C.F.G.S.: DESARROLLO DE APLICACIONES WEB
Módulo: DESARROLLO WEB EN ENTORNO CLIENTE

05 INTERACCIÓN CON EL DOM

05.1 Desplazamiento por los nodos (transversing)

Además de los filtros que hemos visto que permiten desplazarnos por el DOM seleccionando nodos, jQuery nos ofrece **métodos** específicos del objeto jQuery para navegar por el DOM del documento.

1. Métodos de Filtrado

Muchos de estos son equivalentes a los filtros vistos anteriormente. En general es indiferente utilizar método o filtro, pero en ciertos casos puede ser mejor usar el método porque se gana algo de rapidez.

- **first()**

  Reduce la selección al primer elemento, por lo que es equivalente al filtro :first.

  Ejemplo: `$("p").first().css("background-color","yellow");`

- **last()**

  Reduce la selección al último elemento, por lo que es equivalente al filtro :last.

  Ejemplo: `$("p").last().css("background-color","yellow");`

- **eq(**índice**)**

  Reduce la selección al elemento indicado por el índice, por lo que es equivalente al filtro :eq.

  Ejemplo: `$("p").eq(2).css("background-color","yellow");`

- **filter(**selector**)**

  Reduce la selección a los nodos que son identificados por el selector.

Ejemplo:	*$("p").filter(":even")*	*los párrafos de índice par*
	$("p").filter(":odd")	*los párrafos de índice impar*
	$("p").filter(":animated")	*los párrafos con alguna animación*
	$("p").filter(".clase")	*los párrafos con class "clase"*

- **not(**selector**)**

Es el contrario al anterior. Reduce la selección a los nodos que NO son identificados por el selector.

Ejemplo:	*$("p").not(":even")*	*los párrafos de índice no par*
	$("p").not(".clase")	*los párrafos que no tienen class "clase"*

- **slice(**inicio, [fin]**)**

Reduce la selección a los elementos del objeto jQuery cuyos índices están comprendidos entre *inicio* (incluido) y, el último del objeto (incluido) si no se incluye el argumento *fin*, o hasta el que ocupa el índice *fin* (excluido) si sí se aporta este argumento. Es más eficiente utilizar el método slice que los filtros :gt y :lt.

Ejemplo:	*$("p").slice(0,3)*	*los párrafos de índice 0,1,2*
	$("p").slice(2)	*los párrafos de índice 2 hasta el último*

2. Métodos para acceder a los Descendientes

Un descendiente es un hijo, nieto, bisnieto, etc. Con jQuery se puede recorrer el árbol DOM para encontrar descendientes de un elemento.

- **children (**[selector])

Devuelve un jQuery con los ***hijos (directos, no nietos)*** del jQuery inicial. Si se incluye como argumento un selector sólo se seleccionarán los hijos que puedan ser identificados por éste.

Ejemplo:	*$("div").children()*	*todos los hijos directos de cualquier div*
	$("div").children("p")	*todos los hijos directos párrafos de cualquier div*
	$("div").children("")*	*todos los hijos directos de cualquier div*

- **find (**selector**)**

Devuelve un jQuery con los **descendientes (de cualquier nivel)** del jQuery inicial que pueden ser identificados por el *selector*.

Ejemplo: *$("div").find("*")* *todos los hijos, nietos… de cualquier div*
$("div").find("p") *todos los hijos, nietos… párrafos de cualquier div*

(obligatorio indicarle un selector, find() no funciona)

3. Métodos para acceder a los Ancestros

Un ancestro es un padre, abuelo, bisabuelo, y así sucesivamente. Con jQuery se puede recorrer el árbol DOM para encontrar antepasados de un elemento.

- **parent (** [selector] **)**

Devuelve un jQuery con los padres de los elementos del jQuery inicial. Si se utiliza el argumento opcional selector, los padres sólo se incluirán en el jQuery devuelto si pueden ser identificados por ese selector.

Ejemplo: *$("p").parent()* *padre directo de cada párrafo*
$("p").parent("div") *padre directo de cada párrafo en caso de que sea un div*

- **parents(** [selector] **)**

Devuelve un jQuery con los **ancestros (no sólo los padres)** de los elementos del jQuery inicial. Si se utiliza el argumento opcional selector, los ancestros sólo se incluirán en el jQuery devuelto si pueden ser identificados por ese selector.

Ejemplo: *$("p").parents()* *ancestros de cada párrafo*
$("p").parents("div") *ancestros de cada párrafo en caso de que sea un div*

- **closest (** [selector] **)**

Retorna el primer ancestro del tipo indicado por el selector

Ejemplo: *$("p").closest("div")* *primer ancestro div de cada párrafo*

4. Métodos para acceder a los Hermanos

- **siblings (** [selector] **)**

Devuelve todos los hermanos del elemento seleccionado
Ejemplo: *$("#elem").siblings()* *todos los hermanos del elemento elem*

$("#elem").siblings("h2") *todos los hermanos del elemento elem que son de tipo h2*

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_siblings

- **next (** [selector] **)**

Devuelve el hermano inmediatamente menor del elemento seleccionado

Ejemplo: *$("#elem").next()* *hermano inmediatamente menor del elemento elem*
$("#elem").next("h2") *hermano inmediatamente menor del elemento elem si este es de tipo h2*

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_next

- **nextAll (** [selector] **)**

Devuelve todos los hermanos menores del elemento seleccionado

Ejemplo: *$("#elem").nextAll()* *hermanos menores del elemento elem*
$("#elem").nextAll("h2") *hermanos menores del elemento elem si son de tipo h2*

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_nextall

- **prev (** [selector] **)**

Devuelve el hermano inmediatamente mayor del elemento seleccionado

Ejemplo: *$("#elem").prev()* *hermano inmediatamente mayor del elemento elem*
$("#elem").prev("h2") *hermano inmediatamente mayor del elemento elem si este es de tipo h2*

- **prevAll (** [selector] **)**

Devuelve todos los hermanos mayores del elemento seleccionado

Ejemplo: *$("#elem").prevAll()* *hermanos* mayores *del elemento elem*
$("#elem").prevAll("h2") *hermanos mayores del elemento elem si son de tipo h2*

5. Encadenar métodos de filtrado

Se pueden encadenar filtros y métodos.

$("#div1").find("p").eq(2).html("nuevo texto para el tercer párrafo");

EJEMPLO 05 01 ejemplo desplazamiento.html

05.2 Manipulación de nodos

http://api.jquery.com/category/manipulation/

jQuery nos ofrece 3 tipos de métodos del objeto jQuery para modificar los atributos y propiedades de los nodos:

(Podemos considerar atributos a aquellas especificaciones de un elemento html que toman un valor, por ejemplo: id, class, name, value, href, src, title, style, type … y propiedades a los que toman un valor booleano, true o false, por ejemplo: readonly, disabled, selected, checked… Los metodos que vamos a ver ahora se pueden utilizar en muchos casos tanto para atributos como para propiedades)

- Los generales para cualquier atributo de elemento o propiedad de nodo, que son **attr(), prop(), removeAttr(), removeProp() y val().**

- Los específicos para el atributo class, que son **addClass(), hasClass(), removeClass() y toggleClass()**. Es muy frecuente cambiar el estilo de los elementos modificando su asignación de clases.

- Uno específico para aplicar estilos CSS (afectando al atributo style de los elementos, es decir, aplicando estilos en línea), que es **css().**

Vamos a ir viendo cada uno de ellos:

- **attr**(atributo,*[valor]*)

  Obtiene el valor del atributo del primer elemento del objeto jQuery si no se aporta el atributo opcional *valor*.

  Ejemplo: `alert($("img").attr("width"));` *muestra el width de la primera imagen*

  Si se aporta el argumento opcional *valor*, establece el *valor* del *atributo* (por ejemplo, 'src' o 'alt') en todos los elementos del objeto jQuery.

  Ejemplo: `$("img").attr("width","500");` *pone 500 de anchura a todas las imágenes*

  Si necesitásemos gestionar el atributo de todos los elementos del objeto jQuery, no sólo del primero, podríamos recurrir al método each.

  Ejemplo:
  ```
  $("img").each(function(i){
          $(this).attr("width",(i+1)*100);
  });
  ```
  la primera imagen tendrá de anchura 100, la segunda 200, la tercera 300…

http://api.jquery.com/attr/

- **prop**(atributo,*[valor]*)

Obtiene el valor de la propiedad del primer elemento del objeto jQuery. Si no se aporta el argumento opcional *valor*, establece el *valor* de la *propiedad* (por ejemplo, 'disabled' o 'checked') en todos los elementos del objeto jQuery.

Ejemplo: `$(":checkbox").prop("checked",true);`

https://api.jquery.com/prop/

- **removeAttr**(*atributo*)

Elimina el atributo de todos los elementos del objeto jQuery

Ejemplo: `$("p").removeAttr("style");`

- **removeProp**(*propiedad*)

El método removeProp () elimina una propiedad establecido por el método prop().

Este método NO debe usarse para eliminar propiedades intrínsecas de los elementos (como selected, disabled,…), porque luego no podrán volver a añadirse con el método prop.

- **val**(*[valor]*)

Obtiene el valor del elemento jQuery. Si se aporta el argumento opcional *valor*, establece el atributo value del primer elemento del objeto jQuery. Como afecta al atributo value se utiliza fundamentalmente con controles de formularios como input o textArea.

Ejemplo:
```
$("#campo").val("HOLA");
alert($("#campo").val());
```

- **addClass**(*clase[s]*)

Asigna a los elementos del objeto jQuery la(s) clase(s) indicadas. Las clases deben especificarse como una cadena de caracteres, con espacios que separen los nombres de las clases en caso de que haya más de una. Por ejemplo
$('p').addClass('noticia destacada');

Ejemplo: `$("p:first").addClass("intro");`

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_html_addclass

- **removeClass**(*clase[s]*)

Elimina de los elementos del objeto jQuery la(s) clase(s) indicadas

Ejemplo: `$("p ").removeClass("intro");`

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_html_removeclass

- **hasClass**(*clase*)

Devuelve un valor booleano que indica si alguno (no necesariamente todos) de los elementos del objeto jQuery tiene asignada la *clase* indicada.

Ejemplo: `alert($("p").hasClass("intro"));`

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_html_hasclass

- **toggleClass**(*clase[s]*)

Alterna la asignación de la(s) clase(s) indicada(s) en todos los elementos del objeto jQuery, es decir, si pertenecían a esa clase dejan de hacerlo y, si no, pasan a sí pertenecer a ella. El argumento debe ser una cadena de caracteres con los nombres de las clases separados por espacios.

Ejemplo: `$("p").toggleClass("intro");`

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_html_toggleclass

- **css**(*propiedadCSS,[valor]*)

Obtiene, si no se aporta el argumento opcional valor, o establece, si se aporta, el valor de la propiedad CSS indicada del primer. elemento del objeto jQuery. Este método actúa sobre el valor del atributo style, pero no lo sustituye; por ejemplo, si sobre el elemento aplicamos el método css('border-width','1px'), el borde se aplicará sin eliminar el fondo rojo.

Ejemplo: `$("p").css("color","red");`

EJEMPLO 05 02 ejemplo manipulacion.html

05.3 Crear, Insertar, Eliminar y Copiar elementos

Hasta ahora hemos utilizado la función jQuery() sólo para seleccionar elementos del DOM, pero también sirve para crear elementos. Para ello, en lugar de pasarle como argumento un selector le pasamos un código HTML

```
nuevo= $('<a href="imagen.jpg">Ver imagen</a>').
p = $('<p>');
p.html('Soy un nuevo párrafo');
```

Usando esta sintaxis obtendremos un objeto jQuery con una referencia al elemento creado, pero ese elemento no se incluirá en el DOM porque aún no tiene un padre; para insertarlo en el DOM disponemos de varios tipos de métodos que se podrían clasificar en las siguientes categorías:

1. Inserción como hijos.

- **append**(*objeto_jQuery*)

  Añade el objeto jQuery pasado como argumento al final del elemento al que aplicamos el método

- **appendTo**(*objeto_jQuery*)

  Es el método contrario al anterior; añade el l elemento sobre el que se aplica el método al final de los elementos del objeto jQuery que recibe como argumento.

- **prepend**(*objeto_jQuery*)

  Añade el objeto jQuery pasado como argumento al principio del elemento al que aplicamos el método.

- **prependTo**(*objeto_jQuery*

  Es el método contrario al anterior; añade el elemento sobre el que se aplica el método al principio de los elementos del objeto jQuery que recibe como argumento.

Ejemplo:

```
var uno=$("<li> Nuevo en la lista </li>");
$("ul").append(uno);
 var otro=$("<li> Nuevo en la lista </li>");
otro.appendTo($("ul"));
```

2. Inserción como hermanos.

- **after**(*objeto_jQuery*)

Inserta el objeto jQuery que recibe como argumento después del elemento al que se aplica el método.

- **before**(*objeto_jQuery*)

Inserta el objeto jQuery que recibe como argumento antes del elemento al que se aplica el método.

Ejemplo:
```
var nuevo=$("<p> soy nuevo </p>");
$("#uno").after(nuevo);
```

- **clone()**

Estos métodos desplazan elementos pero no los clonan. Si necesitamos mover duplicados utilizamos **clone()** para duplicar.

```
$("#uno").clone().appendTo($("#otro"));
```

Si se necesita copiar información y eventos relacionados al elemento que se han especificado posteriormente, se debe pasar true como argumento de clone

http://www.w3schools.com/jquery/html_clone.asp

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_html_clone_eventhandler

```
$(document).ready(function(){
  $("button").click(function(){
    $("body").append($("p:first").clone(true));
  });
  $("p").click(function(){
    $(this).animate({fontSize: "+=1px"});
  });
});
```

3. Inserción como padres.

- **wrap**(*codigoHTML* u *objeto_jQuery*)

Anida cada uno de los elementos del objeto jQuery al que se aplica el método como hijo de un elemento nuevo creado a partir de *codigoHTML* o clonado de *objeto_jQuery*, y devuelve el objeto jQuery original, no el de sus nuevos padres.

Ejemplo: `$("p").wrap("<div></div>");`

(A cada p le va a poner como padre un div)

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_html_wrap

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_html_wrap_element

- **unwrap**().

Es el método contrario del anterior; elimina del DOM el padre de cada uno de los elementos del objeto jQuery al que se aplica, de modo que ahora sus padres (y los de sus hermanos si los tuvieran) pasarán a ser los que antes eran sus abuelos.

Ejemplo: `$("p").unwrap();`

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_html_unwrap

- **wrapAll**(*estructuraHTML* u *objeto_jQuery*).

Crea un objeto jQuery a partir de la estrucutraHTML o clona el objeto_jQuery que recibe como argumento, y anida dentro de ese objeto (creado o clonado) todos los elementos del objeto jQuery que recibe como argumento.

Ejemplo: `$("p").wrapAll("<div></div>");`

(Va a poner un mismo div para todos los p)

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_html_wrapall

4. Reemplazar elementos

Otra forma de modificación del DOM es el reemplazo de elementos, que podemos realizar con los siguientes métodos:

- **replaceAll**(objeto_jQuery)

Utiliza los elementos del objeto sobre el que se aplica el método para sustituir a cada uno de los del objeto jQuery que recibe como argumento.

Ejemplo: `$("<h2>Esto eran antes párrafos</h2>").replaceAll("p");`

Reemplaza todos los p por estos h2

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_html_replaceall

- **replaceWith**(objeto_jQuery).

Es el método inverso del anterior, es decir, utiliza los elementos del objeto jQuery que recibe como argumento para sustituir a cada uno de los del objeto jQuery sobre el que se aplica.

Ejemplo: `$("p:first").replaceWith("Este era el primer párrafo");`

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_html_replacewith

- **html**(*[códigoHTML]*)

Este método es el equivalente a la propiedad innerHTML del DOM, de modo que, si no se aporta el argumento opcional, devuelve el código HTML que contiene el primer elemento del objeto jQuery al que se aplica, pero si sí se aporta ese argumento reemplaza el contenido de cada elemento del objeto jQuery sobre el que se aplica por el *códigoHTML*.

- **text**(*[cadenaCaracteres]*)

Si no se aporta el argumento opcional, devuelve una cadena de caracteres que es la combinación de todos los nodos de texto que contiene (a cualquier nivel de profundidad; hijos, nietos, …) cada elemento del objeto jQuery sobre el que se aplica. Si sí se aporta el argumento, sustituye el contenido de cada elemento del objeto jQuery sobre el que se aplica por *cadenaCaracteres*; tenga en cuenta que jQuery escapa automáticamente los caracteres de esta cadena para que sean aceptables en HTML (en otras palabras: si escribe código HTML en cadenaCaracteres, los signos < y > se reemplazarán por las entidades < y >, respectivamente, de modo que el código no se interpretará como HTML, sino que se mostrará directamente dentro del elemento).

```
$("p").text("Hola caracola!");
$("li").text("<p>adios </p>"); muestra literalmente <p>adiós</p>
```

5. Otros Métodos

El hecho de que los objetos jQuery puedan estar asociados o no al DOM da lugar a que existan varios métodos diferentes para eliminar elementos:

- **remove**(*[selector]*)

Este método elimina definitivamente los elementos del objeto jQuery al que se aplica y que pueden ser identificados por el *selector* opcional.

```
$("li").last().remove();
```

- **detach**(*[selector]*)

  Este método elimina del DOM los elementos del objeto jQuery al que se aplica y que pueden ser identificados por el selector opcional, pero devuelve esos elementos en un nuevo objeto jQuery no asociado al DOM por si queremos conservarlos en una variables para re-asociarlos al DOM posteriormente.

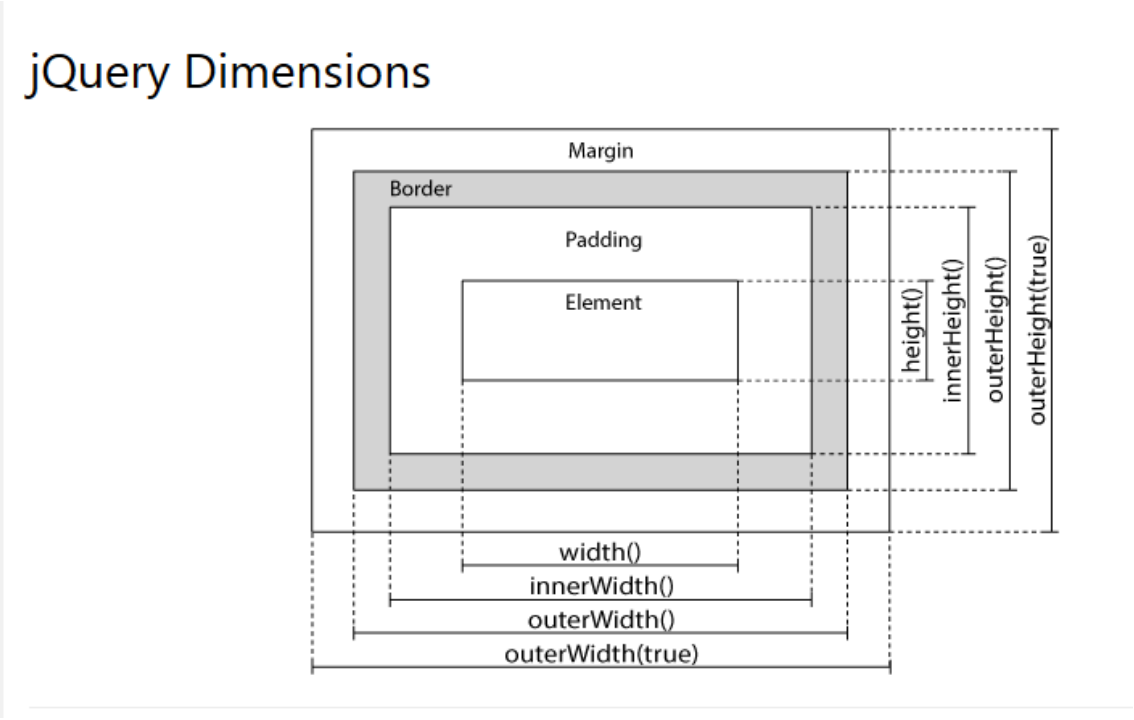
- **empty**()

  Vacía los elementos del objeto jQuery sobre el que se aplica, elimina toda su descendencia.

EJEMPLO 05 03 ejemplo insertar-borrar nodos.html

05.4 Posición y dimensiones de los elementos

Vamos a ver ahora distintos métodos para trabajar con la posición y dimensiones de los elementos del documento.



Métodos para trabajar con las dimensiones

- width()
- height()
- innerWidth()
- innerHeight()
- outerWidth()
- outerHeight()
- outerWidth(true)
- outerHeight(true)

- **height**(*[valor]*)

Obtiene, si no se aporta el argumento opcional *valor*, o establece, si sí aporta, la altura del elemento (sin padding), es decir, el valor de la propiedad CSS height. En modo get (obteniendo el valor) afecta al primer elemento del objeto jQuery, pero en modo set (estableciendo el valor) afecta a todos los elementos del objeto jQuery.

Es equivalente al método attr("height"[,valor]) cuando se aplica sobre un elemento que puede tener el atributo html height, por ejemplo una imagen.

Para elementos de tipo div no podemos utilizar attr("height "[,valor]), utilizaremos el método height ([valor]) o css("height"[,valor])

- **width** (*[valor]*)

Obtiene, si no se aporta el argumento opcional *valor*, o establece, si sí aporta, la anchura del elemento (sin padding), es decir, el valor de la propiedad CSS width. En modo get (obteniendo el valor) afecta al primer elemento del objeto jQuery, pero en modo set (estableciendo el valor) afecta a todos los elementos del objeto jQuery.

Es equivalente al método attr("width"[,valor]) cuando se aplica sobre un elemento que puede tener el atributo html width, por ejemplo con una imagen.
Para elementos de tipo div no podemos utilizar attr("width"[,valor]), utilizaremos width ([valor]) o css("width"[,valor])

Ejemplo: $("#div1").width(500).height(500);

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_dim_width_height
http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_dim_width_height_set

- **innerHeight()**

Obtiene la altura interior entre bordes, es decir, la distancia entre el interior del borde superior y el interior del borde inferior, del primer elemento del objeto jQuery. Es decir, la altura incluyendo el padding.

- **innerWidth()**

Obtiene la anchura interior entre bordes, es decir, la distancia entre el interior del borde izquierdo y el interior del borde derecho, del primer elemento del objeto jQuery. Es decir, la anchura incluyendo el padding.

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_dim_innerwidth_height

Observación: si intentamos pasar un valor a innerWidth o a innerHeight, lo que cambia es el width o el height en función del padding que tuviera

EJEMPLO 05 04 ejemplo dimensiones.html

- **outerHeight()**

Obtiene la altura incluyendo el padding y el borde, del primer elemento del objeto jQuery.

- **outerWidth()**

Obtiene la anchura incluyendo el padding y el borde, del primer elemento del objeto jQuery.

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_dim_outerwidth_height

- **outerHeight(true)**

Obtiene la altura incluyendo el padding, el borde y el margin, del primer elemento del objeto jQuery.

- **outerWidth(true)**

Obtiene la anchura incluyendo el padding, el borde y el margin, del primer elemento del objeto jQuery.

http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_html_outerwidthmargin_ref

Métodos offset() y position():

Ambos métodos **devuelven la posición de un elemento en la página**. Reciben un objeto jQuery y devuelven la localización del primer elemento que haya en ese objeto jQuery. La posición siempre se indica como valor de retorno del método **por medio de un objeto que tiene dos atributos, "top" y "left"**, indicando los píxeles que está separado de la esquina superior izquierda del documento. La diferencia entre estos dos métodos es que offset() indica la posición del elemento real, teniendo en cuenta los márgenes del elemento, lo que suele ser más útil. Por su parte, position() indica la posición respecto a su elemento padre.

- **position()**

Obtiene la posición del primer elemento del objeto jQuery respecto a su elemento contenedor.

Ejemplo:
```
x=$("p").position();
alert("Top: " + x.top + " Left: " + x.left);
```

https://www.w3schools.com/jquery/css_position.asp

- **offset**(*[coordenadas]*)

Obtiene o establece la posición del elemento respecto al documento. En modo get (obteniendo el valor) afecta al primer elemento del objeto jQuery, pero en modo set (estableciendo el valor) afecta a todos los elementos del objeto jQuery. Las coordenadas que devuelve y las que requiere como argumento son un objeto con las propiedades top y left. Por ejemplo, $('#titular').offset({top: 10, left: 50});.

https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_css_offset

```
$(selector).offset()
$(selector).offset({top:value,left:value})
```

```
posicion.left=10;
posicion.top=50;
$(selector).offset(posicion);
```

EJEMPLO 05 05 ejemplo posicionamiento.html

EJEMPLO 05 06 ejemplo movimiento.html