JavaScript

ANEXO 3.-

JQuery:

Selección y manipulación de elementos

DISTRIBUIDO POR:

CENTRO DE INFORMÁTICA PROFESIONAL S.L.

C/ URGELL, 100
08011 BARCELONA
TFNO: 93 426 50 87

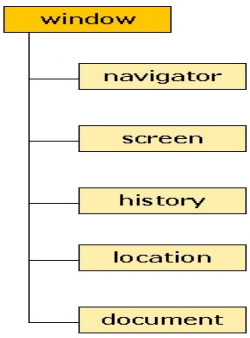
C/ RAFAELA YBARRA, 10
48014 BILBAO
TFNO: 94 448 31 33

www.cipsa.net

Modelo de Objetos del Navegador (BOM)

Un *modelo de objetos* define una familia jerárquica de objetos en los que uno (el objeto base o raíz) permite obtener el resto. En Javascript pueden diferenciarse dos modelos de objetos en función de su origen:

El *modelo de objetos de Navegador (BOM)* Contiene el conjunto de objetos relacionados con el navegador del usuario. El objeto raíz es ***window*** que permite obtener el resto de objetos: ***navigator***, ***screen***, ***history***, ***location*** y ***document***.



Esquema representativo del modelo de objetos dependientes del navegador

El objeto *window*

El objeto window representa la ventana del navegador. A partir de las propiedades de este objeto puede accederse a otros objetos como:

- ***navigator*** → Representa el navegador del usuario y permite obtener información sobre sus características.
- ***screen*** → Representa la pantalla del dispositivo del usuario. Permite obtener informacion sobre la resolución.
- ***history*** → Representa el histórico de direcciones por las que el usuario ha pasado hasta llegar a la página en la sesión actual. No confundir con el historial del navegador.
- ***location*** → Representa la barra de direcciones del navegador.
- ***document*** → Representa la página web. Este objeto permite acceder a los elementos HTML que conforman la maquetación de la propia página.

Ejemplo: El siguiente código obtiene y muestra el tamaño de la ventana del navegador, la resolución de la pantalla y los detalles del navegador empleado:

```
console.log("Ventana: " + window.innerWidth + " x " + window.innerHeight);
console.log("Pantalla " + window.screen.width + " x " + window.screen.height);
console.log("Navegador: " + window.navigator.appVersion );
```

```
Ventana: 913 x 987                                                    HtmlPage1.html:9
Pantalla 1920 x 1080                                                  HtmlPage1.html:10
Navegador: 5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)   HtmlPage1.html:11
Chrome/53.0.2785.116 Safari/537.36
```

Para examinar las propiedades de los objetos mencionados puede emplearse el método ***dir()*** del objeto ***console***:

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title></title>
    <meta charset="utf-8" />
    <script>
        // Detalle de objeto pantalla
        console.dir(window.screen);
    </script>
</head>
<body>
</body>
</html>
```

Algunas de las propiedades más importantes del objeto window son:

Property	Description
document	Returns the Document object for the window (See Document object)
history	Returns the History object for the window (See History object)
innerHeight	Returns the inner height of a window's content area
innerWidth	Returns the inner width of a window's content area
location	Returns the Location object for the window (See Location object)
name	Sets or returns the name of a window
navigator	Returns the Navigator object for the window (See Navigator object)
outerHeight	Returns the outer height of a window, including toolbars/scrollbars
outerWidth	Returns the outer width of a window, including toolbars/scrollbars
pageXOffset	Returns the pixels the current document has been scrolled (horizontally) from the upper left corner of the window
pageYOffset	Returns the pixels the current document has been scrolled (vertically) from the upper left corner of the window
parent	Returns the parent window of the current window
screen	Returns the Screen object for the window (See Screen object)
screenLeft	Returns the x coordinate of the window relative to the screen
screenTop	Returns the y coordinate of the window relative to the screen
screenX	Returns the x coordinate of the window relative to the screen
screenY	Returns the y coordinate of the window relative to the screen

Ref: http://www.w3schools.com/jsref/obj_window.asp

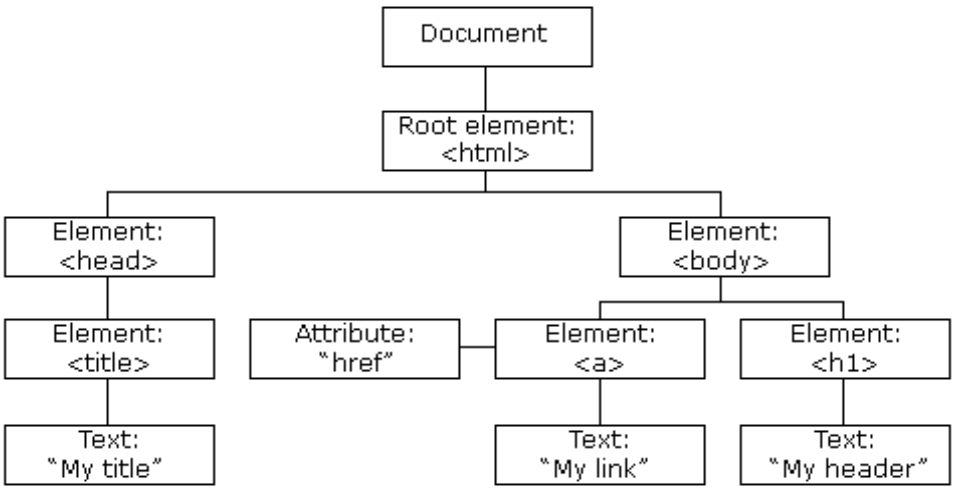
Modelo de Objetos de Documento (DOM)

Una de las tareas más habituales con *javascript* es la manipulación de los elementos que componen las páginas (capas, tablas, elementos de formularios, imágenes, enlaces..., etc).

Todas estas tareas se llevan a cabo empleando el *modelo de objetos de documento (DOM)*. DOM presenta el contenido de una página web como un árbol de objetos relacionados donde cada elemento HTML se corresponde con un objeto:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>New Web Project</title>
        <script src="codigo.js"></script>
    </head>
        <body>
        <h1>My Header</h1>
        <a href="page2.html">My link</a>
        </body>
</html>
```

El esquema de objetos correspondiente a la página es el siguiente:



Estructura de elementos de la página web.

La función de DOM es permitir a través de Javascript la obtención y manipulación de los elementos HTML. Por ejemplo:

- ✓ Modificar la imagen mostrada por un elemento ***<img>***
- ✓ Modificar el contenido de un elemento como un párrafo, o una capa añadiendo, modificando o eliminando otros elementos HTML.
- ✓ Modificar el estilo asignado, o asignar una clase de estilo diferente a un elemento para modificar su aspecto.
- ✓ Obtener el valor introducido en una caja de texto ***<input type='text'>***

Fundamentos de JQuery

¿Qué es jQuery?

jQuery es una librería programada a partir de Javascript que ayuda al desarrollo de páginas interactivas simplificando su programación y reduciendo la cantidad de código necesario. Las principales tareas que simplifica el uso jQuery frente a la programación en Javascript pueden resumirse en:

- Manipulación de elementos HTML / DOM
- Manipulación de CSS
- Control de eventos a elementos HTML
- Implementación de animaciones y efectos visuales diversos.
- Implementación de operaciones asíncronas AJAX.
- Utilidades complementarias

El uso de JQuery requiere no obstante conocimientos de HTML, CSS y del lenguaje Javascript.

Versiones de JQuery

Tradicionalmente han convivido dos versiones de JQuery:

- ***Versiones 1.X*** → Versión compatible con navegadores previos IE 8.0
- ***Versiones 2.X*** → Versión optimizada para navegadores IE 8.0 en adelante.

A partir de Julio de 2015 apareció JQuery 3.0 con lo que se cambio la nomenclatura:

- ***JQuery 3.0*** → Versión moderna compatible con navegadores IE 9.0 en adelante sucesora de JQuery 2.1.1.
- ***JQuery Compact 3.0*** → Versión retrocompatible con navegadores IE 8.0 sucesora de JQuery 1.11.1

Instalación de jQuery

La librería jQuery puede descargarse del sitio web oficial:

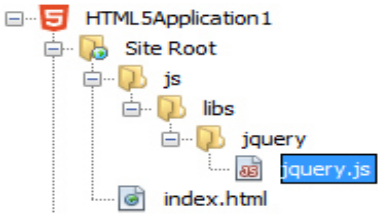
http://jquery.com/download/

Para cada una de las versiones existen dos distribuciones de jQuery:

- **Versión descomprimida *(uncompressed)*** → Versión para desarrollo y depuración. El fichero .js puede abrirse para ver el código fuente.
- **Versión comprimida *(minified)*** → Versión compacta más ligera y óptimizada para uso en sitios web en producción.

En cualquiera de los casos la librería puede descargarse como un único fichero con extensión ***.js***. Para su uso debe referenciarse en el atributo ***src*** de la etiqueta ***<script>***:

Ejemplo: Sea el siguiente sitio web:



Código: *index.html*

```
<!DOCTYPE html>
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="/js/libs/jquery/jquery.js"></script>
    </head>
    <body>
        <div>TODO write content</div>
    </body>
</html>
```

Instalación desde un *CDN (Content Delivery Network)*

Si no se desea descargar la librería en el propio sitio web puede hacerse referencia a un ***CDN*** (*Content Delivery Network*). Existen múltiples *CDNs* soportados por diferentes compañías; p.ej: ***MaxCDN***. (*http://code.jquery.com*)

Ejemplo: Referencia a librería jQuery 3.x para desarrollo. (*uncompressed*)

```
<script src="https://code.jquery.com/jquery-3.0.0.js"></script>
```

Selectores de JQuery

El uso más básico de JQuery consiste seleccionar uno o varios elementos HTML de la página y manipularlos obteniendo y modificando sus propiedades.

Para manipular uno o varios elementos HTML de la página primero debe obtenerse el/los objetos que lo representan. Por ejemplo; si se desea modificar la imagen mostrada por un elemento *<img>* primero debe obtenerse el objeto que la representa.

Los elementos que conforman una página no pueden ser obtenidos ni manipulados hasta que la página no se ha cargado correctamente. Para ello con JQuery se declara una función anónima asociadas al evento ***ready*** del ***document***:

```
$(document).ready(function () {
      // Código a ejecutarse al completarse la carga de la página
});
```

También puede emplearse la forma abreviada:

```
$(function () {
     // Código a ejecutarse al completarse la carga de la página
});
```

La selección de elementos mediante JQuery emplea por defecto ***$()*** que es una alias de la función ***jquery()***. Como argumento se indica una cadena con un selector que determina los elementos a seleccionar. JQuery emplea los selectores de CSS3 además de añadir algunos selectores propios:

Selector de ID

Esto selecciona aquel elemento cuyo atributo id = "*identificador*"

```
$("#identificador")
```

Ejemplo: La siguinte página modifica la imagen mostrada por el element <img> con id ="imagen" empleando JQuery:

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
      <meta charset="utf-8" />
    <script src="scripts/jquery-3.1.0.js"></script>
     <script>
         $(function () {
             $("#imagen").attr("src", "images/final.bmp");
         })
     </script>
</head>
<body>
    <img id="imagen" src="images/inicio.bmp" />
</body>
</html>
```

Todo ***selector*** devuelve una ***selección***, que representa a uno o varios elementos seleccionados. En este caso la selección es ún único elemento al que se modifica el valor de su atributo *src*.

Selector de estilo

Esto selecciona aquellos elementos que tengan la clase de estilo *".nombreClase"*.

```
$(".nombreClase")
```

Selector de etiqueta (tag)

Esto selecciona todos los elementos *<a>*

```
$("a")
```

Selector de atributos

Esto selecciona todos los enlaces *<a>* con el atributo *name = 'enlaces'.*

```
$("a[name='enlaces']")
```

En este tipo de selector existen las siguientes variaciones:

- `$("a[name|='pos']")`

Devuelve todos los elementos con el atributo “name” que empiece por “pos”.

- `$("a[name*='pos']")`

Devuelve todos los elementos con el atributo “name” que contenga “pos” delimitada con espacios delante y detrás.

- `$("a[name~='pos']")`

Devuelve todos los elementos con el atributo “name” que contenga exactamente el valor “pos” diferenciando mayúsculas/minúsculas:

- `$("a[name$='pos']")`

Ejemplo: La siguiente página modifica el valor de los párrafos con name=”valores” mediante JQUery:

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
       <meta charset="utf-8" />
    <script src="scripts/jquery-3.1.0.js"></script>
     <script>
         $(function () {
             $("P[name='valores']").text("0");
         })
     </script>
</head>
<body>
    <p name="valores">PARRAFO 1</p>
    <p name="valores">PARRAFO 2</p>
    <p>PARRAFO 3</p>
    <p>PARRAFO 4</p>
</body>
</html>
```

En este caso, la selección devuelta por el selector comprende más de un elemento. La modificación del contenido mediante el método ***text()*** se aplica a todos ellos.

Composición de selectores

El primer caso selecciona aquella capa * * con la clase de estilo “estilo”.

El segundo caso selecciona los elementos *<li>* contenidos en la lista *<ul>* con el estilo *“personas”* contenida en un elemento con el *id = “contenidos”.*

```
        $( "#contenido ul.personas li" );

        $("div.estilo");
```

PseudoSelectores

JQuery permite el uso de pseudo-selectores para permitir una selección de elementos más precisa más allá de los selectores tradicionales CSS. Lo habitual es emplear estos pseudoselectores en combinación con otro selector o mediante un filtro como el método ***$.filter()***. (se vé después).

Ejemplo: Los siguientes códigos devuelven todos los elementos <input type='radio'> presentes en los formularios de la página.

```
$("form :radio");

$("form").filter(":radio");
```

PseudoSelectores para elementos de formularios.

- :***password***
- :***reset***
- :***radio***
- :***text***
- :***submit***
- :***checkbox***
- :***button***
- :***image***
- :***hidden***
- :***file***
- :***input*** → Devuelve todos los elementos *<input>, <textarea>, <select>* y *<button>*

A estos se añaden:

- :***checked*** → Devuelve los elementos de tipo '*checkbox*' marcados.
- :***selected*** → Devuelve los elementos de un elemento *<select>* seleccionados.
- :***disabled*** → Devuelve elementos *<input>* con el atributo "*disabled*" presente.
- :***enabled*** → Devuelve los elementos sin el atributo "*disabled*".
- ***:focus*** --> Devuelve el elemento que tiene el foco en ese instante.

PseudoSelectores por contenido (*contains(), empty()*)

:contains() → Devuelve todos los elementos que contengan el texto especificado como argumento. La siguiente selección devuelve todas las capas * * que contengan "Jhon" en su contenido.

```
$("div:contains('John')")
```

:empty → Devuelve todos los elementos sin elementos anidados. La siguiente seleccion devuelve todos los elementos *<td>* sin contenido.

```
$("td:empty")
```

PseudoSelectores posicionales (*eq*(), *even*, *odd*, *lt*(), *gt*())

- ***:eq()*** → Devuelve el elemento indicado por el índice dado siendo el primero el 0.
- ***:even*** → Devuelve los elementos pares.
- ***:odd*** → Devuelven los elementos impares.
- ***:lt()*** → Devuelve todos los elementos por debajo del indicado.
- ***:gt()*** → Devuelve todos los elementos por encima del indicado.

Ejemplo: Sea el siguiente código

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title><meta charset="utf-8" />
    <script src="https://code.jquery.com/jquery-3.1.0.js"></script>
    <script>
        $().ready(function () {
            $("tr:even").css("background-color", "#ff0000");
            $("tr:odd").css("background-color", "#00ffff");
        });
    </script>
</head>
<body>
    <table border="1">
        <tr><td>Row with Index #0</td></tr>
        <tr><td>Row with Index #1</td></tr>
        <tr><td>Row with Index #2</td></tr>
        <tr><td>Row with Index #3</td></tr>
    </table>
</body>
</html>
```

El resultado pintan de rojo el fondo de las celdas impares y de azul las impares:

Row with Index #0
Row with Index #1
Row with Index #2
Row with Index #3

Si añadimos el siguiente código:

```
    <script>
        $().ready(function () {
            $("tr:even").css("background-color", "#ff0000");
            $("tr:odd").css("background-color", "#00ffff");
            $("tr:eq(1)").css("background-color", "#ffff00");
        });
    </script>
```

El resultado es que se pinta de amarillo el fondo de la celda nº2 (índice 1):

Row with Index #0
Row with Index #1
Row with Index #2
Row with Index #3

Los pseudoselectores ***:lt()*** y ***:gt()*** devuelven todos los elementos por encima o por debajo de un determinado valor dado su índice.

De igual modo si sustituimos el siguiente código:

```
    <script>
        $().ready(function () {
            $("tr:gt(1)").css("background-color", "#ff0000");
            $("tr:lt(1)").css("background-color", "#00ff00");
        });
    </script>
```

El resultado muestra de color rojo todas las siguientes a partir de las segunda "***:gt(1)***", y de verde todas las anteriores "***:lt(1)***".

(*) Los indices toman como base el valor 0 para el primer elemento. Si se indica un valor negativo, se comienza la cuenta a partir del último elemento.

Los pseudoselectores ***:first*** y ***:last*** devuelven el primer y último elemento de una selección:

```
    <script>
        $().ready(function () {
            $("tr:first").css("background-color", "#ff0000");
            $("tr:last").css("background-color", "#00ff00");
        });
    </script>
```

El resultado mostrado es el fondo de color rojo en el primer elemento fila, y el color verde de fondo en el último.

PseudoSelector condicional (*has()*)

El pseudoselector *:has()* devuelve aquellos elementos que contengan al menos un elemento que coincida con el selector indicado como argumento:

Ejemplo: El siguiente código muestra la selección de todas las capas *** *** que contengan un elemento ***<p>***:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
        <meta charset="utf-8" />
    <script src="https://code.jquery.com/jquery-3.1.0.js"></script>
    <script>
        $().ready(function () {
            $("div:has(p)").css("background-color", "#ff0000");
        });
```

```
    </script>
</head>
<body>
    <div><p>Capa 1</p></div>
    <div>Capa 2</div>
    <div><p>Capa 3</p></div>
</body>
</html>
```

El resultado mostrado es el siguiente:

Filtros

Los filtros son métodos que se llaman a partir de un selector para refinar el conjunto de elementos devuelto retornando únicamente aquellos que cumplen tanto con el selector como con el/los filtros indicados.

Existen filtros que son versiones alternativas de los pseudoselectores que funcionan más rápido. Es el caso de los pseudoselectores con argumentos tales como ***:eq(), :lg(), :hg(), :has()***, los cuales también pueden invocarse también como filtros.

Ejemplo: Selecciones empleando el pseudoselector ***:eq()***, y el método de filtro equivalente ***$().eq():***

```
$("div").eq(2).css("background-color", "#ff0000");  // Más rápido

$("div:eq(2)").css("background-color", "#ff0000");
```

El método ***$().filter()*** representa un filtro que retorna aquellos elementos que cumplen con el selector indicado como argumento:

Ejemplo: El siguiente código devuelve aquellos elementos *<li>* con la clase de estilo "*current*" contenidos en un elemento *<ul>*:

```
$("ul li").filter(".current");
```

También puede invocarse al método $().filter() indicando como argumento pseudoselectores como ***:even***, ***:odd***..., etc.

```
$("div").filter(":even").css("background-color", "#ff0000");  // Más eficiente.

$("div:even").css("background-color", "#ff0000");
```

También es posible emplear el método ***$().filter()*** indicando como argumento una función que debe devolver un valor lógico cierto o falso indicando que elementos pasan el filtro y cuales no. Cada elemento devuelto por el primer selector es pasado a la función mediante los parámetros *index* y *element*.

Ejemplo: Los siguientes códigos seleccionan aquellas capas *** *** que contenga como valor el texto "*Capa 1*":

```
// Con pseudoselector
$("div:contains('Capa1')").css("background-color", "#ff0000");

// Pseudoselector a través de método $().filter()
$("div").filter(":contains('Capa1')").css("background-color", "#ff0000");

// Método $().filter() con función anidada.
$("div").filter(function (index, element) {
    return element.innerHTML == "Capa1";
}).css("background-color", "#ff0000");
```

Filtro de comprobacion *$().is()*

Este filtro devuelve un valor lógico (cierto/falso) indicando si al menos uno de los elementos de la selección cumple el selector indicado como argumento.

Ejemplo: El siguiente código devuelve un valor lógico indicando si la casilla de verificación *<input type="checkbox">* con *id="estado"* está marcada o no:

```
var estado = $("#estado").is(":checked");
```

Recomendación sobre el uso de Selectores

Para optimiza el rendimiento de los scripts se recomienda emplear selectores cortos aunque no sean necesariamente los más precisos.

Ejemplo: Ambos selectores obtienen todos los elementos ***<th>*** con el estilo "*especial*" dentro de la tabla con el *id="datos"*. Sin embargo, el segundo es más óptimo.

```
$("#datos thead tr th.special")
$("#datos th.special")                    // Mas óptimo.
```

Para más información sobre los selectores y pseudoselectores consultar la web oficial de JQuery: ***http://api.jquery.com/category/selectors/***

Tratamiento de las selecciones

Guardado de selecccion

Los selectores devuelven una selección de elementos. Estos pueden almacenarse en una variable para aplicar posteriormente diferentes operaciones sobre sus elementos.

La asignación a una variable se hace mediante una asignación indicando el selector en el lado derecho. Es habitual anteponer el signo $ delante de la variable para indicar que hace referencia a una selección:

```
var capas = $("div");
var $capas = $("div");   // Indica que “capas” contiene una selección de JQuery
```

Comprobacion de selección

Para saber si el selector ha devuelto algún elemento o no puede emplearse la propiedad ***$().length*** que devuelve el nº de elementos contenidos en la selección:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <script src="https://code.jquery.com/jquery-3.1.0.js"></script>
    <script>
        $(function () {
            var capas = $("div");
            alert("El nº de capas presentes es : " + capas.length);
        });
    </script>
</head>
<body>
    <div>Capa1</div>
    <div>Capa2</div>
    <div>Capa3</div>
</body>
</html>
```

Cada selección es única.

Cada selector devuelve un objeto ***JQuery*** que representa la selección resultante. Aunque dos selectores devuelven el mismo conjunto de elementos, los objetos selección devueltos son siempre distintos.

Por tanto; si se comparan directamente, el resultado siempre será falto incluso si tienen el mismo conjunto de elementos:

```
var logo1 = $("div");
var logo2 = $("div");
```

```
// Comparacion de selecciones = FALSE
alert(logo1 === logo2);
// Comparacion del primer elemento de la selecciones = TRUE
alert(logo1.get(0) === logo2.get(0));
```

La primera sentencia ***alert()*** devuelve el resultado "falso", puesto que son objetos selección distintos a pesar de representar el mismo conjunto de elementos. La segunda sentencia ***alert()*** devuelve el resultado "true", ya que compara el primer elemento contenido en ambas selecciones y es el mismo.

Encadenamiento de selecciones (*chaining*)

Los métodos de filtro vistos anteriormente se invocan a partir de una selección y devuelven a su vez otra selección. Esto permite invocar múltipes filtros uno tras otro para restringuir más la selección. Esta propiedad se denomina encadenamiento.

Ejemplo: La siguiente selección asigna un contenido al el tercer elemento *<h3>* contenido dentro de la capa con *id = "content"*:

```
$("#content")
    .find("h3")
    .eq(2)
    .html("nuevo contenido");
```

También puede emplearse el encadenamiento para seleccionar diferentes elementos sucesivamente en vez de refinar una única selección. Para ello debe emplearse el método ***$().end()***:

```
$("#content")                       // Seleccion <div id='content'>
    .find("h3")                     // Seleccion todos los <h3> contenidos.
    .eq(2)                          // Seleccion 3er <h3>
        .html("Nuevo contenido")        // Asignacion contenido 3er <h3>
        .end()                      // Retorno a seleccion de todos los <h3>
    .eq(0)                          // Seleccion 1er <h3>
        .html("Nuevo contenido");       // Asignacion contenido 1er <h3>
```

Manipulación de elementos

Métodos obtenedores / Establecedores

JQuery define una serie de métodos que permiten obtener y establecer el valor de ciertas propiedades de elementos seleccionados en una selección. Estos métodos pueden emplearse de dos modos:

- ***Obtención*** → Devuelve el valor indicado del primer elemento de la selección si no es el único:

```
// Obtencion de contenido
var contenido = $("h1").html();
```

- ***Asignación*** → Ajusta el valor indicado como argumento a todos los elementos de la selección. El método retorna además la propia selección permitiendo encadenar más métodos en la misma línea:

```
// Modificacion de contenido
$("h1").html("bacalao");
```

Manipulación de estilos (*$().css()*)

Este método permite obtener o establecer el valor de una regla de estilo de un elemento.

Obtención: Devuelve el valor del estilo del primer elemento de la selección.

```
$('h1').css('font-size'); // Devuelve el tamaño de la fuente establecido, p.ej: “24px”
```

Asignación: Modifica el estilo en todos los elementos de la selección.

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

Ejemplo: El siguiente código muestra una página web provista de una caja de texto y dos botones. El botón “*ver*” muestra por pantalla el valor del estilo “*background-color*” asociado a la caja de texto. El botón “*cambiar*” modifica el valor del mismo.

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <script src="Scripts/jquery-3.0.0.js"></script>
    <script>
        $(function () {
            // Modifica los estilos ‘background-color” de la caja de texto
            $("#cambiar").click(function () {
                $("#texto").css("background-color","red");
            });
            // Muestra el valor del estilo “background-color”
            $("#ver").click(function () {
                alert($("#texto").css("background-color"));
        });
    });
    </script>
```

```
</head>
<body>
    <form id="datos">
        <input type="text" id="texto" />
        <input type="button" id="ver" value="ver" />
        <input type="button" id="cambiar" value="cambiar" />
    </form>
</body>
</html>
```

También es posible asignar varias propiedades al mismo tiempo indicando como argumento un objeto on los estilos y valores deseados en JSON:

Ejemplo: El siguiente código aplicado en el script anterior modifica al mismo tiempo las propiedades de estilo "*background-color*" y "*color*" de la caja de texto:

```
            $("#cambiar").click(function (event) {
                $("#texto").css(
                    {"background-color": "red",
                    "color":"white"}
                );
            });
```

En el caso de propiedades de estilo con valores numéricos es posible establecer valores relativos incrementando o reduciendo el actual con los operadores += y -=

```
        $('h1').css({
            'fontSize': '+=15px',     // suma 15px al tamaño original del elemento
            'paddingTop': '-=20px'   // resta 20px al padding superior original del elemento
        });
```

Manipulación de clases (*$().addClass(), $().removeClass(), $().toggleClass()*)

Una de las operaciones más comunes contra los elementos de una página es la modificación de la hoja de estilo. Esto puede hacerse empleando el método ***$().prop()*** indicando el nombre de la propiedad "*class*" y el nombre de la clase de estilo:

Ejemplo: La siguiente sentencia aplica la clase de estilo "*error*" al elemento con el atributo id = "texto":

```
        $("#texto").prop("class", "error");
```

JQuery no obstante, define otros métodos adicionales para añadir y eliminar hojas de estilo a uno o varios elementos seleccionados:

- ***$().addClass()*** → Añade la hoja de estilo indicada como argumento a los elementos seleccionados. Esto implica que las propiedades de estilo definidas en la hoja de estilos se añaden a las ya definidas en el control por otras hojas de estilo.

- ***$().removeClass()*** → Elimina de los elementos seleccionados las propiedades de estilo definidas en la hoja de estilo indicada como argumento.

- ***$().toggleClass()*** → Este método sustituye una hoja de estilos. Si se invoca indicando una hoja de estilo como único argumento, la elimina si ya existe y la añade si no.

Ejemplo 1: El siguiente código muestra un formulario provisto de una caja de texto y dos botones "*OK*" y "*ERROR*". La caja de texto lleva asignado por defecto la hoja de estilo "*normal*". Cuando se pulsa el botón "ERROR" se le añade la hoja de estilo "error". Al pulsar el botón "OK" se elimina para recuperar el estilo original del control:

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title></title>
     <style>
         .normal {
             background-color:yellow;
         }
         .error {
             background-color:red;
             color: white;
         }
     </style>
    <script src="Scripts/jquery-3.0.0.js"></script>
    <script>
         $().ready(function () {
             $("#error").click(function () {
                 $("#texto").addClass("error");        // Añade el estilo "error"
             });
             $("#ok").click(function () {
                 $("#texto").removeClass("error");     // Elimina el estilo "error"
             });
         });
    </script>
</head>
<body>
    <form id="datos">
        <input type="text" id="texto" class="normal"/>
        <input type="button" id="ok" value="OK" />
        <input type="button" id="error" value="ERROR" />
    </form>
</body>
</html>
```

Ejemplo 2: La siguiente página web muestra una caja de texto y un botón "*Cambiar*" que añade y elimina alternativamente a la caja de texto la hoja de estilo "*especial*" cada vez que es pulsado:

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title></title>
    <script src="Scripts/jquery-3.0.0.js"></script>
     <style>
         .normal {
             background-color:yellow;
         }
         .especial {
             background-color:red;
             color: white;
         }
     </style>
    <script>
         $().ready(function () {
             $("#toogle").click(function () {
                 $("#texto").toggleClass("especial");
             });
         });
    </script>
</head>
```

```
<body>
    <form id="datos">
        <input type="text" id="texto" class="normal"/>
        <input type="button" id="toogle" value="Cambiar" />
    </form>
</body>
</html>
```

Si se invoca el método añadiendo una variable lógica como segundo argumento, el método añade o elimina el estilo en función de su valor. De este modo, si la variable tiene por valor "*cierto*" lo añade, y si es "*falso*" lo elimina:

Ejemplo 3: La siguiente página muestra una caja de texto acompañada de una casilla de verificación y un botón "*Cambiar*". Al pulsar este botón, si la casilla de verificación está marcada se añade a la caja de texto el estilo "*especial*". En caso contrario se le retira:

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title></title>
    <script src="Scripts/jquery-3.0.0.js"></script>
    <meta charset="utf-8" />
    <style>
        .normal {
            background-color:yellow;
        }
        .especial {
            background-color:red;
            color: white;
        }
    </style>

    <script>
        $().ready(function () {
            $("#toogle").click(function () {
                // Obtencion del estado de marcado de la casilla de verificacion id='estado'
                var estado = $("#estado").is(":checked");
                // Añade el estilo "especial" en función del valor de la variable estado.
                $("#texto").toggleClass("especial", estado);
            });
        });
    </script>
</head>
<body>
    <form id="datos">
        <input type="text" id="texto" class="normal"/>
        <input type="checkbox" id="estado" />
        <input type="button" id="toogle" value="Cambiar" />
    </form>
</body>
</html>
```

Manipulación de atributos y propiedades (*$().attr() / $().prop()*)

Los métodos *$().attr()* y *$().prop()* se emplean para obtener y modificar los atributos/propiedades de los objetos de una página.

- *Los **atributos** son considerados las características de una etiqueta HTML* que define un determinado elemento en una página web. Cuando el navegador carga una página, cada uno de esos elementos HTML se convierten en objetos formado *modelo de objetos del documento* o *DOM*.

- *Cada objeto DOM consta de una serie de **propiedades*** (*cuyos nombres coinciden con los de los atributos*) que reflejan los *valores actuales* de los atributos del elemento.

Ejemplo: Sea el siguiente campo de texto de un formulario:

```
<input type="text" id="texto" value="VIEJO" />
```

Si ejecutamos el siguiente código obtendremos los siguientes resultados:

```
$("#texto").val("NUEVO");           // El control pasa a mostrar 'NUEVO'
alert($("#texto").attr("value"));   // Muestra valor original del atributo= 'VIEJO'
alert($("#texto").prop("value"));   // Muestra valor actual de la propiedad= 'NUEVO'
```

(*) Para obtener y modificar el valor de ciertas propiedades tales como "***checked***", "***selected***" o "***disabled***" se recomienda el empleo del método ***prop()***, en lugar de ***attr()***.

Para obtener el valor de un atributo/propiedad de un elemento debe invocarse el método indicando su nombre. Para modificar el valor de un atributo/propiedad debe invocarse indicando su nombre y el nuevo valor.

Ejemplos: La siguiente página web muestra una imagen cuyo valor del atributo "*src*" es mostrado al pulsarse el botón "*ver*" y es modificado al pulsar el botón "*cambiar*":

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <script src="Scripts/jquery-3.0.0.js"></script>
    <script>
        $().ready(function () {
            // Altera valor del atributo "src" del elemento <img>
            $("#cambiar").click(function (event) {
                $("img").attr("src", "images/imagen1.jpg");
            });
            // Muestra valor del atributo "src" del element <img>
            $("#ver").click(function (event) {
                alert($("img").attr("src"));
            });
        });
    </script>
</head>
<body>
    <form id="datos">
        <img src="images/imagen0.png" alt="texto" />
        <input type="button" id="ver" value="ver"/>
        <input type="button" id="cambiar" value="cambiar" />
    </form>
</body>
</html>
```

Es posible modificar varios atributos/propiedades de un mismo elemento indicando como argumento una matriz de pares "***<atributo/propiedad>:<valor>***":

```
<script>
    $().ready(function () {
        // Altera valor del atributo "src" del elemento <img>
        $("#cambiar").click(function (event) {
            $("img").prop({
                src: "images/imagen1.jpg",
                alt: "imagen"
            });
        });
        // Muestra valor del atributo "src" del element <img>
        $("#ver").click(function (event) {
            alert($("img").prop("src"));
            alert($("img").prop("alt"));
        });
    });
</script>
```

Manipulación de Elementos

JQuery ofrece muchos métodos para modificar tanto cada elemento como los elementos contenidos en él. Entre las operaciones más comunes está la obtención y modificación del contenido y/o valores contenidos en un elemento. Para estas operaciones comunes JQuery dispone de métodos específicos:

- ***$().html()*** → Obtiene/Establece el contenido HTML de un elemento.
- ***$().text()*** → Obtiene/Establece el conteindo de un elemento (sin código HTML).
- ***$().width()*** → Obtiene/Establece el ancho de un elemento.
- ***$().height()*** → Obtiene/Establece el alto de un elemento.
- ***$().val()*** → Obtiene/Establece el valor de elementos de formulario (*value*).

$().html()

Este método permite obtener y modificar el contenido de un elemento.

Ejemplo: La sentencia anterior añade el párrafo indicado *a todas las capas * del documento:

```
$("div").html("<p>HOLA</p>");
```

Si se desea modificar una capa concreta deberá emplearse un selector más preciso o emplear pseudoselectores como ***:first***, ***:last***, o ***:eq()*** para obtener un determinado elemento de la selección.

```
// Modifica la primera capa seleccionada
$("div:first").html("<p>CAPAS</p>");
$("div").filter(":first").html("<p>CAPAD</P>");

// Modifica la última capa seleccionada
$("div:last").html("<p>CAPAS</p>");
$("div").filter(":last").html("<p>CAPAD</P>");

// Modifica la segunda capa seleccionada
$("div:eq(1)").html("<p>CAPAS</p>");
$("div").eq(1).html("<p>CAPAS</p>");
```

El método *$().html()* también permite obtener el contenido HTML de un elemento seleccionado. Si la selección contiene más de un elemento, *sólo se retorna el contenido del primero.*

```
// Muestra el contenido HTML de la primera capa seleccionada
alert($("div:first").html());
```

$().val()

Este método obtiene/modifica los valores introducidos o seleccionados en elementos de formularios tales como cajas de texto, áreas de texto, listas de selección, botones de opción, casillas de verificación… etc.

- Este método se emplea principalmente para obtener el valor de cajas de **texto *<input type="text">*** y elementos ***<textarea>***.
- Si se emplea con listas ***<select>*** devuelve el valor del elemento *<option>* seleccionado. En caso de listas múltiples, el valor devuelto es una matriz con los valores de todas las opciones seleccionadas.
- Para obtener el valor de la casilla de verificación ***<input type="checkbox">*** o el botones de opción ***<input type="radio">*** seleccionado puede emplearse el método *val()* combinado en selectores con el pseudoselector "*:checked*".

Ejemplo: La siguiente página web consta de un formulario y un script que se activa al pulsarse el botón "*COMPROBAR*" que muestra los valores de los elemento seleccionados.

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <script src="Scripts/jquery-3.0.0.js"></script>
    <script>
        $().ready(function () {
            $(":button").click(function (event) {
                // Devuelve el valor de la opcion seleccionada
                alert($("#lista_mono").val());
                // Devuelve matriz con valores de opciones seleccionadas
                alert($("#lista_multiple").val());
                // Devuelve el valor de la casilla de verificacion seleccionada
                alert($("input:checkbox:checked").val());
                // Devuelve el valor seleccionado de un conjunto de botones de opcion.
                alert($("input:radio[name=r]:checked").val());
            });
        });
    </script>
</head>
<body>
    <form id="datos">
        <select id="lista_mono">
            <option>Single</option>
            <option>Single2</option>
        </select>
        <select id="lista_multiple" multiple="multiple">
            <option selected="selected">Multiple</option>
            <option>Multiple2</option>
            <option selected="selected">Multiple3</option>
        </select>
        <input type="checkbox" name="checkboxname" value="check1"> check1
        <input type="checkbox" name="checkboxname" value="check2"> check2
        <input type="checkbox" name="checkboxname" value="check3"> check3
        <input type="checkbox" name="checkboxname" value="check4"> check4
        <input type="radio" name="r" value="radio1"> radio1
        <input type="radio" name="r" value="radio2"> radio2
        <br />
        <input type="button" value="COMPROBAR"/>
    </form>
</body>
</html>
```

El método ***val()*** también puede emplearse para modificar el valor inscrito o seleccionado en los distintos elementos de un formulario.

- Para insertar un valor en un control caja de texto *<input type="text">* o *<textarea>,* se invoca el método *val()* indicando como argumento el texto a insertar.

- Para seleccionar elementos de una lista *<option>*, casillas de verificación *<input type="checkbox">*, o botones de opción *<input type="radio">*, puede invocarse el método pasando como argumento una matriz con los valores de las opciones a seleccionar y casillas.

Ejemplo: El siguiente código muestra un formulario provisto de unas listas y un conjunto de botones de opción y casillas de verificación que son seleccionados desde script al pulsarse el botón "SELECCIONAR".

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <script src="Scripts/jquery-3.0.0.js"></script>
        <meta charset="utf-8" />
    <script>
         $().ready(function () {
             $(":button").click(function (event) {
                 // Selección de opción en lista simple
                 $("#lista_mono").val("Single2");
                 // Selección de opciones en lista múltiple
                 $("#lista_multiple").val(["Multiple2", "Multiple3"]);
                 // Marcación de casillas de verificación
                 $("input[name=c]").val(["check1", "check2"]);
                 // Marcación de botón de opción
                 $("input[name=r]").val(["radio1"]);
             });
});
    </script>
</head>
<body>
    <form id="datos">
        <select id="lista_mono">
            <option>Single</option>
            <option>Single2</option>
        </select>
        <select id="lista_multiple" multiple="multiple">
            <option selected="selected">Multiple</option>
            <option>Multiple2</option>
            <option selected="selected">Multiple3</option>
        </select>
        <input type="checkbox" name="c" value="check1"> check1
        <input type="checkbox" name="c" value="check2"> check2
        <input type="checkbox" name="c" value="check3"> check3
        <input type="checkbox" name="c" value="check4"> check4
        <input type="radio" name="r" value="radio1"> radio1
        <input type="radio" name="r" value="radio2"> radio2
        <br />
        <input type="button" value="SELECCIONAR"/>
    </form>
</body>
</html>
```

$().text()

Este método permite obtener y modificar el contenido de los elementos de una página web eliminando las etiquetas HTML. Este método se comporta de diferente manera que el resto al invocarlo en selecciones de múltiples elementos:

- Empleado para asignar un contenido; lo inserta en todos los elementos de la selección.
- Empleado para obtener un contenido, devuelve el contenido combinado de todos los elementos de la selección eliminando las etiquetas HTML.

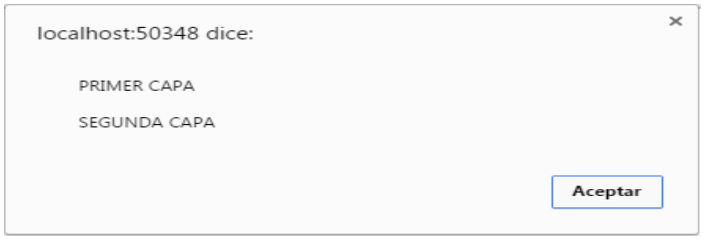
Ejemplo: La siguiente página web muestra dos párrafos cuyo contenido es sustituido por el valor "0" al ser pulsado el botón "*RESET*":

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title></title>
    <script src="Scripts/jquery-3.0.0.js"></script>
    <meta charset="utf-8" />
     <script>
         $().ready(function () {
             $("#reset").click(function () {
                 $("p").text("0");    // Añade el valor "0" como contenido a todos los párrafos
             });
         });
     </script>
</head>
<body>
    <div><p>PRIMER CAPA</p></div>
    <div><p>SEGUNDA CAPA</p></div>
    <input type="button" value="RESET" id="reset" />
</body>
</html>
```

Ejemplo: El siguiente código obtiene y muestra en un ***alert()*** el texto combinado de las dos capas presentes en la página:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <script src="Scripts/jquery-3.0.0.js"></script>
    <meta charset="utf-8" />
     <script>
         $().ready(function () {
             alert($("div").text());
         });
     </script>
</head>
<body>
    <div><p>PRIMER CAPA</p></div>
    <div><p>SEGUNDA CAPA</p></div>
</body>
</html>
```

El resultado mostrado por pantalla es:



Funciones de retrollamada (*callbacks*)

Las funciones ***$().text()***, ***$().html()*** y ***$().val()*** también admiten como argumento una función. Ésta se ejecuta por cada elemento de la selección y tiene como parámetros el índice del elemento en la selección (siendo 0 el primero), y el valor actual del mismo. El valor retornado por la función es el que se asigna como nuevo.

Ejemplo: La siguiente página muestra en una capa un encabezado *<h1>* con el valor 0, el cual se actualiza con la hora actual cada vez que se pulsa el *botón id="botón"* mediante la función *$().html()* y una función de retrollamada.

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title></title>
    <script src="Scripts/jquery-3.0.0.js"></script>
    <meta charset="utf-8" />
     <script>
        $().ready(function () {
            // Cuando se pulsa el botón
            $("#boton").click(function () {
                // Se modifica el contenido de la capa
                $("#contador").html(function (i, v) {
                    // El parámetro v tiene el valor actual de $().html()
                    alert("Valor anterior: " + v);
                    // El nuevo valor para $().html() es el retornado por la función
                    var d = new Date();
                    return "<h1>"+d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds() + "</h1>";
                });
            });
        });
     </script>
</head>
<body>
    <div id="contador">
        <h1>0</h1>
    </div>
    <input type="button" id="boton" value="actualizar" />
</body>
</html>
```

(*) La ventaja del uso de funciones de retrollamada con estos métodos en vez de indicando directamente el valor a asignar es que permite obtener al mismo tiempo el valor actual y preparar el valor nuevo en base a él si es necesario.

Recorrido de una selección

Los métodos de modificación vistos modifican por igual todos los elementos contenidos en la selección contra la que se invocan. Sin embargo, hay ocasiones en la que los elementos seleccionados deben modificarse de manera distinta en función del valor que contienen o de alguna de sus propiedades. En estos casos, la modificación debe hacerse elemento a elemento recorriendo la selección para ello. Para ello se emplea el método ***$().each().***

El método ***$().each()*** *recorre los elementos de la selección contra la que se invoca* y *ejecuta para cada uno la función de retrollamada indicada como argumento*. Dentro de la funcion de retrollamada el elemento es accesible mediante la referencia ***this***:

```
$("seleccion").each(function (index) {
        // Código de obtención y manipulación para cada elemento
});
```

Ejemplo 1: La siguiente página muestra un formulario con tres cajas de texto a los que se asigna un fondo rojo si el valor indicado NO es par al pulsar el botón de envío:

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title></title>
    <script src="Scripts/jquery-3.0.0.js"></script>
    <meta charset="utf-8" />
     <script>
         $().ready(function () {
             // Cuando se pulsa el botrón de envío del formulario
             $("#formulario").submit(function (event) {
                 // Por cada uno de las cajas de texto con name="datos"
                 $("form input[name='datos']").each(function (index) {
                     // Obtencion del valor de cada elemento.
                     var valor = $(this).val();
                     if (valor % 2 != 0) {
                         // Valor no par. Se tiñe de rojo el fondo del elemento
                         event.preventDefault(); // Cancela envio formulario.
                         $(this).css("background-color", "#ff0000");
                     } else {
                         // Valor par. Se tiñe de blanco el fondo del elemento.
                         $(this).css("background-color", "#ffffff");
                     }
                 });
             })
         });
     </script>
</head>
<body>
    <form id="formulario">
        <input type="number" name="datos" id="uno" />
        <input type="number" name="datos" id="dos" />
        <input type="number" name="datos" id="tres" />
        <input type="submit" value="Comprobar" />
    </form>
</body>
</html>
```

Ejemplo 2: Supóngase que deseamos añadir la clase de estilo "*foo*" a todos los elementos *<li>* de la página web uno por uno recorriendo la selección que los contiene:

```
$("li").each(function () {
    $(this).addClass("foo");
});
```

(*) El elemento referenciado en cada ejecucion se obtiene mediante *this*. La función *$().addClass()* se invoca a partir de una selección, así que se crea una selección con el elemento *this* mediante *$(this)*.

Si la modificación es la misma para todos los elementos puede invocarse directamente la función ***$().addClass()*** sobre la selección tal cual y el resultado es el mismo.

```
$("li").addClass("foo");
```

Índice de elementos.

JQuery permite obtener la posición de un elemento en una selección. Esta posición se obtiene mediante la función ***$().index()*** como un valor numérico entero siendo 0 el primero. Este método puede emplearse de dos modos:

Si la función se llama sin indicar ningún argumento, devuelve la posición del elemento contra el que se invoca en relación a sus elementos hermanos en el documento web.

Ejemplo: La siguiente página presenta una lista de valores *<ul>*, donde cada elemento *<li>* ejecuta una función de Javascript al ser pulsado que muestra la posición del elemento seleccionado respecto al resto de los elementos de la lista.

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title></title>
    <script src="Scripts/jquery-3.0.0.js"></script>
    <meta charset="utf-8" />
    <script>
        $().ready(function () {
            // Por cada elemento <li> al ser pulsado...
            $("li").click(function () {
                // Se muestra la posicion relativa del elemento seleccionado $(this)
                alert("Elemento: " + $(this).index());
            });
        });
    </script>
</head>
<body>
    <ul>
        <li>A</li>
        <li>B</li>
        <li>C</li>
    </ul>
</body>
</html>
```

Si la función se llama indicando un elemento como argumento devuelve la posición del mismo en relacion a los elementos de la selección contra la que se invoca. En tal caso, si el elemento indicado como argumento no estuviese presente en la selección, la función retorna como posición el valor -1.

Ejemplo: El siguiente código es equivalente al de la página anterior, pero con la función *$().index()* invocada contra la selección de todos los elementos *<li>* indicando como argumento el propio elemento seleccionado mediante *$(this)*:

```
<script>
    $().ready(function () {
        // Por cada elemento <li> al ser pulsado...
        $("li").click(function () {
            // Se muestra la posicion del elemento seleccionado
            // en relacion a los elementos <li>
            alert("Elemento: " + $("#lista li").index($(this)));
        });
    });
</script>
```

Obtención de datos vinculados *$().data()*

En HTML5 se define el atributo data-* que puede ser añadido a cualquier elemento de la página para vincular una información al mismo:

```
<!-- valor clave foo -->
<a id='foo' data-foo='valor' href='#'>Foo</a>
<!-- valor clave foo-bar -->
<a id='foobar' data-foo-bar='valor' href='#'>Foo Bar</a>
```

Estos valores pueden obtenerse mediante JQuery empleando la función ***$().data()*** e indicando como argumento la clave del atributo:

```
// Obtención el valor del atributo “data-foo”
console.log($('#foo').data('foo'));

// Obtención el valor del atributo “data-foo-bar”
console.log($('#foobar').data('fooBar'));
```

(*) En el caso de atributos *data* con clave compuesta (*“data-foo-bar”*), la funcion ***$().data()*** de JQuery debe indicar la clave como: “*fooBar*”.

Estos atributos se emplean para almacenar datos en los elementos HTML que luego pueden obtenerse para diferentes fines desde JQuery.

Ejemplo 1: El siguiente código muestra la obtención del valor "pk" asociado al elemento seleccionado de una lista desde JQuery:

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <meta charset="utf-8" />
    <script src="Scripts/jquery-3.1.0.js"></script>
    <script>
        $(function () {
            $("#lista").change(function () {
                // Obtencion del valor del atributo value de opcion seleccionada
                console.log("valor: " + $(this).val());
                // Obtencion del texto de la opcion seleccionada
                console.log("elemento: " + $("#lista option:selected").html());
                // Obtencion del valor del atributo 'data-
                console.log("pk: " + $("#lista option:selected").data("pk"));
            })
        });
    </script>
</head>
<body>
    <form>
        <select id="lista">
            <option value="001" data-pk="pk001">petrov</option>
            <option value="002" data-pk="pk002">roger</option>
            <option value="003" data-pk="pk003">victor</option>
            <option value="004" data-pk="pk004">ivan</option>
            <option value="005" data-pk="pk005">yuri</option>
        </select>
    </form>
</body>
</html>
```

Ejemplo 2: El siguiente código muestra una página provista de una imagen y dos botones "*ROJO*" y "*VERDE*", tal que al ser pulsados cada uno de ellos se muestra una imagen con un faro de color verde o rojo respectivamente.

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <meta charset="utf-8" />
    <script src="Scripts/jquery-3.1.0.js"></script>
    <script>
        $(function () {
            $("button").on("click", function () {
                // Obtencion del valor del atributo "data-color" del boton pulsado
                var color = $(this).data("color");
                // Asignacion de archivo de imagen correspondiente
                $("#imagen").attr("src", "images/" + color + ".png");
            });
        });
    </script>
</head>
<body>
    <img id="imagen" />
    <button data-color="green">VERDE</button>
    <button data-color="red">ROJO</button>
</body>
</html>
```

Funciones de JQuery

Las funciones de JQuery pueden dividirse en dos tipos según el modo de llamarlas:

- *Funciones de selección* (***$().funcion()***) → Son aquellas que se invocan a partir de una selección y se ejecutan para cada uno de los elementos presentes en la selección.

  (*) En la documentación de la API de JQuery estas funciones se muestran como "***.funcion()***".

- *Funciones de utilidad* (***$.funcion()***) → Son aquellas que se invocan contra el operador ***$*** directamente y reciben los elementos sobre los que actuar como argumentos.

  (*) En la documentación de la API de JQuery estas funciones se muestran como "***$.funcion()***".

$.isNumeric() → Devuelve un valor lógico cierto/falso indicando si el valor indicado como argumento contiene un valor numérico o no.

Ejemplo: El siguiente código muestra una página con un formulario y un campo de entrada de tipo texto. Se comprueba mediante JQuery que el valor introducido sea un valor numérico válido al pulsarse el botón de envío.

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title></title>
    <script src="Scripts/jquery-3.0.0.js"></script>
     <script>
         $().ready(function () {
             // Evento de envio de datos de formulario
             $("#formulario").submit(function (event) {
                 // Obtencion del valor inscrito en caja de texto
                 var valor = $("form input:text").val();
                 // Comprobación: Es un valor numérico?
                 if (!$.isNumeric(valor)) {
                     // NO LO ES --> Error
                     alert("Valor no válido!")
                     event.preventDefault();
                 }
             });
         });
     </script>
</head>
<body>
    <form id="formulario">
        <input type="text" name="valor" id="uno" />
        <input type="submit" value="Comprobar" />
    </form>
</body>
</html>
```

$.inArray() → Busca un valor en una matriz y devuelve la posicion del mismo si es que se encuentra o -1 en caso contrario. La función recibe como argumentos el valor a buscar y la matriz donde buscarlo.

Ejemplo: La siguiente página muestra dos campos de texto donde el usuario debe introducir un nombre y una contraseña almacenados en posiciones en las matrices *usuarios* y *claves* correlativamente. Mediante JQuery se comprueba que si el usuario indicado existe en la matriz *usuarios*, y su contraseña coindice con la indicada.

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title></title>
    <script src="Scripts/jquery-3.0.0.js"></script>
    <meta charset="utf-8" />
     <script>
         var usuarios = ["Roger", "Ivan", "Yuri"];
         var claves = ["ROG", "IVN", "YRI"];
         $().ready(function () {
             // Evento de envio de datos de formulario
             $("#formulario").submit(function (event) {
                 // Obtencion del usuario indicado
                 var usuario = $("form input[name='usuario']").val();
                 // Obtención de la contraseña indicada
                 var clave = $("form input[name='clave']").val();
                  // Obtención de indice de usuario indicado en matriz usuarios
                  var indice_usuario = $.inArray(usuario, usuarios);
                 // Comprobación: Existe el usuario en la matriz?
                 if (indice_usuario == -1) {
                     // Indice = -1 --> Usuario no existente en matriz usuarios
                     alert("Usuario desconocido");
                     event.preventDefault();
                 // Comprobación: Clave indicada coincide con la del usuario
                 } else if (clave != claves[indice_usuario]) {
                     // No coincide --> Clave incorrecta.
                     alert("Clave incorrecta");
                     event.preventDefault();
                 }
             });
         });
     </script>
</head>
<body>
    <form id="formulario">
        Usuario<input type="text" name="usuario" /><br/>
        Clave<input type="text" name="clave" />
        <input type="submit" value="Comprobar" />
    </form>
</body>
</html>
```

$.trim() → Retorna la cadena indicada como argumento eliminando los espacios en blanco a la izquierda y derecha.

```
                var con_espacios = " HOLA ";
                var sin_espacios = $.trim(con_espacios);
                // Muesta "HOLA";
                alert(sin_espacios);
```

$.each() → Devuelve uno a uno los valores almacenados en una matriz u objeto de Javascript. Esta función recibe como argumentos la matriz u objeto, y una función que es invocada para cada valor. Dicha función recibe dos parámetros:

- *idx* → Indica el índice o nombre del atributo.
- *val* → Indica el valor asociado

Ejemplo: El siguiente código muestra el uso de la función ***$.each()*** para obtener y mostrar todos los valores de una matriz y los atributos de un objeto.

```
<script>
    // Declaración de una matriz de elementos
    var matriz = ["foo", "bar", "baz"];
    // Declaracion de un objeto empleando JSO
    var objeto = { foo: "bar", baz: "bim" };

    $().ready(function () {
        // Muestra todos los valores de la matriz
        $.each(matriz, function( indice, valor ) {
            document.write( "element " + indice + " is " + valor + "<br />");
        });
        // Muestra todos los atributos del objeto
        $.each(objeto, function( atributo, valor ) {
            document.write( atributo + " : " + valor + "<br />" );            });
    });
</script>
```

El resultado mostrado es el siguiente:

```
element 0 is foo
element 1 is bar
element 2 is baz
foo : bar
baz : bim
```

Existen muchas más funciones de utilidad de JQuery que permiten agilizar y simplificar tareas comunes en la programación en Javascript.

Pueden consultarse en la web oficial de la API de JQuery:

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

Ejercicios

Partiendo de la página *index.html* implementar las siguientes selecciones empleando JQuery:

- Seleccionar todos los elementos *div* que poseen la clase "*module*".
- Seleccionar el tercer ítem de la lista desordenada con id= *#myList*.
- Seleccionar el elemento *<label>* del elemento *input* utilizando un selector de atributo.
- Averiguar cuantos elementos en la página están ocultos (length).
- Averiguar cuantas imágenes en la página poseen el atributo alt.
- Seleccionar todas las filas impares del cuerpo de la tabla.

(*) Para comprobar las selecciones almacenalas en una variable y muéstrala en la consola del navegador mediante el comando *Console.dir()*.