Jquery, la librería de Javascript

Es una librería que permite hacer múltiples acciones sobre un documento **HTML**, como manipulación de documentos **HTML** en el **DOM** y los estilos **CSS**, eventos, efectos y animaciones, trabajo con **AJAX** facilitando la vida de los desarrolladores.

Primeros pasos con JQuery

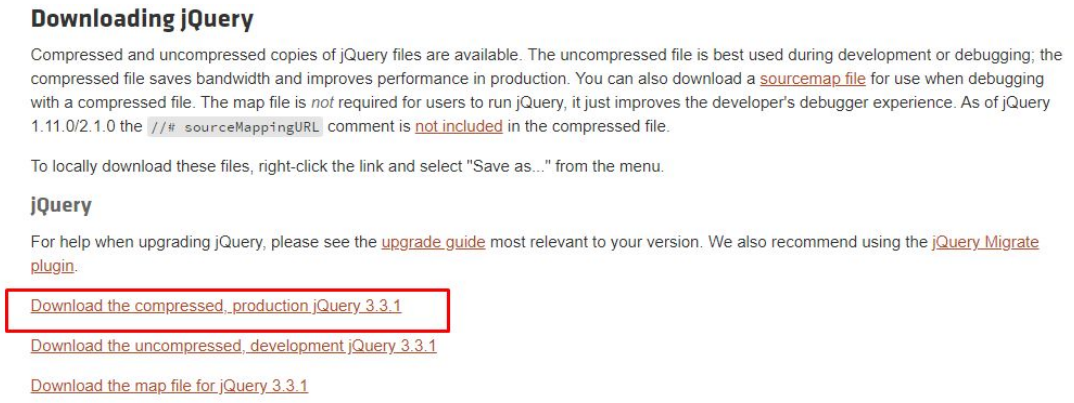
Los primeros pasos con Jquery son básicamente trabajar como con cualquier framework (estructura más compleja pero con propósitos similares a una librería), o librería.

Debemos vincularla o implementarla. Hay varias formas de hacerlo pero en principio, tenemos dos alternativas,

Trabajo desde la página oficial

Vamos a ir la página oficial de JQuery, luego haremos clic en el siguiente botón

A partir de esa acción , veremos la siguiente pantalla



Trabajaremos con la versión de producción ya que las otras versiones tiene fines distintos a los que generalmente buscamos al momento de simplemente implementar nuestra librería en nuestra web o desarrollo.

Nos llevará a la url que vemos en la imagen anterior, y tenemos dos alternativas o guardar el código en un archivo.js o también vincularlo directamente , por ejemplo

```
<script src="js/codigo.js"> </script>
```

O la segunda alternativa

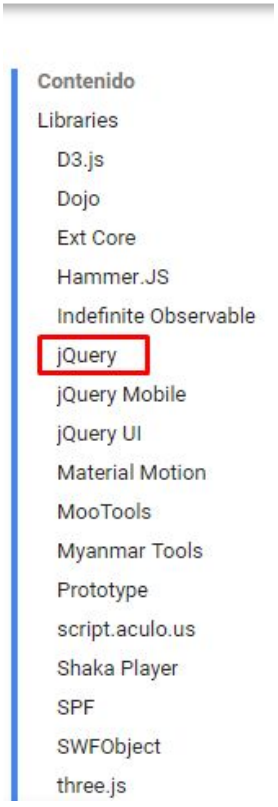
```
<script src="https://code.jquery.com/jquery-3.3.1.min.js"> </script>
```

La última versión es llamada CDN ya que es un servidor que nos permite sin necesidad de bajar el código utilizar el archivo.

Esta segunda alternativa es similar a lo que también puede ser una forma de trabajo en general con librerías

Google APIS

Podemos ir a la página de Google APIS, de esta forma, nos da acceso a múltiples rutas para trabajar con distintas librerías, trabajaremos con JQuery



A partir de esto, nos llevará a las distintas opciones, trabajaremos siempre con la producción

Luego la vinculamos como usualmente lo hacemos

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```

Este código puede trabajarse tanto en el head como en el body, lo que sí el código qué luego realicemos lo trabajaremos en el body después de los elementos que vamos a afectar.

Acceso al DOM

El acceso al **HTML** por parte de **JQuery** es una de las primeras cosas que nos genera el impacto de lo fácil y sencillo qué es acceder utilizando esta librería.
Lo principal, es entender que se utilizan los mismos selectores qué **CSS.** Si no conocés **CSS** no es un problema, es simple aplicar los selectores básicos, por supuesto, si luego tu conocimiento de CSS es mayor, ya te contamos que absolutamente todos los selectores de **CSS**, así sean simples o complejos son válidos para ser utilizados por **JQuery**.

Selectores de ID

Los selectores de **ID** acceden a todos aquellos elementos que tengan ID. El id es un tipo de selectores utilizable sólo una vez por **HTML**, por lo tanto en el mismo **HTML**, no pueden existir más de un **ID** con el mismo nombre, si por supuesto con otro nombre, pero con el mismo no. Por caso si tuviésemos un id llamado encabezado, solo un id con ese nombre podría existir.

Los ID son case sensitive eso significa que no podemos dejar de lado si está escrito en mayúscula o minúscula, por ejemplo, no es lo mismo **Encabezado** que **encabezado**.

```html
<header id="header"></header>
```

Selectores de CLASS

Los selectores de **CLASS** a diferencia de los selectores de **ID**, son reutilizables en el mismo **HTML** y sirven cuándo queremos afectar a varios elementos a la vez.

```html
<h1 class="escrito"> Titulo </h1>
<p class="escrito"> parrafo </p>
```

Selectores de Etiqueta

El selector de etiqueta, es un selector que simplemente se compone a partir del nombre de elemento de **HTML** al cual afecta, por eso a veces este tipo de selector se llama también selector de elemento.

```html
<p> parrafo </p>
<h2> Titulo de nivel 2 </h2>
```

Sintaxis de JQuery

La sintaxis de Jquery utiliza entonces los selectores de **CSS**, por ejemplo, entre otros los ya vistos (**CLASS, ID,Etiqueta**). Por ejemplo, si quisiera afectar a un elemento p, haría lo siguiente

```
$('p')
```

Si el caso fuera afectar a un **ID**,

```
$('#encabezado')
```

Si por el contrario quisiéramos afectar a una clase

```
$('.escrito')
```

Luego trabajamos siempre con métodos, la sintaxis general sería la siguiente

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$('selector').metodo(parametro)

</script>
```

El método es aquello que hacemos sobre un elemento.

Método text()

El método **text()** permite modificar los textos de un elemento, por ejemplo de la siguiente forma

```
<p> Mi párafo <strong> importante </strong> </p>
```

```
<script>
var mensaje = $('p').text()
//guardamos el contenido del parrafo
//con el método text
//en la variable mensaje

alert(mensaje)

//lo alertamos

</script>

</body>
```

De esta manera lo que hacemos es guardar en la variable mensaje el valor de texto de un párrafo. Si en todo caso queremos modificarlo haremos lo siguiente,

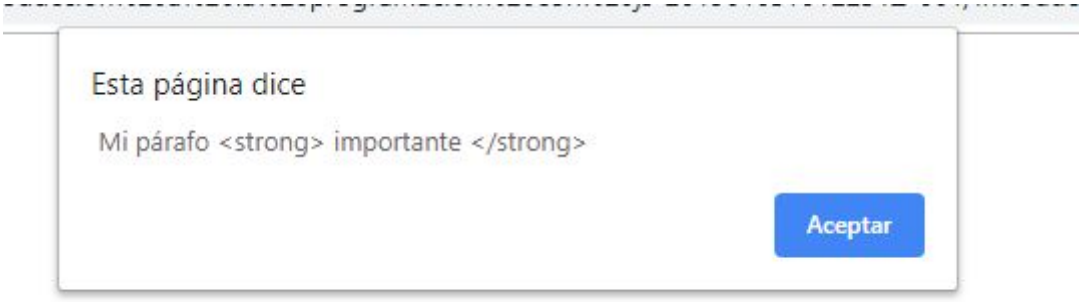
```
<script>
$('p').text('nuevo mensaje')

</script>
```

Método html()

La diferencia entre este método y el anterior, es que el método **html()** conserva el formato.
Si hiciéramos lo mismo con el texto anterior, el problema sería que en el mensaje se mostraría también que ese párrafo tiene un strong dentro. Por eso si lo que queremos es simplemente mostrar un contenido en una alerta, como la alerta es una ventana que no soporta formato lo ideal es hacerlo con el método **text()** sino pasaría lo siguiente,

```
<script>
var mensaje = $('p').html()
alert(mensaje)

</script>
```

El resultado en el navegador será el siguiente



Si lo hubiésemos hecho con el método **text()** sería

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>
var mensaje = $('p').text()
alert(mensaje)

</script>
```

En el navegador, el resultado será

Por eso lo ideal era trabajar con el método **text(),** veamos un ejemplo con **ID** cambiando el formato con **html()**

```
<script>
$('p').html('mi páraffo que ahora tiene <em> italica </em>')

</script>
```

Como el método **html()** soporta formato, podemos incluir por ejemplo esta etiqueta.

Funciones y Eventos con JQuery

Los métodos y funciones desde **JQuery** se trabajan sin el prefijo on, y son mucho más simples de generar, por ejemplo la sintaxis es la siguiente

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></
script>

<script>
$(document).ready(inicio)


function inicio(){

$('selector1').evento(nombreFuncion1)
$('selector2').evento(nombreFuncion2)
$('selector3').evento(nombreFuncion3)}


function nombreFuncion1(){}

function nombreFuncion2(){}


function nombreFuncion3(){}

</script>
```

Un caso real sería de la siguiente forma,

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
<script>
$().ready(ini)

function ini(){
$('p').click(alertar)
$('p').dblclick(alertar2)
$('h2').mouseover(alertar3)
$('h2').mouseout(alertar4)

}


function alertar1(){alert('Me hiciste click')}

function alertar2(){alert('Me hiciste doble click')}

function alertar3(){alert('Pasaste el mouse por arriba del h2')}

function alertar4(){alert('Sacaste el cursor del mouse del h2')}

</script>

</head>


<body>

<p> Haceme click o doble click </p>

<h2> Pasa el mouse por arriba o saca el cursor </h2>
```

Es decir los eventos son los mismos sólo que no los trabajamos con el **prefijo on**, y también el llamado a la función difiere. Por ejemplo algo que también se simplifica es pasar parámetros, veamos un ejemplo,

```
<script>
function alertar(parametro){

alert('Hola como estás' + parametro)

}

</script>

<p onclick="alertar('Juana')"> Haceme click </p>

<p onclick="alertar('Roberto')"> Haceme click </p>
```

Método val()

Ahora que ya trabajamos con eventos, si por ejemplo queremos obtener el valor de un elemento de formulario también todo se simplifica por ejemplo,

```
<select>

<option value="1"> Uno </option>
<option value="2"> Dos </option>
<option value="3"> Tres </option>

</select>

<script>
$('select').change(mostrarValor);

function mostrarValor(){

var valor = $('select').val()

alert(valor)

}


</script>
```

Efectos simples con JQuery

Podemos hacer que un elemento aparezca o desaparezca, también que alterne entre aparecer y desaparecer de forma simple con **JQuery**, primero generamos en el **HTML** lo siguiente

```
<div> Mi caja </div>

<button id="aparecer">Aparecer </button>
<button id="desaparecer"> Desaparecer </button>
```

En el navegador se visualizará de la siguiente manera,

Luego generamos en el código las siguientes sentencias,

```
<script>
$('#aparecer').click(aparecer)
$('#desaparecer').click(desaparecer)

function aparecer(){
$('div').show()

}

function desaparecer(){

$('div').hide()
}

</script>
```

Si queremos que el mismo botón realice ambas acciones, cambiaremos un poco nuestro HTML()

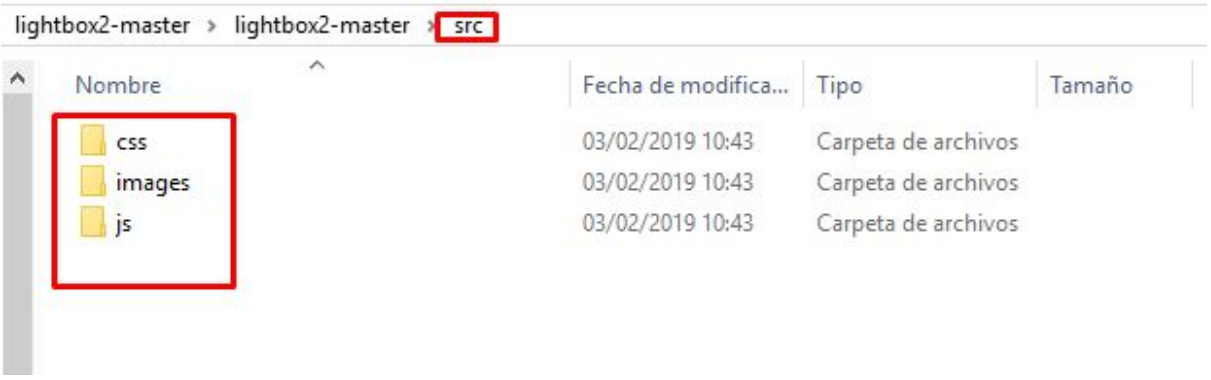
```
<div> Mi caja </div>

<button>Boton de control</button>
```

Plugins

Los plugins, son códigos formados a partir de una librería o framework en particular, en este caso JQuery para agilizar procesos y desarrollos.

En este sentido, uno de los más fáciles de implementar es el **lightbox.** Como en cualquier caso debemos implementarlo vinculandolo en nuestro body en este caso. Iremos a la Página Oficial , luego nos bajaremos los archivos haciendo clic en ,

Es importante remarcar que al descargar los archivos, se descargaron múltiples elementos muchos de ellos parte de una demo que no nos sirve al momento de trabajar más que como referencia en todo caso, por esa razón, los archivos importantes son los que se encuentran en esta carpeta,



Pasos a seguir

1# Vamos a vincular JQuery a nuestro HTML, ya que este es un plugin que depende de JQuery, en este caso para organizarnos, ya qué Lightbox trabaja en el body, vamos a trabajar con todo debajo antes de que finalice la etiqueta.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script src="js/lightbox.js"> </script>
```

Este archivo llamado lightbox lo saqué de la carpeta de demo que bajamos anteriormente, y lo coloqué en una carpeta js en mi propio proyecto. Por otro lado, debemos vincular nuestro código, donde trabajaremos,

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script src="js/lightbox.js"> </script>
<script src="js/codigo.js"> </script>




</body>
```

Prestaremos atención al hecho de que todas estas vinculaciones deben mantener el orden anterior, y encontrarse antes del cierre del body no en este caso en el head.

También crearemos en nuestro proyecto una carpeta llamada css, y dentro colocaremos otro archivo qué también nos viene en la demo lightbox llamado lightbox.css, la estructura quedará así,

Lo debemos vincular de la siguiente forma, y en este caso sí en el head de nuestro html,

```
<head>
    <meta charset="UTF-8">
    <title>Proyecto</title>

    <link href="css/lightbox.css" rel="stylesheet">

</head>
```

Tampoco debemos olvidar de colocar la otra carpeta qué es importante llamada images,

De esta forma ahora no queda más que hacer lo siguiente, la idea del lightbox es generar una galería de imágenes, por lo tanto lo primero a realizar es colocar una imagen en nuestro HTML,

```
<img src="imagenes/empleado01.png">

<img src="imagenes/empleado02.png">

<img src="imagenes/empleado03.png">

<img src="imagenes/empleado04.png">
```

La etiqueta img simplemente nos permite integrar imágenes a nuestro html, y el atributo src le dice donde se encuentran estas imágenes.

Luego **anidaremos** (colocar una etiqueta dentro de otra) , la imagen en un vínculo. Lo importante es agregarle al valor del href del vínculo la imagen que queremos que se vea en grande cuándo hacemos clic a la imagen miniatura o pequeña.

```
<a href="imagenes/empleado01-grande.png"> <img src="imagenes/empleado01.png"> </a>
```

Sumaremos ahora al vínculo un atributo a través del cual **lightbox** nos permite trabajar,

```
<a data-lightbox="grupo" href="imagenes/empleado01-grande.png"> <img src="imagenes/empleado01.png"> </a>
```

El valor de data-lightbox puede ser cualquiera, pero lo importante es entender lo siguiente, si le ponemos el mismo valor que al resto de las imágenes, se formará un grupo donde se podrá al agrandar la imagen ir también a las imágenes grandes del mismo grupo. Si por el contrario queremos que se muestren de manera individual, el valor deberá ser distinto.

Veamos un ejemplo de imágenes individuales,

```
<a data-lightbox="imagen1" href="imagenes/empleado01-grande.png"> <img src="imagenes/empleado01.png"> </a>
<a data-lightbox="imagen2" href="imagenes/empleado02-grande.png"> <img src="imagenes/empleado02.png"> </a>
```

Ahora, veamos un ejemplo de imágenes del mismo grupo,

```
<a data-lightbox="grupo" href="imagenes/empleado01-grande.png"> <img src="imagenes/empleado01.png"> </a>
<a data-lightbox="grupo" href="imagenes/empleado02-grande.png"> <img src="imagenes/empleado02.png"> </a>
```

Si queremos a su vez sumar un título o epígrafe a la imágen lo haremos de la siguiente manera,

```
<a data-lightbox="grupo" data-title="Juan Romero" href="imagenes/empleado01-grande.png"> <img src="imagenes/
empleado01.png"> </a>
```