1.- Descarga la última versión del framework

Accede a la página de jQuery (http://jquery.com/download/) para descargar la última versión del framework. Dan dos posibilidades para descargar, una que le llaman PRODUCTION, que es la adecuada para páginas web en producción, puesto que está minimizada y ocupa menos espacio, con lo que la carga de nuestro sitio será más rápida. La otra posibilidad es descargar la versión que llaman DEVELOPMENT, que está con el código sin comprimir, con lo que ocupa más espacio, pero se podrá leer la implementación de las funciones del framework, que puede ser interesante en etapa de desarrollo, porque podremos bucear en el código de jQuery por si tenemos que entender algún asunto del trabajo con el framework. Verás que la descarga es un archivo js que contiene el código completo del framework. Coloca el archivo en una carpeta en tu ordenador para hacer las pruebas.

2.- Crea una página HTML simple

Ahora, en el mismo directorio donde has colocado el archivo js, coloca un archivo html con el siguiente código.

```
<html>
   <head>
      <script src="jquery-3.3.1.js" type="text/javascript"></script>
      <script>

      </script>
   </head>
<body>
<a href="http://ieslavereda.es/">http://ieslavereda.es/</a>
</body>
</html>
```

Como podrás ver, es una página bien simple, en la que tenemos una llamada a un script externo llamado jquery-3.3.1.js. Este es el código de jQuery que hemos descargado de la página del framework. Si has descargado una versión distinta, quizás el archivo se llame de otra manera, así que es posible que tengas que editar ese nombre de archivo para colocar el que tengas en el directorio.

Con ese script ya hemos incluido todas las funciones de jQuery dentro de nuestra página. Sólo tienes que prestar atención a que tanto el archivo .html de esta página, como el archivo .js del framework estén en el mismo directorio. Y, como

decía, que el archivo que incluimos con la etiqueta SCRIPT sea el .js que nosotros hemos descargado.

Además, como se puede ver, hemos dejado dentro del HEAD una etiqueta SCRIPT de apertura y cierre que está vacía. Todo el código que vamos a explicar a continuación tendrás que colocarlo en dentro de esa etiqueta.

3.- Ejecutar código cuando la página ha sido cargada

JQuery incluye una manera de hacer acciones justo cuando ya está lista la página, aunque haya elementos que no hayan sido cargados del todo. Esto se hace con la siguiente sentencia.

```
$(document).ready(function(){
   //código a ejecutar cuando el DOM está listo para recibir instrucciones.
});
```

Por dar una explicación a este código, digamos que con $(document) se obtiene una referencia al documento (la página web) que se está cargando. Luego, con el método ready() se define un evento, que se desata al quedar listo el documento para realizar acciones sobre el DOM de la página.

4. Selección de elementos

El concepto más básico de jQuery es el de "*seleccionar algunos elementos y realizar acciones con ellos*". La biblioteca soporta gran parte de los selectores CSS3 y varios más no estandarizados. En api.jquery.com/category/selectors se puede encontrar una completa referencia sobre los selectores de la biblioteca.

A continuación se muestran algunas técnicas comunes para la selección de elementos:

Selección de elementos en base a su `id`

```
$('#myId'); // notar que los IDs deben ser únicos por página
```

Selección de elementos en base al nombre de clase

```
$('div.myClass'); // si se especifica el tipo de elemento,
                  // se mejora el rendimiento de la selección
```

Selección de elementos por su atributo

```
$('input[name=first_name]'); // tenga cuidado, que puede ser muy lento
```

Selección de elementos en forma de selector CSS

```
$('#contents ul.people li');
```

Pseudo-selectores

```
// selecciona el primer elemento <a> con la clase 'external'

$('a.external:first');

// selecciona todos los elementos <tr> impares de una tabla

$('tr:odd');

//selecciona todos los elementos del tipo input dentro del formulario #my
Form

$('#myForm :input');

// selecciona todos los divs visibles

$('div:visible');

// selecciona todos los divs excepto los tres primeros

$('div:gt(2)');

// selecciona todos los divs actualmente animados

$('div:animated');
```

Nota Cuando se utilizan los pseudo-selectores `:visible` y `:hidden`, jQuery comprueba la visibilidad actual del elemento pero no si éste posee asignados los estilos CSS `visibility` o `display` — en otras palabras, verifica si el alto y ancho físico del elemento es mayor a cero. Sin embargo, esta comprobación no funciona con los elementos `<tr>`; en este caso, jQuery comprueba si se está aplicando el estilo `display` y va a considerar al elemento como oculto si posee asignado el valor `none`.

4.1. Comprobar selecciones

Una vez realizada la selección de los elementos, querrá conocer si dicha selección entregó algún resultado. Lo que se debe hacer es preguntar por la cantidad de elementos que posee la selección que se ejecutó. Esto es posible realizarlo utilizando la propiedad JavaScript `length`. Si la respuesta es `0`, la

condición evaluará falso, caso contrario (más de `0` elementos), la condición será verdadera.

Evaluar si una selección posee elementos

```
if ($('div.foo').length) { ... }
```

4.2. Guardar selecciones

Cada vez que se hace una selección, una gran cantidad de código es ejecutado. jQuery no guarda el resultado por si solo, por lo tanto, si va a realizar una selección que luego se hará de nuevo, deberá salvar la selección en una variable.

Guardar selecciones en una variable

```
var $divs = $('div');
```

NotaEn el ejemplo "Guardar selecciones en una variable", la variable comienza con el signo de dólar. Contrariamente a otros lenguajes de programación, en JavaScript este signo no posee ningún significado especial — es solamente otro carácter. Sin embargo aquí se utilizará para indicar que dicha variable posee un objeto jQuery. Esta práctica — una especie de Notación Húngara — es solo una convención y no es obligatoria.

Una vez que la selección es guardada en la variable, se la puede utilizar en conjunto con los métodos de jQuery y el resultado será igual que utilizando la selección original.

NotaLa selección obtiene sólo los elementos que están en la página cuando se realizó dicha acción. Si luego se añaden elementos al documento, será necesario repetir la selección o añadir los elementos nuevos a la selección guardada en la variable. En otras palabras, las selecciones guardadas no se actualizan *mágicamente* cuando el DOM se modifica.

4.3. Refinamiento y filtrado de selecciones

A veces, puede obtener una selección que contiene más de lo que necesita; en este caso, es necesario refinar dicha selección. jQuery ofrece varios métodos para poder obtener exactamente lo que desea.

Refinamiento de selecciones

```
$('div.foo').has('p');       //el elemento div.foo contiene elementos <p>
```

```
$('h1').not('.bar');            // el elemento h1 no posse la clase 'bar'
$('ul li').filter('.current');  // un item de una lista desordenada
                                // que posse la clase 'current'
$('ul li').first();             //el primer item de una lista desordenada
$('ul li').eq(5);               // el sexto item de una lista desordenada
```

4.4. Selección de elementos de un formulario

jQuery ofrece varios pseudo-selectores que ayudan a encontrar elementos dentro de los formularios, éstos son especialmente útiles ya que dependiendo de los estados de cada elemento o su tipo, puede ser difícil distinguirlos utilizando selectores CSS estándar.

- `:button` Selecciona elementos `<button>` y con el atributo `type='button'`
- `:checkbox` Selecciona elementos `<input>` con el atributo `type='checkbox'`
- `:checked` Selecciona elementos `<input>` del tipo checkbox seleccionados
- `:disabled` Selecciona elementos del formulario que están deshabitados
- `:enabled` Selecciona elementos del formulario que están habilitados
- `:file` Selecciona elementos `<input>` con el atributo `type='file'`
- `:image` Selecciona elementos `<input>` con el atributo `type='image'`
- `:input` Selecciona elementos `<input>`, `<textarea>` y `<select>`
- `:password` Selecciona elementos `<input>` con el atributo `type='password'`
- `:radio` Selecciona elementos `<input>` con el atributo `type='radio'`
- `:reset` Selecciona elementos `<input>` con el atributo `type='reset'`
- `:selected` Selecciona elementos `<options>` que están seleccionados
- `:submit` Selecciona elementos `<input>` con el atributo `type='submit'`
- `:text` Selecciona elementos `<input>` con el atributo `type='text'`

Utilizando pseudo-selectores en elementos de formularios

```
// obtiene todos los elementos inputs dentro del formulario #myForm

$('#myForm :input');
```

5. Trabajar con selecciones

Una vez realizada la selección de los elementos, es posible utilizarlos en conjunto con diferentes métodos. éstos, generalmente, son de dos tipos: obtenedores (en

inglés *getters*) y establecedores (en inglés *setters*). Los métodos obtenedores devuelven una propiedad del elemento seleccionado; mientras que los métodos establecedores fijan una propiedad a todos los elementos seleccionados.

5.1. Encadenamiento

Si en una selección se realiza una llamada a un método, y éste devuelve un objeto jQuery, es posible seguir un "encadenado" de métodos en el objeto.

Encadenamiento

```
$('#content').find('h3').eq(2).html('nuevo texto para el tercer elemento h
3');
```

Por otro lado, si se está escribiendo un encadenamiento de métodos que incluyen muchos paso s, es posible escribirlos línea por línea, haciendo que el código luzca más agradable para leer.

Formateo de código encadenado

```
$('#content')

    .find('h3')

    .eq(2)

    .html('nuevo texto para el tercer elemento h3');
```

Si desea volver a la selección original en el medio del encadenado, jQuery ofrece el método `$.fn.end` para poder hacerlo.

Restablecer la selección original utilizando el método $.fn.end

```
$('#content')

    .find('h3')

    .eq(2)

        .html('nuevo texto para el tercer elemento h3')

    .end()// reestablece la selección a todos los elementos h3 en #conten
t

    .eq(0)

        .html('nuevo texto para el primer elemento h3');
```

Nota El encadenamiento es muy poderoso y es una característica que muchas bibliotecas JavaScript han adoptado desde que jQuery se hizo popular. Sin embargo, debe ser utilizado con cuidado. Un encadenamiento de métodos extensivo pueden hacer un código extremadamente difícil de modificar y depurar.

No existe una regla que indique que tan largo o corto debe ser el encadenado — pero es recomendable que tenga en cuenta este consejo.

5.2. Getters y Setters

jQuery "sobrecarga" sus métodos, en otras palabras, el método para establecer un valor posee el mismo nombre que el método para obtener un valor. Cuando un método es utilizado para establecer un valor, es llamado método setter. En cambio, cuando un método es utilizado para obtener (o leer) un valor, es llamado getter.

El método $.fn.html utilizado como setter

```
$('h1').html('hello world');
```

El método html utilizado como getter

```
$('h1').html();
```

Los métodos setters devuelven un objeto jQuery, permitiendo continuar con la llamada de más métodos en la misma selección, mientras que los métodos getters devuelven el valor por el cual se consultó, pero no permiten seguir llamando a más métodos en dicho valor.

6. CSS, estilos y dimensiones

jQuery incluye una manera útil de obtener y establecer propiedades CSS a los elementos.

Nota Las propiedades CSS que incluyen como separador un guión del medio, en JavaScript deben ser transformadas a su estilo CamelCase. Por ejemplo, cuando se la utiliza como propiedad de un método, el estilo CSS `font-size` deberá ser expresado como `fontSize`. Sin embargo, esta regla no es aplicada cuando se pasa el nombre de la propiedad CSS al método `$.fn.css` — en este caso, los dos formatos (en CamelCase o con el guión del medio) funcionarán.

Obtener propiedades CSS

```
$('h1').css('fontSize'); // devuelve una cadena de caracteres como "19px"

$('h1').css('font-size'); // también funciona
```

Establecer propiedades CSS

```
// establece una propiedad individual CSS

$('h1').css('fontSize', '100px');
```

```
// establece múltiples propiedades CSS

$('h1').css({ 'fontSize' : '100px', 'color' : 'red' });
```

Notar que el estilo del argumento utilizado en la segunda línea del ejemplo — es un objeto que contiene múltiples propiedades. Esta es una forma común de pasar múltiples argumentos a una función, y muchos métodos establecedores de la biblioteca aceptan objetos para fijar varias propiedades de una sola vez.

6.1. Utilizar clases para aplicar estilos CSS

Para obtener valores de los estilos aplicados a un elemento, el método `$.fn.css` es muy útil, sin embargo, su utilización como método establecedor se debe evitar (ya que, para aplicar estilos a un elemento, se puede hacer directamente desde CSS). En su lugar, lo ideal, es escribir reglas CSS que se apliquen a clases que describan los diferentes estados visuales de los elementos y luego cambiar la clase del elemento para aplicar el estilo que se desea mostrar.

Trabajar con clases

```
var $h1 = $('h1');

 $h1.addClass('big');

$h1.removeClass('big');

$h1.toggleClass('big');

 if ($h1.hasClass('big')) { ... }
```

Las clases también pueden ser útiles para guardar información del estado de un elemento, por ejemplo, para indicar que un elemento fue seleccionado.

6.2. Dimensiones

jQuery ofrece una variedad de métodos para obtener y modificar valores de dimensiones y posición de un elemento.

El código mostrado en el ejemplo "Métodos básicos sobre Dimensiones" es solo un breve resumen de las funcionalidades relaciones a dimensiones en jQuery; para un completo detalle puede consultar api.jquery.com/category/dimensions.

Métodos básicos sobre Dimensiones

```
$('h1').width('50px');  // establece el ancho de todos los elementos H1

$('h1').width();        // obtiene el ancho del primer elemento H1
```

```
$('h1').height('50px'); // establece el alto de todos los elementos H1

$('h1').height();       // obtiene el alto del primer elemento H1

// devuelve un objeto conteniendo información sobre la posición

//del primer elemento relativo al "offset"(posición) de su elemento padre

$('h1').position();
```

7. Atributos

Los atributos de los elementos HTML que conforman una aplicación pueden contener información útil, por eso es importante poder establecer y obtener esa información.

El método `$.fn.attr` actúa tanto como método establecedor como obtenedor. Además, al igual que el método `$.fn.css`, cuando se lo utiliza como método establecedor, puede aceptar un conjunto de palabra clave-valor o un objeto conteniendo más conjuntos.

Establecer atributos

```
$('a').attr('href', 'allMyHrefsAreTheSameNow.html');

$('a').attr({

    'title' : 'all titles are the same too',

    'href' : 'somethingNew.html'

});
```

En el ejemplo, el objeto pasado como argumento está escrito en varias líneas. Como se explicó anteriormente, los espacios en blanco no importan en JavaScript, por lo cual, es libre de utilizarlos para hacer el código más legible. En entornos de producción, se pueden utilizar herramientas de minificación, los cuales quitan los espacios en blanco (entre otras cosas) y comprimen el archivo final.

Obtener atributos

```
$('a').attr('href');  // devuelve el atributo href perteneciente

                      // al primer elemento <a> del documento
```

8. Recorrer el DOM

Una vez obtenida la selección, es posible encontrar otros elementos utilizando a la misma selección. En api.jquery.com/category/traversing puede encontrar una completa documentación sobre los métodos de recorrido de DOM (en inglés *traversing*) que posee jQuery.

Nota Debe ser cuidadoso en recorrer largas distancias en un documento — recorridos complejos obligan que la estructura del documento sea siempre la misma, algo que es difícil de garantizar. Uno o dos pasos para el recorrido esta bien, pero generalmente hay que evitar atravesar desde un contenedor a otro.

Moverse a través del DOM utilizando métodos de recorrido

```
// seleccionar el inmediato y próximo elemento <p> con respecto a H1

$('h1').next('p');

// seleccionar el elemento contenedor a un div visible

$('div:visible').parent();

// seleccionar el elemento <form> más cercano a un input

$('input[name=first_name]').closest('form');

// seleccionar todos los elementos hijos de #myList

$('#myList').children();

// seleccionar todos los items hermanos del elemento <li>

$('li.selected').siblings();
```

También es posible interactuar con la selección utilizando el método `$.fn.each`. Dicho método interactúa con todos los elementos obtenidos en la selección y ejecuta una función por cada uno. La función recibe como argumento el índice del elemento actual y al mismo elemento. De forma predeterminada, dentro de la función, se puede hacer referencia al elemento DOM a través de la declaración `this`.

Interactuar en una selección

```
$('#myList li').each(function(indice, elemento) {

    console.log(

        'El elemento ' + indice +

        ' contiene el siguiente HTML: ' +

        $(elemento).html()

); });
```

9. Manipulación de elementos

Una vez realizada la selección de los elementos que desea utilizar, *la diversión comienza*. Es posible cambiar, mover, eliminar y duplicar elementos. También crear nuevos a través de una sintaxis simple.

9.1. Obtener y establecer información en elementos

Existen muchas formas por las cuales se puede modificar un elemento. Entre las tareas más comunes están las de cambiar el HTML interno o algún atributo del mismo. Para este tipo de tareas, jQuery ofrece métodos simples, funcionales en todos los navegadores modernos. Incluso es posible obtener información sobre los elementos utilizando los mismos métodos pero en su forma de método getter.

Nota Realizar cambios en los elementos, es un trabajo trivial, pero hay que recordar que el cambio afectará a todos los elementos en la selección, por lo que, si desea modificar un sólo elemento, tiene que estar seguro de especificarlo en la selección antes de llamar al método establecedor.

Nota Cuando los métodos actúan como getters, por lo general, solamente trabajan con el primer elemento de la selección. Además, no devuelven un objeto jQuery, por lo cual no es posible encadenar más métodos en el mismo. Una excepción es el método `$.fn.text`, el cual permite obtener el texto de los elementos de la selección.

- `$.fn.html` Obtiene o establece el contenido HTML de un elemento.
- `$.fn.text` Obtiene o establece el contenido en texto del elemento; en caso se pasarle como argumento código HTML, este es despojado.
- `$.fn.attr` Obtiene o establece el valor de un determinado atributo.
- `$.fn.width` Obtiene o establece el ancho en pixeles del primer elemento de la selección como un entero.
- `$.fn.height` Obtiene o establece el alto en pixeles del primer elemento de la selección como un entero.
- `$.fn.val` Obtiene o establece el valor (*value*) en elementos de formularios.

Cambiar el HTML de un elemento

```
$('#myDiv p:first')


    .html('Nuevo <strong>primer</strong> párrafo');
```

9.2. Mover, copiar y eliminar elementos

Existen varias maneras para mover elementos a través del DOM; las cuales se pueden separar en dos enfoques:

- Querer colocar el/los elementos seleccionados de forma relativa a otro elemento
- Querer colocar un elemento relativo a el/los elementos seleccionados.

Por ejemplo, jQuery provee los métodos `$.fn.insertAfter` y `$.fn.after`. El método `$.fn.insertAfter` coloca a los elementos seleccionados después del elemento que se haya pasado como argumento; mientras que el método `$.fn.after` coloca al elemento pasado como argumento después del elemento seleccionado. Otros métodos también siguen este patrón: `$.fn.insertBefore` y `$.fn.before`; `$.fn.appendTo` y `$.fn.append`; y `$.fn.prependTo` y `$.fn.prepend`.

La utilización de uno u otro método dependerá de los elementos que tenga seleccionados y el tipo de referencia que se quiera guardar con respecto al elemento que se está moviendo.

Mover elementos utilizando diferentes enfoques

```
// hacer que el primer item de la lista sea el último

var $li = $('#myList li:first').appendTo('#myList');

// otro enfoque para el mismo problema

$('#myList').append($('#myList li:first'));

// debe tener en cuenta que no hay forma de acceder a la

// lista de items que se ha movido, ya que devuelve la lista en sí
```

Clonar elementos

Cuando se utiliza un método como `$.fn.appendTo`, lo que se está haciendo es mover al elemento; pero a veces en lugar de eso, se necesita mover un duplicado del mismo elemento. En este caso, es posible utilizar el método `$.fn.clone`.

Obtener una copia del elemento

```
// copiar el primer elemento de la lista y moverlo al final de la misma

$('#myList li:first').clone().appendTo('#myList');
```

NotaSi se necesita copiar información y eventos relacionados al elemento, se debe pasar `true` como argumento de `$.fn.clone`.

Eliminar elementos

Existen dos formas de eliminar elementos de una página:
Utilizando `$.fn.remove` o `$.fn.detach`. Cuando desee eliminar de forma permanente al elemento, utilize el método `$.fn.remove`. Por otro lado, el método `$.fn.detach` también elimina el elemento, pero mantiene la información y eventos asociados al mismo, siendo útil en el caso que necesite reinsertar el elemento en el documento.

NotaEl método `$.fn.detach` es muy útil cuando se esta manipulando de forma severa un elemento, ya que es posible eliminar al elemento, trabajarlo en el código y luego restaurarlo en la página nuevamente. Esta forma tiene como beneficio no tocar el DOM mientras se está modificando la información y eventos del elemento.

Por otro lado, si se desea mantener al elemento pero se necesita eliminar su contenido, es posible utiliza el método `$.fn.empty`, el cual "vaciará" el contenido HTML del elemento.

9.3. Crear nuevos elementos

jQuery provee una forma fácil y elegante para crear nuevos elementos a través del mismo método `$()` que se utiliza para realizar selecciones.

Crear nuevos elementos

```
$('<p>Un nuevo párrafo</p>');
$('<li class="new">nuevo item de la lista</li>');
```

Crear un nuevo elemento con atributos utilizando un objeto

```
$('<a/>', {
    html : 'Un <strong>nuevo</strong> enlace',
    'class' : 'new',
    href : 'foo.html'
});
```

Note que en el objeto que se pasa como argumento, la propiedad `class` está entre comillas, mientras que la propiedad `href` y `html` no lo están. Por lo general, los nombres de propiedades no deben estar entre comillas, excepto en el caso que se utilice como nombre una palabra reservada (como es el caso de `class`).

Cuando se crea un elemento, éste no es añadido inmediatamente a la página, sino que se debe hacerlo en conjunto con un método.

Crear un nuevo elemento en la página

```
var $myNewElement = $('<p>Nuevo elemento</p>');

$myNewElement.appendTo('#content');

 // eliminará al elemento <p> existente en #content

$myNewElement.insertAfter('ul:last');

// clonar al elemento <p> para tener las dos versiones

$('ul').last().after($myNewElement.clone());
```

Estrictamente hablando, no es necesario guardar al elemento creado en una variable — es posible llamar al método para añadir el elemento directamente después de `$()`. Sin embargo, la mayoría de las veces se deseará hacer referencia al elemento añadido, por lo cual, si se guarda en una variable no es necesario seleccionarlo después.

Crear y añadir al mismo tiempo un elemento a la página

```
$('ul').append('<li>item de la lista</li>');
```

Nota La sintaxis para añadir nuevos elementos a la página es muy fácil de utilizar, pero es tentador olvidar que hay un costo enorme de rendimiento al agregar elementos al DOM de forma repetida. Si esta añadiendo muchos elementos al mismo contenedor, en lugar de añadir cada elemento uno por vez, lo mejor es concatenar todo el HTML en una única cadena de caracteres para luego anexarla al contenedor. Una posible solución es utilizar un array que posea todos los elementos, luego reunirlos utilizando `join` y finalmente anexarla.

```
var myItems = [], $myList = $('#myList');

 for (var i=0; i<100; i++) {

    myItems.push('<li>item ' + i + '</li>');

}
```

```
$myList.append(myItems.join(''));
```

9.4. Manipulación de atributos

Las capacidades para la manipulación de atributos que ofrece la biblioteca son extensos. La realización de cambios básicos son simples, sin embargo el método `$.fn.attr` permite manipulaciones más complejas.

Manipular un simple atributo

```
$('#myDiv a:first').attr('href', 'newDestination.html');
```

Manipular múltiples atributos

```
$('#myDiv a:first').attr({
    href : 'newDestination.html',
    rel : 'super-special'
});
```

Utilizar una función para determinar el valor del nuevo atributo

```
$('#myDiv a:first').attr({
    rel : 'super-special',
    href : function(indice, antiguohref) {
        return '/new/' + antiguohref;
    }
});

$('#myDiv a:first').attr('href', function(idx, href) {
    return '/new/' + href;
});
```