UNIDAD 7.2 - JQUERY

JQuery. Introducción

JQuery es una de las librerías de Javascript **más utilizada** tal y como se puede observar en el análisis de W3techs. Permite **interactuar con los documentos HTML** de una manera muy sencilla, pudiendo así manipular el DOM, manejar eventos, añadir animaciones, etc. y facilitando enormemente el trabajo con Javascript.

Se trata de **software libre y código abierto** licenciado bajo la Licencia MIT y la Licencia Pública General de GNU v2, por lo que puede ser utilizada tanto en proyectos libres como privados.

Para incluir jQuery en nuestros proyectos podemos hacerlo de dos maneras:

- **Descargando los archivos fuente de jQuery de la web oficial** e incluyéndolos en nuestro proyecto, pudiendo referenciarlos de manera interna: esta opción nos permite no depender de Internet ni que cambios en las versiones hagan fallar nuestro código.
- **Haciendo referencia directamente a la url pública del CDN(red de entrega de contenidos)** donde se encuentra la última versión de jQuery: dependemos de Internet pero nos aseguramos que siempre estamos haciendo referencia a una versión correcta

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

A la hora de incluir cualquiera de las opciones podemos hacerlo con:

- La **versión comprimida** (o **minificada**), que reduce en gran medida el tamaño del archivo.

- La **versión descomprimida**, que es recomendable si queremos disponer del código fácilmente visualizable.

La clave principal para el uso de jQuery radica en el uso de la función **$()**, que es un alias de **jQuery()**.

Esta función se podría comparar con el clásico ***document.getElementById()***, pero con una diferencia muy importante, ya que soporta selectores CSS, y puede devolver arrays. Por lo tanto **$()** es una versión mejorada de ***document.getElementById().***

Esta función **$("selector")**, acepta como parámetro una cadena de texto, que será un selector CSS, pero también puede aceptar un segundo parámetro, que será el contexto en el cuál se va a hacer la búsqueda del selector citado.

Otro uso de la función, puede ser el de **$(function(){..});** equivalente a la instrucción **$(document).ready(function() {...});** que nos permitirá detectar cuando el DOM está completamente cargado.

Por último veremos la sintaxis de las sentencias en jQuery, diferenciando tres opciones:

- La sintaxis con una **función anónima.**
- La sintaxis con una **función con nombre.**
- La sintaxis **reducida**.

```
<!DOCTYPE html>
<html>
<head>
    <!-- JQUERY
    - Librería de Javascript que permite "escribir menos y hacer más".
    - Hace más sencillo el uso de JavaScript en una web: reduce el código.
    - Simplifica el trabajo con DOM y AJAX. -->

    <meta charset="utf-8" />
        <!--script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"> </script>
        <script
src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.1.1.min.js"></script>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script> version 3.6.0 en
jquery.com
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

```
Versión 3.5.1 en google
<script src="https://code.jquery.com/jquery.min.js"></script> Ultima versión
-->

        <script src="js/jquery-3.1.1.min.js"></script>
    <script>
        /* SINTAXIS: $(selector).accion() */

        /* ESPERAR A LA CARGA DEL HTML */
        //Con función anónima
        /*$(document).ready(function(){
            alert("Página cargada correctamente");
        });*/
        //Con función con nombre
        $(document).ready(inicio);
        function inicio(){
            alert("Página cargada correctamente");
            document.getElementById("hola").innerHTML = "¡Hola, mundo!";
        }

        //Versión reducida
        /*$(function(){
            alert("Página cargada correctamente");
        });*/
    </script>
    </head>

<body>
    <p id="hola"></p>
</body>
</html>
```

JQuery. Selectores y Eventos

Con jQuery podremos hacer referencia a elementos indicando su **etiqueta**, **identificador**, sus **clases**, o incluso **pseudoselectores**; todo ello de una única vez, sin necesidad de recorrer elementos, hacer bucles, o acceder a arrays, tal y como hacíamos con **Javascript**. Además, veremos cómo guardar los **elementos seleccionados en una variable** para su uso posterior.

Ver posibles selectores de JQuery en:

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

y muchos ejemplos en:

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

Del mismo modo, veremos cómo aplicar manejadores de eventos a un selector, al igual que hacíamos en **Javascript** con *addEventListener*, donde mencionaremos:

- Eventos de **ratón**: click, dblclick, mouseover, mouseout, mouseenter, mouseleave.
- Eventos de **teclado**: keypress, keydown, keyup.
- Eventos relacionados con **formularios**: submit, change, focus, blur.
- Eventos asociados a **document/window**: load, resize, scroll, unload.

Aquí podemos ver una lista de los eventos más comunes:

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

y una lista de todos los eventos:

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

Veamos el funcionamiento en un ejemplo en el que integraremos **selectores y eventos**,

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scr
ipt>

    <script>
        $(document).ready(inicio);
        function inicio(){
            /* SELECTORES
            $("p") //Selecionar elementos por su etiqueta
            $("#primero") //Seleccionar elementos por su id
            $(".importante") //Seleccionar elementos por su clase
            $("p[name=primero]") //Seleccionar elementos por su atributo
            $("a[target='_blank']") //Seleccionar elementos por pseudoselectores */

            //GUARDAR UNA SELECCIÓN EN UNA VARIABLE
            var parrafos = $("p"); //Guarda los elementos EN EL MOMENTO

            /* EVENTOS*/

            $("p").mouseover(function(){
                //$("p").css("color","red");
                $(this).css("color","red");
            });
            $("p").mouseout(function(){
                //$("p").css("color","black");
                $(this).css("color","black");
            });

            $("p#primero").click(function(){
               alert("Has pulsado el primer párrafo")
            });
            $("p.importante").click(function(){
                $(this).hide();
            })
        }
    </script>
    </head>

<body>
    <p id="primero">Parrafo 1</p>
    <br/>
```

```
    <p>Parrafo 2</p>
    <br/>
    <p class="importante">Parrafo 3</p>
</body>
</html>
```

Más opciones avanzadas para la gestión de eventos de forma similar a como funciona addEventListener en JS es el metodo ***.on()***, que permite **adjuntar uno o más manejadores a uno o más elementos seleccionados**. Este método a su vez puede contener una función con o sin parámetros e incluso funciones anónimas: veremos cómo utilizarlo en cada uno de los casos.
También veremos el método ***.one()***, que únicamente **permite que el evento se ejecute una única vez** para cada elemento de la selección.

Además, **trabajaremos con varios eventos asociados** al mismo selector utilizando el mismo ***.on(),*** que nos permitirá mostrar un código mucho más limpio y comprensible.

A su vez, en caso de que necesitemos **eliminar el manejador de eventos asociado a un elemento**, utilizaremos el método ***.off()***.

Por último veremos qué son ***.trigger()*** y ***.trigglerHandler()***, que simulan la ejecución de un evento.

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scr
ipt>

    <style>

    </style>
    <script>
        $(document).ready(inicio);

        function inicio() {
            //Llamada a una función sin parámetros
            $("p").on("click", mensaje);
```

```
            function mensaje() {
                alert("Párrafo pulsado");
            }

            //Llamada a una función con parámetros
            $("p").on("click", {
                nombre: "Ada",
                apellido: "Lovelace"
            }, mensajeParametros);

            function mensajeParametros(e) {
                alert(e.data.nombre + " " +
                    e.data.apellido);
            }

            //Ejecución de una función anónima
            $("p").on("click", function () {
                alert($(this).text());
            });

            //.one: el evento se ejecuta una única vez PARA CADA ELEMENTO de la
selección
            $("p").one("click", function () {
                alert("El párrafo " + $(this).text() + " ha sido pulsado por primera
vez");
            });

            //Varios eventos asociados al mismo selector
            $("p").on({
                mouseenter: function () {
                    $(this).css("background-color", "lightgray");
                },
                mouseleave: function () {
                    $(this).css("background-color", "lightblue");
                },
                click: function () {
                    $(this).css("background-color", "yellow");
                }
            });

            //.off: eliminamos el manejador de eventos.
            $("#quitarEvento").click(quitarEvento);

            function quitarEvento() {
                $("p").off();
            }
            //.trigger: simula la ejecución de un evento(click, mouseover)
            //.triggerHandler: tiene diferencias respecto a trigger(ver ayuda)
            $("#cuenta1").click(function () {
                $("#contador1").text(parseInt($("#contador1").text()) + 1);
            });
            $("#cuenta2").click(function () {
```

```
                $("#contador2").text(parseInt($("#contador2").text()) + 1);
                $("#cuenta1").trigger("click");
            });

            //Otras: preventDefault()...
        }
    </script>
</head>

<body>
    <button id="quitarEvento">Quitar evento</button>
    <button id="cuenta1">Cuenta 1</button>
    <button id="cuenta2">Cuenta 2</button>
    <p>Parrafo 1</p>
    <p>Parrafo 2</p>
    <p>Parrafo 3</p>
    <p>Parrafo 4</p>
    <p>Contador 1: <span id="contador1">0</span></p>
    <p>Contador 2: <span id="contador2">0</span></p>
</body>

</html>
```

JQuery. Filtrado

En la lección anterior aprendimos cómo **trabaja jQuery con los selectores**. Pues bien, sobre estas agrupaciones de elementos podemos realizar una serie de filtros que nos permitirán escoger ciertos elementos en función de unas u otras características.

Los métodos de filtrado con los que trabajaremos son los siguientes:

- ***.has():*** devuelve elementos de un tipo seleccionado que **contienen otros elementos** que se identifican con el selector incluido entre paréntesis.
- .***not():*** similar al anterior, pero opera sobre elementos que **NO contienen otros elementos** identificados por el selector incluido entre paréntesis.
- ***.filter():*** opera sobre **elementos que coinciden con la búsqueda**, de manera que elige de entre los elementos que ya han sido seleccionados aquellos que cumplen con la condición.

- ***.find():*** devuelve los **descendientes** de los elementos que coinciden con la búsqueda.
- ***.first:*** devuelve el **primer elemento** de una lista.
- ***.last:*** devuelve el **último elemento** de una lista.
- .***eq (número):*** devuelve el elemento que se encuentra en la **posición indicada** entre paréntesis.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <script src="js/jquery-3.1.1.min.js"></script>
    <script>
        $(document).ready(inicio);
        function inicio(){
            /* FILTRADO DE SELECCIONES:
            //.has: opera sobre elementos que contienen otros elementos incluidos en el has
            $("div.textos").has("p"); //Devuelve div con id textos que tengan dentro un p por lo menos
            //.not: opera sobre elementos que NO contienen otros elementos indicados en el has
            $("p").not(".importante"); //Devuelve párrafos sin clase importante
            //.filter: opera en elementos que coinciden con la búsqueda
            $("p").filter(".importante"); //Devuelve p con clase importante
            //.find: devuelve descendientes de un elemento
            $("p#primero").find("span"); //Devuelve los span del p#primero

            //.first, .last: Devuelve primer o último elemento de un alista
            $("p").first(); //Devuelve el primer párrafo
            //.eq(número): Devuelve el elemento que coincide con la posición indicada
            $("p").eq(5); //Devuelve el sexto párrafo*/

            /* ENCADENAMIENTO DE SELECCIONES */
            $("div#textos") //Etiqueta con id=textos
                .find("p") //Solo párrafos
                .eq(0) //Primer párrafo
                .html("TEXTO CAMBIADO EN PRIMER PÁRRAFO") //Cambio el html
                .end() //Restablece a los elementos del párrafo
                .eq(2) //Tercer párrafo
                .html("TEXTO CAMBIADO EN TERCER PÁRRAFO");
        }
    </script>
    </head>

<body>
    <div id="textos">
        <p id="primero">Parrafo 1</p>
```

```
        <p>Parrafo 2</p>
        <h3>Titulo</h3>
        <p class="importante">Parrafo 3</p>
    </div>
</body>
</html>
```

JQuery. HTML

veremos **cómo extraer información de un elemento** mediante:

- ***.html()***: extrae el código HTML.
- .***text()***: extrae el texto de la etiqueta.
- ***.val()***: extrae el valor de un elemento de tipo input.

También veremos cómo **modificar y añadir información a un elemento**:

- ***.html(«valor»)***: añade a un elemento el código html introducido entre paréntesis.
- ***.text(«valor»)***: añade a un elemento el texto introducido entre paréntesis.
- ***.val(«valor»)***: introduce el valor entre paréntesis en el elemento input.

Además, trabajaremos con **atributos de un elemento** mediante los siguientes métodos:

- ***.attr(«atributo»)***: extrae la información del atributo cuyo nombre está indicado entre paréntesis.
- ***.attr(«atributo», «valor»)***: añade el valor del segundo parámetro al atributo cuyo nombre está indicado en el primer parámetro.
- ***.attr({ «atributo1»: «valor1», «atributo2»: «valor2»})***: permite añadir o modificar la información de más de un atributo simultáneamente.

Por último, veremos las ***funciones callback*** que pueden recibir como parámetro los métodos **.text**, **.html**, **.val** y **.attr**. Estas contienen dos parámetros: el índice del elemento actual en la lista de elementos seleccionados y el valor original.

Ejemplo con función callback:

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

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scr
ipt>

    <script>
        $(document).ready(inicio);
        function inicio(){
            //EXTRAER INFORMACIÓN DE UN ELEMENTO (GET): paréntesis sin parámetros
            console.log($("p#primero").html()); //Extraigo código html
            console.log($("p#primero").text()); //Extraigo texto
            console.log($("input").val()); //Extraigo el valor de un input

//MODIFICAR/AÑADIR INFORMACIÓN DE UN ELEMENTO (SET): valor entre paréntesis
            $("p#primero").html("Nuevo valor del párrafo 1"); //Introducir código
html
            $("p#primero").text("Nuevo valor del párrafo 1"); //Introducir texto
            $("input").val("Nuevo valor del input"); //Introducir un valor en un
input

            //EXTRAER INFORMACIÓN DE UN ATRIBUTO (GET)
            console.log($("a").attr("href"));
             //MODIFICAR/AÑADIR INFORMACIÓN DE UN ATRIBUTO (SET)
            //Un solo atributo
            /* $("a").attr("href","todoslosenlaces.html");
            //Varios atributos
            $("a").attr({
                "title":"Todos los enlaces",
                "href":"todoslosenlaces2.html"
            }); */
//FUNCIONES CALLBACK: los métodos text, html, val y attr tienen como segundo
parametro una función //callback con dos parámetros: el índice del elemento actual
en la lista de elementos seleccionados, y el valor original

        $("button#cambiar").click(function(){
               $("a").attr("href",function(i,original){
                   return original + "/nuevo";
               })
            });
        }
    </script>
</head>

<body>
    <div id="textos">
```

```
        <h1>Cabecera</h1>
        <p id="primero">Parrafo 1</p>
        <p>Parrafo 2</p>
        <h3>Titulo</h3>
        <p class="importante">Parrafo 3</p>
        <input value="Texto del input"/><br/>
        <a href="http://www.miweb.com">Enlace</a><br/>

        <button id="cambiar">Cambiar</button>
    </div>
</body>
</html>
```

JQuery. CSS

veremos cómo trabajar con las propiedades CSS de estos elementos.

Así, veremos el método css, que según los parámetros pasados ejecuta una u otra operación:

- *.css*(“propiedad”): entre paréntesis pasamos el nombre de la propiedad; si está compuesta por dos palabras deberemos escribirlas en minúsculas separadas por un guión o en formato camelCase. Por ejemplo: «font-size» o «fontSize». Nos permite extraer el valor de esa propiedad.
- *.css*(“propiedad”:”valor”): permite modificar una única propiedad indicada en el primer parámetro, asignándole el valor indicado en el segundo parámetro.
- *.css*({“propiedad1”:”valor1”,”propiedad2”:”valor2″…}):permite modificar más de una propiedad de un elemento simultáneamente.

Además, veremos cómo modificar clases CSS:

- *.addClass*(«nombre de la clase»): añade una clase al elemento.
- *.removeClass*(«nombre de la clase»): borra una clase del elemento.
- *.toggleClass*(«nombre de la clase»): añade una clase si no la tiene; de lo contrario, la elimina.
- *.hasClass*(«nombre de la clase»): devuelve true o false si el elemento tiene la clase.

Por último, mencionaremos algunos métodos propios para cambiar dimensiones CSS, como son:

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

- *.width()*
- *.height()*

- .innerWidth()
- .innerHeight()
- .outerWidth()
- .outerHeigth()

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
     $("p#primero").html("Nuevo valor del párrafo 1"); //Introducir código
html

    <style>
        .verde {
            color: green;
        }
    </style>
    <script>
        $(document).ready(inicio);

        function inicio() {
            //EXTRAER PROPIEDADES CSS
            //Utilizando estilos CSS
            console.log($("h1").css("font-size")); //Devuelve el nº de píxeles de la
fuente h1
            //Utilizando formato CamelCase
            console.log($("h1").css("fontSize")); //Devuelve el nº de píxeles de la
fuente h1

            //MODIFICAR PROPIEDADES CSS
            //Una sola propiedad
            $("h1").css("fontSize", "40px");
            //Varias propiedades
            $("h1").css({
                "fontSize": "40px",
                "color": "red"
            });

            //MODIFICAR CLASES CSS
            $("h3").addClass("verde"); //Añade una clase
            $("h3").removeClass("verde"); //Borra una clase
            $("h3").toggleClass("verde"); //Añade una clase si no la tiene; sino la
elimina
            $("h3").hasClass("verde"); //Devuelve true/false si contiene la clase

            $("h3").click(function () {
                $(this).toggleClass("verde");
            });
```

```
            //MÉTODOS PROPIOS PARA CAMBIAR DIMENSIONES CSS
            //.width(), .heigth(), .innerWidth(), .innerHeight(), .outerWidth(),
.outerHeight();


        }
    </script>
</head>

<body>
    <h1>Titulo 1</h1>
    <h3>Titulo3</h3>
</body>

</html>
```

JQuery. Traversing. Recorrido y manipulación de elementos

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

Traversing significa «moverse a través» y se utiliza para encontrar o seleccionar elementos HTML teniendo en cuenta la relación de éstos con otros elementos. Comenzaremos a con una selección de uno o más elementos y nos moveremos a través de ella hasta que lleguemos al que estamos buscando.

Podemos hablar de las siguientes relaciones de elementos:

- ***Current***, o elemento actual.
- ***Ancestors***, es decir, antecesores o predecesores (nodos padre, abuelo, etc.).
- ***Descendants***, o descendientes (nodos hijo, nieto, etc.).
- ***Siblings***, o nodos hermano (nodos que comparten el mismo nodo padre).

jQuery provee de una gran variedad de **métodos que nos permiten atravesar el DOM**, la mayoría de ellos utilizando el **árbol DOM del documento**, entre los que se encuentran:

- Antecesores:
  - ***.parent()***: padre directo.
  - ***.parents()***: todos los antecesores hasta html.
  - ***.parentsUntil()***: todos los antecesores hasta uno en concreto.
  - ***.closest()***: antecesor más cercano que tiene esa etiqueta.
- Descendientes:
  - ***.children()***: hijos directos.
  - ***.find()***: hijos que cumplen con una condición.
  - ***.siblings()***: todos los hermanos.
  - ***.next()*** y ***.prev()***: siguiente hermano y hermano anterior.
  - ***.nextAll()*** y .***prevAll()***: todos los siguientes hermanos y todos los hermanos anteriores.
  - ***.nextUntil()*** y ***.prevUntil()***: el siguiente o anterior hermano hasta uno en concreto.

- Filtros:
  - ***.has()***: elemento que contiene elementos indicados en el selector pasado entre paréntesis.
  - ***.not()***: elemento que no cumple con el criterio indicado entre paréntesis.
  - ***.first()***: primer elemento que cumple con un criterio.
  - ***.last()***: último elemento que cumple con un criterio.
  - ***.eq(número)***: elemento que se encuentra en la posición indicada entre paréntesis.
- Añadir elementos:
  - ***.add()***: añade elementos a una selección (No los crea, sólo los añade a la selección para poder hacer algo sobre todos ellos).

Métodos: [jQuery Traversing Methods]

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <script src="js/jquery-3.1.1.min.js"></script>
    <style>
        .ancestors * {
            display: block;
            border: 2px solid lightgrey;
            padding: 5px;
            margin: 15px;
        }
    </style>
    <script>
        $(document).ready(inicio);

        function inicio() {
            //ASCENDIENTES:
            //.parent(): padre directo
            //$("span").parent().css("border", "2px solid blue");
            //.parents(): todos los antecesores hasta html.
            //$("span").parents().css("border", "2px solid blue");
            //.parentsUntil(): todos los antecesores hasta uno en concreto
            //$("span").parentsUntil("div").css("border", "2px solid blue");
            //.closest("etiqueta"): antecesor más cercano con esa etiqueta
            //$("span").closest("ul").css("border", "2px solid blue");
```

```
            //DESCENDIENTES:
            //.children(): devuelve hijos directos. Podemos pasar parámetros
            //$("li").children().css("border", "2px solid red");
            //$("li").children("i.cur").css("border", "2px solid red");
            //.find(): hijos que cumplan con una condición
            //$("div").find("b").css("border", "2px solid red");

            //HERMANOS:
            //.siblings(): todos los hermanos
            //$("i").siblings().css("border", "2px solid green");
            //.next(), prev()
            //$("i").next().css("border", "2px solid green");
            //.nextAll(), prevAll()
            //$("span").nextAll().css("border", "2px solid green");
            //.nextUntil(), prevUntil()
            //$("span").nextUntil("b").css("border", "2px solid green");

            //FILTRADO
            //.has, .not, .first, .last, .eq(número)

            //AÑADIR ELEMENTOS A UNA SELECCIÓN: ¡Ojo, no crea elementos!
            //.add(): entre paréntesis selector, elemento, html, selección...
            $("span").add("b").add("div").css("border", "2px solid orange");
        }
    </script>
</head>

<body class="ancestors">body (tatarabuelo)
    <div style="width:500px;">div (bisabuelo)
        <ul>ul (abuelo)
            <li>li (padre)
                <span>span</span> <i class="cur">cursiva</i>
<b>negrita</b><u>subrayado</u>
            </li>
        </ul>
    </div>
</body>

</html>
```

Veremos ahora **cómo manipular elementos** de una manera muy sencilla. con los siguientes métodos:

- ***.append***, ***.appendTo*** ***.prepend***: añade y mueve elementos como hijos.
- ***.before*** y ***.after***: añade y mueve elementos como hermanos.
- ***.clone()***: clona un elemento.

- ***.insertAfter()***, ***.insertBefore()***: inserta un elemento antes o después de otro.
- ***.remove()***: borra el elemento y sus hijos.
- ***.empty()***: borra el contenido y los hijos del elemento.

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scr
ipt>

    <style>
        .origen {
            border: 2px solid blue;
            padding: 10px;
        }

        .origen,
        .hijo {
            padding-left: 10px;
            color: green
        }

        .hermano {
            color: red
        }
    </style>
    <script>
        $(document).ready(inicio);

        function inicio() {
            //.append/.prepend: añade/mueve elementos como HIJOS
            //.before/.after: añade/mueve elementos como HERMANOS
            alert("Creamos estructura con append/prepend/before/after");
            $("div")
                .append("<div class='hijo'>1.Append</div>")
                .prepend("<div class='hijo'>2.Prepend</div>")
                .before("<div class='hermano'>3.Before</div>")
                .after("<div class='hermano'>4.After</div>");

            alert("Movemos el primero al final");
            //$(elemento).appendTo(destino);
            $("ul li:first").appendTo("ul");
            alert("Movemos el primero al final otra vez");
            //$(destino).append(elemento)
            $("ul").append($("ul li:first")); //¡¡OJO, le pasamos el elemento!!
```

```
            //CLONAR ELEMENTOS
            //$(elemento).clone();
            alert("CLonamos el primer elemento y lo añadimos al final");
            $("ul li:first").clone().appendTo("ul");

            //CREAR ELEMENTOS
            var enlace1 = $("<a href='http://www.google.com'>Mi enlace</a>");
            var enlace2 = $('<a/>', {
                html: "Mi <strong>otro</strong>enlace",
                "class": "nuevo", //Class entre comillas
                href: "http://www.google.com"
            });
            var enlace3 = $("<a href='http://www.google.com'>Mi tercer enlace</a>");

            alert("Añadimos el primer enlace");
            $("p").append(enlace1);
            alert("Añadimos el segundo enlace");
            enlace2.appendTo($("p"));
            alert("Añadimos tercer enlace después de ul");

            //Insertar y posicionarnos después de un elemento
            enlace3.insertAfter("ul");
            $("li").after("<li>Nuevo li</li>"); //Crea y añade a la vez

            //ELIMINAR ELEMENTOS
            //$(elemento).remove(); Borra el elemento y sus hijos
            alert("Borramos la lista");
            $("ul").remove();
            //$(elemento).empty(); Borra el contenido (hijos) del elemento
            alert("Borramos los hijos del div");
            $("div.origen").empty();
        }
    </script>
</head>

<body>
    <div class="origen">Texto del div
    </div>
    <ul>
        <li>Elemento 1</li>
        <li>Elemento 2</li>
        <li>Elemento 3</li>
        <li>Elemento 4</li>
    </ul>
    <p></p>
</body>

</html>
```

JQuery. Arrays

Para trabajar con arrays trabajaremos con los siguientes métodos:

- ***$.each*** y ***$.map***: nos permiten **recorrer y manipular** elementos de un array u objeto.
- ***$.inArray***: devuelve el **índice de un valor** en un array, o -1 si el valor no se encuentra en el array.
- ***$.merge***: permite **unir dos arrays**.
- ***$.grep***: **busca** un elemento que cumpla con una característica y devuelva *true* o *false*.
- ***$.makeArray***: **convierte** un array jQuery en un array Javascript.

Además, **sobre un selector** que devuelve uno o más elementos, puedes realizar las siguientes operaciones:

- ***.each***: recorre los elementos seleccionados y ejecuta para cada uno de ellos la función pasada por parámetro.
- ***.length***: devuelve el número de elementos de un array.
- .***get***(): extrae el elemento de una posición indicada por parámetro.
- ***.index()***: extrae el número de un elemento respecto a sus hermanos.

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scr
ipt>

    <style>

    </style>
    <script>
        /* ESPACIO DE NOMBRES $.fn

Cuando usamos selecciones utilizamos métodos del espacio de nombres $.fn, y forman
parte del objeto JQuery.
```

```
Ej. $("div")


ESPACIO DE NOMBRES $

Cuando no usamos selecciones, utilizamos métodos del espacio de nombres $, y forman
parte del núcleo del objeto JQuery.

Ej. $.each o jQuery.each

¡Ojo! Hay métodos de $.fn y de $ con el mismo nombre: ¡no los confundas!

*/

        $(document).ready(inicio);

        function inicio() {
            //Recorrer y manipular elementos de un array/objeto
            //$.each(array/obj,function(indice,elem){});
            //$.map(array/obj,function(elem,indice){});
            //¡OJO! .each es un iterador inmutable que devuelve el array original.
            //      .map devuelve un array nuevo basado en el original con los
cambios.

            //Objeto: utiliza pares clave:valor
            var obj = {
                "nombre": "Ada",
                "apellido": "Lovelace"
            };
            $.each(obj, function (clave, valor) {
                alert(clave + " : " + valor);
            });

            var array = [1, 3, 5, 7];
            $.each(array, function (indice, valor) {
                alert(indice + " : " + valor);
            });

            //Número de elementos de un array
            alert("Número de elementos del array " + array.length);

            //Buscar un valor en un array
            //$.inArray(elem,array): devuelve el índice de un valor en un array, o
-1 si el valor no está en el array
            if ($.inArray(3, array) !== -1) {
                alert("Valor encontrado en la posición " + $.inArray(3, array));
            } else {
                alert("El valor no se encuentra en el array");
            }

            //Unir dos arrays
            //$.merge([1,2,3],[4,5,6]);
```

```
            //--------------ARRAYS DE ELEMENTOS JQUERY --------------------
            //Recorrer un array de elementos de un selector
            alert("Recorrer un array de elementos de un selector 'li'");
            $("#recorrer").click(function () {
                $("li").each(function () {
                    alert($(this).text());
                })
            });

            //.length: número de elementos de un array
            alert("Número de elementos li: " + $("li").length);

            //.get(num): extraer el elemento de una posición
            var elem = $("li").get(2);
            alert("El elemento número 3 es de tipo " + elem.nodeName + " y contiene
" + elem.innerHTML);

            //.index(): extraer el número de un elemento respecto a sus hermanos
            $("li").click(function () {
                alert("Has pulsado el elemento " + $(this).index());
            });

            //$.grep: buscar un elemento que cumpla con una característica:
true/false
            alert("Buscamos los elementos que tengan índice mayor que 1");
            var filtrado = $.grep($("li"), function (elem, indice) {
                return indice > 1; //Devolver los elementos con índice > 1
            });
            $.each(filtrado, function (indice, elemento) {
                alert(indice + " : " + elemento.innerHTML);
            });

            //$.makeArray(array): convertir un array jQuery en un array Javascript
            alert("Convertimos un array jQuery en Javascript y lo invertimos");
            var arrayJS = $.makeArray($("li"));
            arrayJS.reverse();
            for (var i = 0; i < arrayJS.length; i++) {
                alert(arrayJS[i].innerHTML);
            }
        }
    </script>
</head>

<body>
    <ul>
        <li>Elemento 1</li>
        <li>Elemento 2</li>
        <li>Elemento 3</li>
        <li>Elemento 4</li>
    </ul>
    <button id="recorrer">Recorrer lista</button>
</body>
```

```
</html>
```

JQuery. Efectos

Para crear **efectos**, utilizaremos la siguiente **sintaxis** sobre el elemento o elementos elegidos:

```
$(selector).efecto(velocidad)
```

Donde **velocidad** puede ser *slow*, *fast*, o expresada en un número de milisegundos.

Además, podemos crear **nuestras propias velocidades** con la siguiente sentencia donde *muyRapido* es el nombre que damos a nuestra velocidad:

```
jQuery.fx.speeds.muyRapido = 50;
```

Pero veamos ahora otros efectos que podemos crear utilizando jQuery:

- .***hide()***: oculta el elemento.
- ***.show()***: muestra el elemento.
- ***.toggle()***: muestra el elemento si está oculto y viceversa.
- ***.fadeIn()***: muestra un elemento con un fundido de entrada.
- ***.fadeOut()***: oculta un elemento con un fundido de salida.
- ***.fadeToggle()***: oculta un elemento con un fundido de salida si está visible o lo muestra con un fundido de entrada si está oculto.
- ***.fadeTo()***: realiza un fundido hasta una opacidad indicada con un parámetro numérico entre 0 y 1.
- ***.slideDown()***: muestra un elemento con un efecto de persiana de arriba a abajo.
- ***.slideUp()***: oculta un elemento con un efecto de persiana de abajo a arriba.
- ***.slideToggle()***: muestra u oculta un elemento con un efecto persiana en función de si está o no visible.

```
<!DOCTYPE html>
```

```
<html>

<head>
    <meta charset="utf-8" />
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scr
ipt>

    <style>
        #panel {
            background-color: #e5eecc;
            border: solid 1px #c3c3c3;
        }
    </style>
    <script>
        $(document).ready(inicio);

        function inicio() {
            //Sintaxis de efectos y animaciones
            //$(selector).efecto(velocidad) //"slow", "fast", nº de ms
            //jQuery.fx.speeds
            jQuery.fx.speeds.myRapido = 50;

            //Ocultar
            $("#ocultar").click(function () {
                //$("p").hide();
                $("p").hide(1500);
            });
            //Mostrar
            $("#mostrar").click(function () {
                $("p").show("fast");
            });
            //Toggle
            $("#toggle").click(function () {
                $("p").toggle();
            });
            //**************************
            //fadeIn
            $("#fadeIn").click(function () {
                $("#div1").fadeIn();
                $("#div2").fadeIn("slow");
                $("#div3").fadeIn(3000);
            });
            //fadeOut
            $("#fadeOut").click(function () {
                $("#div1").fadeOut();
                $("#div2").fadeOut("slow");
                $("#div3").fadeOut(3000);
            });
            //fadeToggle
            $("#fadeToggle").click(function () {
                $("#div1").fadeToggle();
                $("#div2").fadeToggle("slow");
```

```
    $("#div3").fadeToggle(3000);
});
//fadeTo
$("#fadeTo").click(function () {
    $("#div1").fadeTo("slow", 0.15);
    $("#div2").fadeTo("slow", 0.5);
    $("#div3").fadeTo("slow", 0.8);
});

//**************************
//slideDown
$("#slideDown").click(function () {
    $("#panel").slideDown();
});
//slideUp
$("#slideUp").click(function () {
    $("#panel").slideUp();
});
//slideToggle
$("#slideToggle").click(function () {
    $("#panel").slideToggle();
});

//ANIMACIONES ************************
//$(selector).animate({parametros},velocidad,function_callback);
$("#animar").click(function () {
    $("#div1").animate({
        left: '250px',
        opacity: '0.5',
        height: '150px',
        width: '200px'
    });
    //.delay(ms): introducir un retardo
    $("#div1").delay(2000);
    //$("#div1").delay(2000).animate({...});
    $("#div1").animate({
        height: '100px',
        width: '100px',
        opacity: '1'
    }, "slow");
    $("#div1").animate({
        fontSize: '+=10'
    }, 2000);
});

//Parar una animación
//$(selector).stop(stopAll, goToEnd);
$("#pararAnimar").click(function () {
    $("#div1").stop(true, true);
});

//Funciones callback
$("#callback").click(function () {
```

```
                $("p").hide(1000, function () {
                    alert("El párrafo ha sido ocultado");
                });
            });
            /*$("#callback").click(function(){
                $("p").hide(1000);
                alert("El párrafo ha sido ocultado");
            });*/

            //Encadenamiento de efectos
            $("#encadenar").click(function () {
                $("p").css("color", "green").hide(2000).delay(500).show(2000);
            });
        }
    </script>
</head>

<body>
    <p>Texto</p>
    <button id="ocultar">Ocultar</button>
    <button id="mostrar">Mostrar</button>
    <button id="toggle">Toggle</button>
    <button id="fadeIn">FadeIn</button>
    <button id="fadeOut">FadeOut</button>
    <button id="fadeToggle">fadeToggle</button>
    <button id="fadeTo">fadeTo</button>
    <button id="slideDown">SlideDown</button>
    <button id="slideUp">SlideUp</button>
    <button id="slideToggle">SlideToggle</button>
    <button id="animar">Animación</button>
    <button id="pararAnimar">Parar animación</button>
    <button id="callback">Prueba Callback</button>
    <button id="encadenar">Prueba encadenar</button>
    <div id="div1" style="width:80px;height:80px;background-color:red;">¡Hola!</div>
    <br>
    <div id="div2" style="width:80px;height:80px;background-color:green;"></div>
    <br>
    <div id="div3" style="width:80px;height:80px;background-color:blue;"></div>
    <div id="flip">Pulsa para bajar la persiana</div>
    <div id="panel">¡Hola, mundo!</div>
</body>

</html>
```

JQuery. Animaciones

Usaremos el método ***.animate()***, que tiene la siguiente sintaxis:

```
$(selector).animate({parametros}, [velocidad], [function_callback])
```

Donde **parámetros** definen las **propiedades CSS** que se quieren modificar; la **velocidad**, opcional, indica el **tiempo que queremos que dure** la animación y, como en el caso de los efectos, puede ser *«slow»*, *«fast»*, nuestra propia velocidad definida o un número de milisegundos; y la función ***callback***, también opcional, indica la función que debe ser ejecutada **una vez que se finaliza la animación**.

Es importante entender que **las sentencias Javascript se ejecutan una tras otra**; pero en el caso de los efectos, la siguiente línea de código puede ejecutarse incluso si el efecto anterior no ha finalizado, por lo que **pueden ocurrir errores**. Para evitarlo usamos las funciones de ***callback*** que se ejecutan después de que el efecto actual haya finalizado.

En esta lección trabajaremos también con el método ***.stop()***, que nos permite **parar una animación creada en un momento dado**. Este método, a su vez incluye dos parámetros opcionales: *stopAll*, un booleano que indica si queremos parar todos las animaciones de la cola; y *goToEnd*, que indica si queremos completar la animación actual inmediatamente y por defecto es falso.

Por último, veremos **cómo podemos encadenar acciones y métodos** para conseguir efectos más largos sin crear varias líneas de código. Hasta ahora habíamos escrito sentencias jQuery una a una, pero **podemos encadenar varios métodos jQuery del mismo elemento utilizando una sola sentencia**.

```
<!DOCTYPE html>
<html>
```

```
<head>
    <meta charset="utf-8" />
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scr
ipt>

    <style>
        #panel {
            background-color: #e5eecc;
            border: solid 1px #c3c3c3;
        }
    </style>
    <script>
        $(document).ready(inicio);

        function inicio() {
            //Sintaxis de efectos y animaciones
            //$(selector).efecto(velocidad) //"slow", "fast", nº de ms
            //jQuery.fx.speeds
            jQuery.fx.speeds.myRapido = 50;

            //Ocultar
            $("#ocultar").click(function () {
                //$("p").hide();
                $("p").hide(1500);
            });
            //Mostrar
            $("#mostrar").click(function () {
                $("p").show("fast");
            });
            //Toggle
            $("#toggle").click(function () {
                $("p").toggle();
            });
            //**************************
            //fadeIn
            $("#fadeIn").click(function () {
                $("#div1").fadeIn();
                $("#div2").fadeIn("slow");
                $("#div3").fadeIn(3000);
            });
            //fadeOut
            $("#fadeOut").click(function () {
                $("#div1").fadeOut();
                $("#div2").fadeOut("slow");
                $("#div3").fadeOut(3000);
            });
            //fadeToggle
            $("#fadeToggle").click(function () {
                $("#div1").fadeToggle();
                $("#div2").fadeToggle("slow");
                $("#div3").fadeToggle(3000);
            });
```

```
            //fadeTo
            $("#fadeTo").click(function () {
                $("#div1").fadeTo("slow", 0.15);
                $("#div2").fadeTo("slow", 0.5);
                $("#div3").fadeTo("slow", 0.8);
            });

            //**************************
            //slideDown
            $("#slideDown").click(function () {
                $("#panel").slideDown();
            });
            //slideUp
            $("#slideUp").click(function () {
                $("#panel").slideUp();
            });
            //slideToggle
            $("#slideToggle").click(function () {
                $("#panel").slideToggle();
            });

            //ANIMACIONES *************************
            //$(selector).animate({parametros},velocidad,function_callback);
            $("#animar").click(function () {
                $("#div1").animate({
                    left: '250px',
                    opacity: '0.5',
                    height: '150px',
                    width: '200px'
                });
                //.delay(ms): introducir un retardo
                $("#div1").delay(2000);
                //$("#div1").delay(2000).animate({...});
                $("#div1").animate({
                    height: '100px',
                    width: '100px',
                    opacity: '1'
                }, "slow");
                $("#div1").animate({
                    fontSize: '+=10'
                }, 2000);
            });

            //Parar una animación
            //$(selector).stop(stopAll, goToEnd);
            $("#pararAnimar").click(function () {
                $("#div1").stop(true, true);
            });

            //Funciones callback
            $("#callback").click(function () {
                $("p").hide(1000, function () {
                    alert("El párrafo ha sido ocultado");
```

```
                });
            });
            /*$("#callback").click(function(){
                $("p").hide(1000);
                alert("El párrafo ha sido ocultado");
            });*/

            //Encadenamiento de efectos
            $("#encadenar").click(function () {
                $("p").css("color", "green").hide(2000).delay(500).show(2000);
            });
        }
    </script>
</head>

<body>
    <p>Texto</p>
    <button id="ocultar">Ocultar</button>
    <button id="mostrar">Mostrar</button>
    <button id="toggle">Toggle</button>
    <button id="fadeIn">FadeIn</button>
    <button id="fadeOut">FadeOut</button>
    <button id="fadeToggle">fadeToggle</button>
    <button id="fadeTo">fadeTo</button>
    <button id="slideDown">SlideDown</button>
    <button id="slideUp">SlideUp</button>
    <button id="slideToggle">SlideToggle</button>
    <button id="animar">Animación</button>
    <button id="pararAnimar">Parar animación</button>
    <button id="callback">Prueba Callback</button>
    <button id="encadenar">Prueba encadenar</button>
    <div id="div1" style="width:80px;height:80px;background-color:red;">¡Hola!</div>
    <br>
    <div id="div2" style="width:80px;height:80px;background-color:green;"></div>
    <br>
    <div id="div3" style="width:80px;height:80px;background-color:blue;"></div>
    <div id="flip">Pulsa para bajar la persiana</div>
    <div id="panel">¡Hola, mundo!</div>
</body>

</html>
```

JQuery. AJAX

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

Metodos JQuery AJAX: https://www.w3schools.com/jquery/jquery_ref_ajax.asp

Vemos la sintaxis de ***$.ajax()***, que a su vez dispone de una serie de parámetros entre los que destacan:

- **url**: a la que realizamos la petición.
- **data**: datos a enviar como objeto o cadena.
- **success**: función de respuesta si la petición ha ido correcta.
- **error**: función de respuesta si la petición ha fallado.

Además, veremos otros métodos basados en *$.ajax()* pero que disponen de parámetros ya establecidos por defecto, como son:

- ***$.get()***: solicita datos del servidor utilizando una petición HTTP GET, para lo que necesita una URL y, opcionalmente, una función de callback.
- ***$.post()***: solicita datos del servidor utilizando una petición HTTP POST, que también necesita una URL, y opcionalmente datos y una función de respuesta.

Por último, veremos los siguientes métodos que nos serán muy útiles a la hora de cargar información de manera asíncrona en nuestra página web:

- ***$.getScript()***, que obtiene y ejecuta un script de Javascript utilizando una petición AJAX de tipo HTTP GET.
- ***$.getJSON()***, que obtiene un fichero JSON utilizando una petición AJAX de tipo HTTP GET.
- ***$.load()***, que carga datos de un servidor y los coloca en el elemento seleccionado (normalmente para leer archivos html).
  https://www.w3schools.com/jquery/jquery_ajax_load.asp https://api.jquery.com/load/

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scr
ipt>

    <style>

    </style>
    <script>
        $(document).ready(inicio);

        function inicio() {
            $("#ajax").click(function () {
                var nom = $("#nombre").val();
                var ape = $("#apellido").val();
                var parametros = {
                    "nombre": nom,
                    "apellido": ape
                };
                //$.ajax: tiene un conjunto completo de parámetros
                $.ajax({
                    url: "saludo.php", //URL donde realizamos la petición
                    data: parametros, //Datos a enviar, como objeto o como cadena
                    success: function (respuesta) { //Función si la petición ha ido
bien
                        $("#mostrar").text(respuesta);
                    },
                    error: function (xhr, status) { //Función si la petición ha
fallado
                        alert("Ha ocurrido un error");
                    },
                    complete: function (xhr, status) {
                            alert("Petición realizada");
                        }
                        //Otras opciones: beforeSend, async, cache, context,
headers... (ver API)
                    });
            });
            //Otros métodos: basados en $.ajax pero tienen parámetros establecidos
por defecto
            $("#enviarGet").click(function () {
                $.get("saludo.php", {
                    "nombre": "Ada",
                    "apellido": "Lovelace"
                }, function (respuesta) {
                    $("#mostrar").text(respuesta);
                });
            });
```

```
            $("#enviarGetMensajes").click(function () {
                //Función que no envía nada y controla diferentes resultados de la
solicitud, utilizando promesas
                $.get("saludo.php", function () {
                    alert("Exito");
                }).done(function () {
                    alert("Exito 2");
                }).fail(function () {
                    alert("Error");
                }).always(function () {
                    alert("Siempre");
                });
            });

            $("#enviarPost").click(function () {
                //Función que envía datos pero no devuelve nada
                $.post("saludo.php", {
                    "nombre": "Ada",
                    "apellido": "Lovelace"
                });
                //Función que no envía datos pero recibe respuesta
                $.post("holamundo.php", function (respuesta) {
                    alert("Respuesta: " + respuesta);
                });
                //Función que envía datos y recibe respuesta
                $.post("saludo.php", {
                    "nombre": "Ada",
                    "apellido": "Lovelace"
                }).done(function (respuesta) {
                    alert("Respuesta: " + respuesta)
                });
            });

            //Añade un script y ejecuta una función del script
            $("#getScript").click(function () {
                $.getScript("script.js", function () {
                    dentroScript(); //esta función estaría dentro de script.js
                })
            });

            //Obtiene un JSON desde el servidor desde un archivo json o desde un php
//o lo que sea que lo genere
            $("#getJSON").click(function () {
                $.getJSON("json.json", function (respuesta) {
                    $.each(respuesta, function (clave, valor) {
                        alert(clave + " : " + valor);
                    });
                });
            });

            //Carga datos del servidor y el HTML lo introduce en un elemento
            $("#load").click(function () {
                $("#mostrar").load("json.php", function () {
```

```
                    alert("Se ha cargado holamundo.php en #mostrar");
                });
            });
        }
    </script>
</head>

<body>
    <input type="text" name="nombre" id="nombre" />
    <input type="text" name="apellido" id="apellido" />
    <button id="ajax">Enviar Ajax</button>
    <button id="enviarGet">Enviar Get</button>
    <button id="enviarGetMensajes">Enviar Get mensajes</button>
    <button id="enviarPost">Enviar Post</button>
    <button id="getScript">getScript</button>
    <button id="getJSON">getJSON</button>
    <button id="load">Usar Load</button>
    <p id="mostrar"></p>
</body>

</html>
```

saludo.php

```
<?php
$resultado="Bienvenido, " .$_REQUEST['nombre'] . " " .
$_REQUEST['apellido'];
echo $resultado
?>
```

json.json

```
[{"Fruta":
    [
        {"Nombre":"Manzana","Cantidad":10},
        {"Nombre":"Limon","Cantidad":20},
        {"Nombre":"Fresa","Cantidad":1}
    ]
    },
{"Verdura":
        [
            {"Nombre":"Lechuga","Cantidad":20},
            {"Nombre":"Brocoli","Cantidad":101},
            {"Nombre":"Repollo","Cantidad":10}
        ]
 }]
```