JavaScript

ANEXO 4.-

JQuery:

Manipulación y navegación por el DOM

DISTRIBUIDO POR:

CENTRO DE INFORMÁTICA PROFESIONAL S.L.

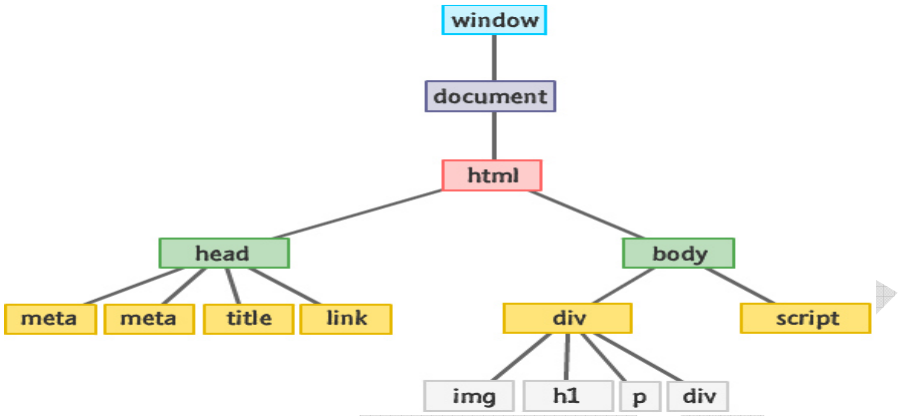
C/ URGELL, 100
08011 BARCELONA
TFNO: 93 426 50 87

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

www.cipsa.net

Manipulación del DOM

La estructura de una página web está compuesta por el conjunto de etiquetas HTML que lo componen y el orden en que se disponen. Cuando una página web se carga en el navegador, cada elemento HTML se convierte en un objeto y se crea una estructura de objetos con forma de árbol dispuestos según los elementos HTML con el objeto *Document* como raíz. Esta estructura es el ***modelo de objetos de documento (DOM).***



Estructura ejemplar del modelo de objetos de documento (DOM) de una página web clásica

JQuery permite tanto modificar el contenido y propiedades de estos objetos, como la propia estructura del documento insertando, moviendo y eliminando los objetos dentro del documento web. Para ello existen una serie de métodos especializados.

Creación de elementos.

JQuery permite crear nuevos elementos HTML con el fin de añadirlos al documento web modificando su estructura original.

Esto puede hacerse de dos modos de crear un elemento HTML:

- Invocando ***$()*** indicando como argumento el código HTML del elemento a crear:

```
$("<p>This is a new paragraph</p>");
$("<li class=\"new\">new list item</li>");
```

- Invocando ***$()*** indicando como primer argumento el elemento HTML que se desea crear, y como segundo argumento una matriz de pares *<atributo>:<valor>* con sus correspondientes atributos y valores:

```
$("<a/>", {
        html: "This is a <strong>new</strong> link",
        "class": "new",
        href: "foo.html"
    });
```

*(*) El propiedad 'class' va entrecomillada debido a que coincide con la palabra clave **class** de Javascript. De no ser así no sería necesario como en el caso de **html** y **href**.*

Los nuevos elementos creados pueden asignarse a variables para insertarlos o añadirlos al documento web. El objeto devuelto por ***$()*** es equivalente a una selección que contiene únicamente el objeto recién creado:

```
      var parrafo = $("<p>GUAY!</p>");     // Nuevo párrafo recién creado.
      var imagen =  $().ready(function () {
            // Imagen creada
            var imagen = $("<img>", {
                src: "images/ok.png"
            })
            // El nuevo elemento imagen es insertado tras cada párrafo.
            imagen.insertAfter("p");
        });
```

Inserción de elementos (*$().insertAfter() / $().after()*)

JQuery provee dos métodos para insertar un nuevo elemento a continuación de otro ya existente:

$().insertAfter()

Este método se invoca a partir del elemento creado y lo inserta después de los elementos indicados en el selector dado como argumento.

Ejemplo: El código JQuery siguiente crea un elemento imagen y lo inserta tras cada uno de los elementos *<p>* ya existentes en el documento web:

```
<!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 () {
            // Imagen creada
            var nueva_imagen = $("<img>", {
                src: "images/ok.png"
            })
            // El nuevo objeto imagen se inserta tras cada párrafo.
            nueva_imagen.insertAfter("p");
        });
    </script>
</head>
<body>
    <p id="estado1">estado 1:</p>
    <p id="estado2">estado 2:</p>
</body>
</html>
```

El resultado se correspondería con el siguiente código HTML:

```
<body>
    <p id="estado1">estado 1:</p>
    <img src="images/ok.png" />
    <p id="estado2">estado 2:</p>
    <img src="images/ok.png" />
</body>
```

$().after()

Este método inserta el elemento indicado como argumento tras los elementos de la selección contra la que se invoca.

Ejemplo: El código siguiente crea un elemento imagen y lo inserta tras cada párrafo presente en el documento web:

```
<!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 () {
            // Imagen creada
            var nueva_imagen = $("<img>", {
                src: "images/ok.png"
            });
            // Inserta el elemento imagen tras cada elemento <p>.
            $("p").after(nueva_imagen);
        });
    </script>
</head>
<body>
    <p id="estado1">estado 1:</p>
    <p id="estado2">estado 2:</p>
</body>
</html>
```

El resultado es exactamente el mismo que en el ejemplo anterior; el objeto imagen creado es insertado tras cada párrafo.

```
<body>
    <p id="estado1">estado 1:</p>
    <img src="images/ok.png" />
    <p id="estado2">estado 2:</p>
    <img src="images/ok.png" />
</body>
```

Movimiento de elementos

Los métodos de inserción vistos anteriormente también pueden emplearse para mover elementos ya existentes en el documento. Para ello, lo que debe hacerse es indicar la selección de los elementos a mover en vez de un elemento recién creado:

Ejemplo: Este script movería todos los elementos *<img>* tras cada elemento *<p>:*

```
// Obtención de los objetos imágenes a mover.
var imagenes = $("img");
// Cada objeto imagen se mueve tras cada objeto parrafo
imágenes.insertAfter("p");
// Versión reducida
$("img").insertAfter("p");
```

Otra opción:

```
// Mueve cada elemento imagen seleccionado tras cada elemento párrafo.
$("p").after($("img"));
```

Ejemplo: La siguiente página mueve todos los elementos imagen *<img>* de la página web y los sitúa tras cada párrafo *<p>*:

```
<!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 () {
            $("p").after($("img"));
        });
    </script>
</head>
<body>
    <img src="images/ok.png" />
    <p id="estado1">estado 1:</p>
    <p id="estado2">estado 2:</p>
</body>
</html>
```

El resultado mostrado sería el equivalente al siguiente código HTML:

```
<body>
    <p id="estado1">estado 1:</p>
    <img src="images/ok.png" />
    <p id="estado2">estado 2:</p>
    <img src="images/ok.png" />
</body>
```

Clonado de elementos

Si se necesita duplicar un elemento en vez de moverlo, sólo es necesario encadenar el método ***$().clone()***. El método retorna una nueva selección con copias de los elementos presentes en la selección contra la que se invoca.

```
<!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 () {
            // Seleccion de elementos <img>, clonado, y reinsercion tras cada elemento <p>
            $("img").clone().insertAfter("p");
        });
    </script>
</head>
<body>
    <img src="images/ok.png" />
    <p id="estado1">estado 1:</p>
    <p id="estado2">estado 2:</p>
</body>
</html>
```

El resultado sería equivalente al siguiente código HTML:

```
    <img src="images/ok.png" />
    <p id="estado1">estado 1:</p>
    <img src="images/ok.png" />
    <p id="estado2">estado 2:</p>
    <img src="images/ok.png" />
```

Inclusión de elementos.

La inclusión de elementos consiste en añadir uno o varios elementos dentro de otros *al final* o *al principio* según los métodos empleados:

$().appendTo()

Añade al final de cada de uno de los elementos de la selección indicada como argumento el/los elementos de la selección contra la que es invoca.

Ejemplo 1: El siguiente código JQuery agrega al final de una lista *<select>* con *id="opciones"* una elemento opción nuevo <option> con el valor *1* y el texto "*OPCION*".

```
        $().ready(function () {
            // Crea el nuevo elemento opcion <option>
            var nuevaOpc = $("<option>", { value: 1, text: "OPCION" });
            // Lo agrega al final del contenido de la lista.
            nuevaOpc.appendTo($("#opciones"));
        });
```

Ejemplo 2: La siguiente página web muestra una lista *<select>* a la que se agrega mediante un script y JQuery las opciones *<option>* de los días de la semana:

```
<!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 () {
            var dias = ["lunes", "martes", "miercoles",
                        "jueves", "viernes", "sabado", "domingo"];
            // Bucle de agregado de opciones a la lista con id="dias".
            for (var i = 0; i < dias.length; i++) {
                $("<option>", { value: i, text: dias[i] }).appendTo($("#dias"));
            }
        });
    </script>
</head>
<body>
    <img src="images/ok.png" />
    <select id="dias">
        <option>Seleccione un pais</option>
    </select>
</body>
</html>
```

El resultado sería el mismo que el del siguiente código HTML:

```
    <select id="dias">
        <option>Seleccione un pais</option>
        <option value="0">lunes</option>
        <option value="1">martes</option>
        <option value="2">miercoles</option>
        <option value="3">jueves</option>
        <option value="4">viernes</option>
        <option value="5">sabado</option>
        <option value="6">domingo</option>
    </select>
```

$().append()

Añade el/los elementos de la selección indicada como argumento al final del/los elementos en la selección contra la que se invoca:

Ejemplo: El siguiente código agrega a la lista *<select id="opciones">* una nuevo elemento opción *<option value="1">OPCION</option>*

```
        $().ready(function () {
            // Crea el nuevo elemento opcion <option>
            var nuevaOpc = $("<option>", { value: 1, text: "OPCION" });
            // Lo agrega al contenido de la lista.
            $("#opciones").append(nuevaOpc);
        });
```

$().prependTo()

Método equivalente a ***$().appendTo()*** que añade los elementos de la selección contra la que se invoca *al principio* de cada uno de los elementos de la selección indicada como argumento.

Ejemplo: La siguiente página muestra una lista *<select>* a la que se le añade un elemento opción *<option>* con valor 1 y contenido "*OPCION*" al principio:

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <script src="Scripts/jquery-3.0.0.js"></script>
    <meta charset="utf-8" />
    <script>
        $().ready(function () {
            // Crea el nuevo elemento opcion <option>
            var nuevaOpc = $("<option>", { value: 1, text: "OPCION" });
            // Lo agrega al principio del contenido de la lista.
            nuevaOpc.prependTo($("#opciones"));
        });
    </script>
</head>
<body>
    <img src="images/ok.png" />
    <select id="opciones">
        <option>Seleccione un pais</option>
    </select>
</body>
</html>
```

El resultado mostrado coincide con el siguiente código HTML:

```
    <select id="opciones">
        <option value="1">OPCION</option>
        <option>Seleccione un pais</option>
    </select>
```

$().prepend() → Método equivalente a ***$().append()*** que añade los elementos de la selección argumento al principio de cada uno de los elementos en la selección contra la que se invoca.

```
        $().ready(function () {
            // Crea el nuevo elemento opcion <option>
            var nuevaOpc = $("<option>", { value: 1, text: "OPCION" });
            // Lo agrega al contenido de la lista.
            $("#opciones").prepend(nuevaOpc);
        });
```

Eliminación de elementos.

$().remove()

Este método elimina todos y cada uno de los elementos del selector contra el que se invoca. También puede indicársele como argumento un selector para restringir el conjunto de elementos a eliminar.

```
        // Elimina los elementos con la hoja de estilo 'estilo'
        $(".estilo").remove();
        // Elimina los elementos con la hoja de estilo 'estilo' dentro de capas.
        $("div").remove(".hello");
```

$().detach()

Este método elimina del documento web el/los elementos de la selección contra la que se invoca, pero retorna al mismo tiempo la selección para reinsertarlos nuevamente en el documento web.

Ejemplo: La siguiente página muestra un párrafo junto con un botón botón. Cada vez que se pulsa el botón, el párrafo es eliminado y reinsertado alternativamente.

```
<!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>
        var p = null;
        $().ready(function () {
            $("#boton").click(function () {
                // Esta el párrafo en memoria.
                if (p == null) {
                    // Elimina el parrafo del documento pero lo mantiene en memoria.
                    p = $("#parrafo").detach();
                } else {
                    // Reinserta el parrafo tras el botón.
                    p.insertAfter("#boton");
                    p = null;
                }
            })
        });
    </script>
</head>
<body>
    <p id="parrafo">parrafo</p>
    <input id="boton" type="button" value="PULSAME" />
</body>
</html>
```

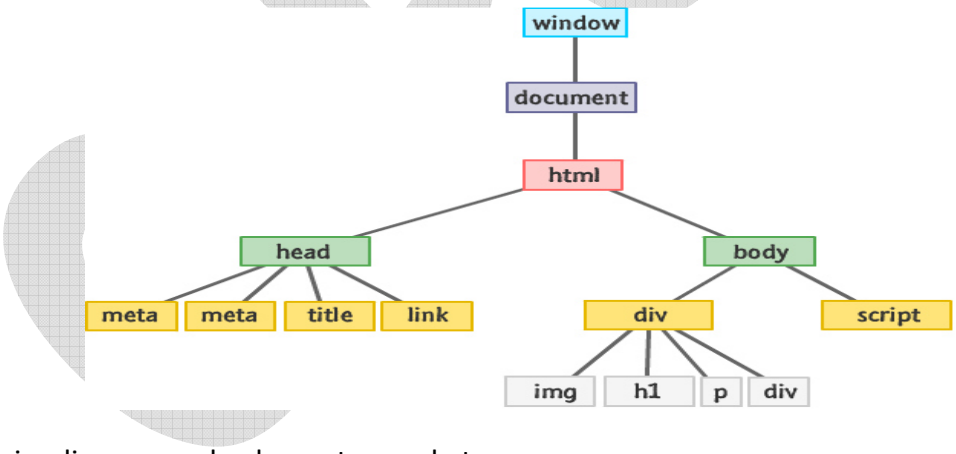
Navegación en el DOM

JQuery dispone de métodos que permiten navegar por los diferentes elementos que componen el documento web. Para ello se sigue la estructura jerárquica del documento dividida en elementos *padres*, *hijos* y *hermanos*.

Ejemplo: Sea la siguiente página web:

```
<!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>
</head>
 <body>
     <div id="base" >
          <img id="imagen" src="images/imagen0.png" />
          <h1>PARRAFO</h1>
          <p>Texto</p>
          <div>
          </div id="subcapa">
     </div>
     <script>
     </script>
 </body>
</html>
```

La representación del DOM correspondiente sería la siguiente:



Esto implica que cada elemento puede tener:

- ***Elemento padre*** → Aquel elemento HTML en el que está contenido, por ejemplo: una capa. En el ejemplo anterior la imagen *id="imagen"* tiene por padre la capa *id="base".*

- ***Elementos hijos*** → Aquellos elementos HTML comprendidos dentro de un elemento. En el ejemplo anterior, la capa id="base" tiene como elementos hijos la imagen *id="imagen"*, un párrafo, un encabezado *<h1>*, y otra capa.

- ***Elementos hermanos*** → Aquellos elementos HTML que tienen el mismo padre. En el ejemplo, el elemento imagen *id="imagen"*, el párrafo, el encabezao *<h1>*, y la capa *id="subcapa"* son hermanos.

JQuery dispone de métodos que permite obtener los elementos padres, hijos y hermanos de un determinado elemento:

Obtención de elementos padre.

- ***$().parent()*** → Devuelve el elemento padre de un elemento seleccionado.
- ***$().parents()*** → Devuelve todos los elementos padre incluyendo los indirectos (elementos padres del elemento padre). Este método acepta un selector como argumento para restringir los elementos a devolver.
- ***$().parentsUntil()*** → Devuelve todos los elementos padres hasta uno determinado por el selector indicado como argumento sin incluirlo.

En ambos casos, si se invoca el método contra una selección de múltiples elementos, se devuelven los padres de todos y cada uno de ellos:

```
    <div class="grandparent">
        <div class="parent">
            <div class="child">
                <span class="subchild"></span>
            </div>
        </div>
        <div class="surrogateParent1"></div>
        <div class="surrogateParent2"></div>
    </div>
```

```
// Retorna el elemento [ div.child ]
$("span.subchild").parent();

// Retorna los elementos [ div.child, div.parent, div.grandparent ]
$("span.subchild").parents();

// Retorna el elemento [ div.parent ]
$("span.subchild").parents("div.parent");

// Retorna los elementos [ div.child, div.parent ]
$("span.subchild").parentsUntil("div.grandparent");
```

Obtención de elementos hijos.

- ***$().children()*** → Devuelve los elementos hijos de un elemento seleccionado.
- ***$().find()*** → Devuelve todos los elementos hijos de un elemento seleccionado incluyendo los indirectos (elementos hijos de sus elementos hijos).

```
    <div class="grandparent">
        <div class="parent">
            <div class="child">
                <span class="subchild"></span>
            </div>
        </div>
        <div class="surrogateParent1"></div>
        <div class="surrogateParent2"></div>
    </div>
```

```
// Retorna los elementos [ div.parent, div.surrogateParent1, div.surrogateParent2 ]
$("div.grandparent").children("div");

// Retorna los elementos [ div.child, div.parent, div.surrogateParent1, div.surrogateParent2 ]
$("div.grandparent").find("div");
```

Obtención de elementos hermanos.

- ***$().next()*** → Devuelve el siguiente elemento hermano respecto al seleccionado.
- ***$().prev()*** → Devuelve el anterior elemento hermano respecto al seleccionado.
- ***$().nextAll()*** → Devuelve todos los elementos hermanos anteriores
- ***$().prevAll()*** → Devuelve todos los elementos hermanos previos.

```
    <div class="grandparent">
        <div class="parent">
            <div class="child">
                <span class="subchild"></span>
            </div>
        </div>
        <div class="surrogateParent1"></div>
        <div class="surrogateParent2"></div>
    </div>
```

```
// Retorna el elemento [ div.surrogateParent1 ]
$("div.parent").next();

// Retorna una seleccion vacía puesto que el elemento indicado no tiene hermanos.
$("div.parent").prev();

// Retorna los elementos [ div.surrogateParent1, div.surrogateParent2 ]
$("div.parent").nextAll();

// Retorna el elemento [ div.surrogateParent1 ]
$("div.parent").nextAll().first();

// Retorna el elemento [ div.surrogateParent2 ]
$("div.parent").nextAll().last();

// Retorna los elementos [ div.surrogateParent1, div.parent ]
$("div.surrogateParent2").prevAll();

// Retorna el elemento [ div.surrogateParent1 ]
$("div.surrogateParent2").prevAll().first();

// Retorna el elemento [ div.parent ]
$("div.surrogateParent2").prevAll().last();
```

Ejercicios

Ejercicios de Navegación

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

- Seleccionar todas las imágenes en la página mostrando en la consola el valor del atributo "*alt*" de cada imagen.
- Seleccionar el elemento *input* del formulario y añadirle la clase "*input_text*".
- Seleccionar el ítem que posee la clase "*current*" dentro de la lista *id=myList*, eliminarlo de la lista, y añadir la clase "*current*" al siguiente ítem de la lista.
- Seleccionar el elemento select dentro del elemento con "*id=specials"*, y a continuación navegar por el DOM hasta el elemento del botón *submit*.
- Seleccionar el primer ítem de la lista con *id="#slideshow"*; añadirle la clase "*current*" al mismo y luego añadir la clase "*disabled*" al resto de elementos hermanos.

Ejercicios de Manipulación

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

- Añadir 5 nuevos ítems al final de la lista desordenada con *id="myList"* empleando un bucle.
- Remover los ítems impares de la lista.
- Añadir un elemento *h2* y un párrafo al final de la capa con el estilo *"module"*.
- Añadir otra opción al elemento select con el valor "*Wednesday*".
- Añadir una nueva capa con el estilo "*module*" a la página después del último; luego añadir una copia de una de las imágenes existentes dentro de la nueva capa.