr('href','https://www.google.es'); //establece el valor de href para el enlace

c) $('#enlace').attr(

'href':'https://www.unizar.es', 'target':'_blank'); //establece la dirección y el target del enlace

Ejercicio 6.10enunciado.html: Realiza un script con JQuery que realice los siguientes cambios:

- ☑ Añade al elemento <p> una nueva clase “nuevo”
- ☑ Elimina la clase “verde” al elemento con id <p2>
- ☑ Comprueba que div2 tiene una clase “rojo”
- ☑ Establece a la caja div1: altura 100 px y color de fondo verde

6.7 Eventos

Para manejar los eventos con JQuery disponemos de los métodos:

Enlace: https://api.jquery.com/category/events

- ☑ El método .on()

El método .on() sirve para unir *handlers* con elementos, pero además nos va a servir para realizar la delegación o asignar de eventos

La sintaxis del método es:
.on(events [, selector] [, data], handler)

events: Son los eventos que vamos a asociar con la selección, pueden ser 1 o varios separados por espacios.

selector: Es un parámetro opcional, nos va a permitir filtrar los elementos hijos de la selección desde los que queremos que se lance el *handler*. Gracias a él podemos usar la delegación funcional.

data: También opcional, cualquier tipo de dato que necesitamos pasar a la función.

handler: Función que ocurrirá cuando se dispare el evento. Podra ser anónima o no.

Ejercicio 6.11Eventoclickmouse.html:

Selecciona un elemento de tipo párrafo y modifica su color según la acción de nuestro ratón. Para ello:

- ☑ Selecciona mediante JQuery el elemento
- ☑ Dentro del método .on() incluye los 3 eventos con el ratón: mouseenter/mouseleave/click
- ☑ Identifica un color diferente para cada uno de los eventos

Ejercicio Cuadros:

A partir del diseño realizado de los 4 cuadrados realiza el siguiente script.

- ☑ Empleando JQuery y con los métodos vistos en clase, aplica los efectos de ocultar y mostrar los 4 cuadrados. El evento se iniciará en el mismo momento que el ratón pase por encima de alguno de ellos.
- ☑ El orden de los eventos será de izquierda a derecha, empezando por arriba y en sentido de las agujas del reloj. Se ejecutará de forma completa (desaparecer y aparecer) y una sola vez.

6.8. Bibliografía

- ☑ W3Schools Jquery
- ☑ Fundamentos de JQuery
- ☑ jQuery in action – Bear Bibeault and Yehuda Katz

6.9. Agradecimientos y reconocimiento

- ☑ A Javier Soriano, profesor de IES Santiago Hernández, por su aportación en la documentación del presente manual.

- ☑ La mayoría de lo que hay escrito aquí está mejor explicado en multitud de tutoriales en Internet.

- ☑ Las guías básicas para realizar este material están en la bibliografía de referencia. Cuando se hagan citas literales o ejemplos se indicará su autor.