# U.D. 6. Resumen y ejemplos jQuey

## Contenido



1. Incluir la librería jquery en nuestro desarrollo.

Para poder utilizar jQuery, necesitamos añadir la librería relacionada a nuestro desarrollo así se puede hacer descargándola de forma local o bien accediendo a un CDN como por ejemplo:

```
<script   src="https://code.jquery.com/jquery-
3.7.1.min.js"   integrity="sha256-
/JqT3SQfawRcv/BIHPThkBvs0OEvtFFmqPF/lYI/Cxo="   crossorigin="anonymous"></
script>
```

Las CDN pueden ofrecer un beneficio de rendimiento al hospedar jQuery en servidores distribuidos por todo el mundo. Esto también ofrece la ventaja de que si el visitante de su página web ya ha descargado una copia de jQuery del mismo CDN, no será necesario volver a descargarla.

Hay disponibles copias comprimidas y sin comprimir de archivos jQuery. El archivo sin comprimir se utiliza mejor durante el desarrollo o la depuración; el archivo comprimido ahorra ancho de banda y mejora el rendimiento en producción

En el siguiente enlace se puede descargarla versión 3.7.1 sin comprimir: https://code.jquery.com/jquery-3.7.1.js

Las versiones slim excluyen los módulos ajax y de efectos :

Versiones de descarga e información: https://jquery.com/download/

Para ver todos los archivos y versiones disponibles, visite: https://releases.jquery.com

Documentación de jQuery: https://api.jquery.com/

2. EJEMPLO BASE

```
<body>

    <p id="pa"> este es el primer párrafo </p>
    <p id="pb"> este es el segundo párrafo </p>
    <button id="btn">Click me</button>

</body>
```

este es el primer párrafo

este es el segundo párrafo

Click me

```
<script>
        $(document).ready(function(){
            $("#btn").click(function(){
                $("p").hide();
            });
        });
    </script>
```

Partiendo del ejemplo de la unidad, vemos como tenemos en nuestro código HTML, dos etiquetas p y un botón. Nuestro código jQuery lo que nos va a realizar es ocultar las etiquetas p al pulsar el botón.

## Documentación de jQuery: https://api.jquery.com/

3. DEPURADOR.

Pero para conseguir este código es posible que antes tengamos que realizar pruebas, así estas se pueden realizar en el depurador de nuestro navegador. Así por ejemplo si queremos ver que elementos son los que queremos seleccionar (una vez cargada la librería de jQuery) podemos realizar pruebas.

Veamos a continuación dos pruebas:

- Como seleccionar las etiquetas p, para ello dentro del depurador escribiremos: $("p") y en la siguiente imagen vemos que se devuelve:


Comprobamos como se devuelven nuestros dos objetos, etiquetas p, cada una con su identificador.

- Como seleccionar la etiqueta p segunda; podríamos hacerlo indicando su identificador.


Como se puede comprobar nos ha devuelto el objeto con identificador pb.

4. SELECTORES

Todos los selectores que aprendimos y que hemos utilizado a lo largo del curso, pueden utilizarse para la selección y manipulación de objetos con jQuery. En la unidad de trabajo estos selectores vienen organizados cómo:

- Selectores básicos.
- Selectores de atributos.
- Selectores de widgets.
- Selectores descendientes.

Como ya habréis supuesto, podremos seleccionar y posteriormente actuar sobre cualquier elemento o conjunto de elementos de nuestro código.

Así, partiendo del ejemplo inicial hemos sido capaces de ocultar todas las etiquetas p de nuestra página con una sola línea: $(“p”).hide();

## 5. MÉTODOS Y EVENTOS.

jQuery provee métodos para asociar controladores de eventos a los selectores antes mencionados. Cuando sucede un evento, se ejecuta la función expresada. Al hablar de eventos se hace referencia al momento preciso en el que ocurre algo: mover el ratón sobre un elemento, seleccionar un botón, hacer clic sobre un elemento, etc.

```
$("#btn").click(function(){

    $("p").hide();

});
```

Como hemos visto en la unidad y en nuestro ejemplo se ocultan las etiquetas p al hacer clic sobre el botón de nuestra página. Pero son múltiples los eventos que podemos utilizar, además de click

### a. Método on() y off()

El método on() nos permite gestionar eventos

En el anterior ejemplo asociamos una función a un evento, pero podemos asociar a varios eventos la misma función o podemos asociar a un mismo objeto diferentes funciones para diferentes eventos, para ellos jQuery nos ofrece el método on, así podíamos escribir nuestro código de la siguiente forma:

```
$("#btn").on("click",function(){
    $("p").hide();
});
```

Para utilizar este método de jQuery, tendremos que indicar el evento del objeto sobre el que queremos actuar y la función que se realizará al desencadenarse el evento.

Como puede comprobarse, estamos asociando el evento click del objeto #btn, para que oculte las etiquetas p.

También se puede realizar con varios eventos a la vez, por ejemplo

```
$("#btn").on("click mouseover",function(){
    $("p").hide();
});
```

El ejemplo anterior, lo hemos analizado con el evento click, pero se puede hacer también con otros eventos con change. Así, en el siguiente ejemplo podemos ver como se activará cuando un campo input de tipo checkbox se modifique o bien para marcarlo o para desmarcarlo.

En el ejemplo vemos como se activa el evento al cambiar el estado y posteriormente realizamos una pequeña comprobación donde se comprueba si la propiedad está marcada o desmarcada, eso lo realizamos con la siguiente instrucción: $(this).is(':checked')

En caso de que esté marcado mostramos el nombre del campo y el valor que tiene asociado e indicamos que está marcado, en caso contrario mostramos también nombre y valor, pero indicamos que no está marcado.

```
 $('input[type=checkbox]').on('change', function() {
            if ($(this).is(':checked')) {
                alert("El campo"+$(this).attr("name")+" tiene el valor:
"+  $(this).val()+ " y está marcado");
            } else {
                alert("El campo"+$(this).attr("name")+" tiene el valor:
"+  $(this).val()+ " y NO está marcado");
            }
});
```

Pero podemos asociar más de un evento a dicho objeto, para ello utilizaremos el siguiente ejemplo:

```
 $(document).ready(function(){
              $("#btn").on({
                  click: function(){$("p").hide();},
                  mouseover: function(){$("p").css("color","blue");},
                  dblclick:function(){$("p").show();},
              })
          });
```

En este ejemplo vemos como asociamos el evento click,  mouseover y dblclick  al objeto #btn de nuestra página.

Así:

- Al hacer click las etiquetas p desaparecerán.
- Al pasar el ratón por encima del botón cambiaremos el color de las etiquetas p.
- Al hacer doble click sobre el botón volverán a aparecer las etiquetas p.

Si quisiéramos desasociar uno de los eventos definidos anteriormente utilizaremos el método off. En el siguiente ejemplo insertamos un nuevo botón y al hacer click sobre él, desasociamos el evento mouseover y volvemos a dejar los párrafos p con color negro.

```
  $("#btn2").click(function(){
                    $("p").css("color","black");
                    $("#btn").off("mouseover");

                })
```

Otro ejemplo con on()

En el siguiente ejemplo extraído de la documentación oficial, podemos ver como podemos añadir un párrafo detrás de otro párrafo al hacer click. Como se puede observar el evento click, función incluso en los párrafos nuevos añadidos.

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>on demo</title>
  <style>
  p {
    background: yellow;
    font-weight: bold;
    cursor: pointer;
    padding: 5px;
  }
  p.over {
    background: #ccc;
  }
  span {
    color: red;
  }
  </style>
  <script src="https://code.jquery.com/jquery-3.7.0.js"></script>
</head>
<body>

  <!-- Ejemplo extraido de la documentación oficial-->
```

```
<p>Click me!</p>
<span></span>

<script>
    var count = 0;
    // Añade un elemento p después del elemento p al que sele hace click.
Se cuenta el número de párrafos añadidos.
    $( "body" ).on( "click", "p", function() {
      $( this ).after( "<p>Another paragraph! " + (++count) + "</p>" );
    });
</script>

</body>
</html>
```

Más información: https://api.jquery.com/on/#on-events-selector-data-handler

b. Método each()

Otra posibilidad que nos ofrece jQuery es la de recorrer todos los objetos que hayamos seleccionado, con un selector, para ello utilizaremos el método each.

En nuestro ejemplo vamos a añadir tres imágenes y vamos a mostrar las dimensiones de cada una de dichas imágenes.

Lo primero que hacemos es cambiar el ancho y alto de nuestras imágenes con el atributo attr

Código HTML insertado:

```
    <img class="miniaturas" src="puesta01.jpg" alt="Puesta de sol primera">
    <img class="miniaturas" src="puesta02.jpg" alt="Puesta de sol segunda">
    <img class="miniaturas" src="puesta03.jpg" alt="Puesta de sol tercera">
```

Modificamos los atributos de todas las imágenes con el siguiente código jQuery

```
    $(".miniaturas").attr("width","300px");
    $(".miniaturas").attr("height","150px");
```

Y posteriormente, vamos a mostrar en pantalla el origen de cada una de las imágenes de nuestra página:

- Con la variable txt construiremos una cadena, la variable i, será un contador por si queremos saber el número de imágenes y así numerarlas.

Como vemos, el método each nos permitirá ejecutar lo que hay dentro para cada uno de los elementos seleccionados con el selector, en nuestro caso ".miniaturas".

Para hacer referencia a cada objeto en cada una de las iteraciones utilizaremos $(this)

```
var txt="";
var origen="";
var i=1;
$(".miniaturas").each(function(index){
     origen=$(this).attr("src");
     txt=txt+"Imagen "+i+" tiene como origen: "+origen+"   "+"</br>";
     i++;
 });
 alert(txt);
```

Este código lo que nos realiza es crear una cadena con el origen de cada una de las imágenes de nuestro programa. Así, en la cadena se indica número de imagen y origen donde se encuentra la fuente de la misma. Finalizado el recorrido de las imágenes (en nuestro caso tres). Pasa a mostrar el mensaje por pantalla.

Si en lugar de mostrar el mensaje por pantalla quisiéramos imprimir en nuestra página el valor de la variable txt, haríamos lo siguiente.

```
var txt="";
var origen="";
var i=1;
$(".miniaturas").each(function(index){
     origen=$(this).attr("src");
     txt=txt+"Imagen "+i+" tiene como origen: "+origen+"   "+"</br>";
     i++;
 });

 $("body").append("<p>"+txt+"</p>");
```

Como se puede apreciar con el método append, hemos añadido el origen de nuestras imágenes al final de la página.

Otro ejemplo puede cambiar el tamaño de la imagen en función de la posición que ocupen, así por ejemplo tendríamos:

```
var txt="";
var origen="";
var i=1;
var ancho=250;
var alto=150;
$(".miniaturas").each(function(index){
     $(this).attr("width",ancho+"px");
```

```
                    $(this).attr("height",alto+"px");
                    ancho=ancho+100;
                    alto=alto+50;
                });
```

O realizar un giro con una transformación en función de la imagen.

```
var ancho=250;
                var alto=150;
                var giro=0;
                $(".miniaturas").each(function(index){
                    $(this).attr("width",ancho+"px");
                    $(this).attr("height",alto+"px");
                    alert(giro);
                    $(this).css("transform","rotate("+giro+"deg)");

                    ancho=ancho+100;
                    alto=alto+50;
                    giro=giro+15;
                });
```

Más información: https://api.jquery.com/each/#each-function

c. Método index()

Imaginemos que tenemos un conjunto de objetos o colección de objetos que han sido seleccionados. Si por ejemplo quisiéramos saber el número de objeto que hemos seleccionado podemos utilizar el método index. Este método nos devolverá la posición que ocupa dentro de los objetos seleccionados y el elemento en cuestión

Para hacer referencia al objeto en el que hemos realizado click utilizamos $(this).

Con el método eq() permite seleccionar un objeto de un conjunto de elementos, recordar que se empieza por 0. Así si tenemos 5 objetos, el primer objeto tendrá el número 0.

Si utilizamos valores negativos, empezaremos por el final del conjunto, así eq(-1) dará el objeto 5, si ponemos eq(-3) dará el objeto 3, ya que falta tres para llegar al final.

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>index demo</title>
  <style>
```

```
    .borde{
            border: solid black 2px;
            width:9vw;
            height: 15vh;
            margin:1vw;

      }
      .enlinea{
        display: inline-block;
      }


        div div{
            background: aqua;
            margin: 5px;
        }
        span {
            color: red;
        }
  </style>
  <script src="https://code.jquery.com/jquery-3.7.0.js"></script>
</head>
<body>
 <!-- Ejemplo extraido de la documentación oficial de jquery -->

<div id="bloque_1">
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>

</div>
<div id="bloque_2">
  <div></div>
  <div></div>
  <div></div>
  <div></div>
  <div></div>


</div>
<script>

    /* Le asignamos a los div dentro del bloque bloque_1 las clases borde
y enlinea */
    $("#bloque_1>div").addClass("borde enlinea");
```

```
    $("#bloque_2>div").addClass("borde");


    // El siguiente ejemplo se dispara al hacer click en uno de los
cuadros superiores
    // Detecta el orden que ocupa dentro de todos los div contenidos en
bloque_1
    // Y posteriormente en el div correspondiente de bloque 2, inserta el
orden y cambia el color y el borde del elemento correspodiente.
    $("#bloque_1>div" ).on("click", function() {
      // `this` is the DOM element that was clicked

      // Seleccionamos el objeto clicado y le cambiamos el borde
      $(this).css("border","5px solid black");
      var index = $( "#bloque_1>div").index(this);



      $("#bloque_2>div").eq(index).css("background-
color","red").css("border-radius","50%");
      $("#bloque_2>div").eq(index).append("<p>"+index+"</p>").css("text-
align","center").css("font-size","20px");

      // Podemos mejorar el código para que solo se haga una vez, así
ponemos una condición if y si el borde es redondeado no se modifica
      // en caso de utilizar color tener en cuenta que lo dará en modelo
RGB
      //var borde=$("#bloque_2>div").eq(index).css("border-radius");
      //if (borde!="50%"){
      //  $("#bloque_2>div").eq(index).css("background-
color","red").css("border-radius","50%");
      //  $("#bloque_2>div").eq(index).append("<p>"+index+"</p>").css("te
xt-align","center").css("font-size","20px");
      //}
    });
</script>

</body>
</html>
```

En este ejemplo se muestran 5 bloques horizontalmente, al hacer click en un elemento horizontal, se detecta la posición y se cambia el color, borde y se inserta el orden en los div verticales.ew

Más información sobre index(): https://api.jquery.com/index/#index

Más información sobre eq(): https://api.jquery.com/eq/

d. Método toggle()

El método Toggle de jQuery permite mostrar y ocultar elementos.

6. MODIFICACIÓN INTERACTIVA DE NUESTROS ELEMENTOS.
   a. Estilos, atributos y propiedades (css, attr y prop).

Anteriormente hemos visto como hemos cambiado los estilos o el atributo de uno o varios elementos de nuestra página. Así para ello hemos utilizado las propiedades css y attr. En el siguiente ejemplo vemos como se modifica una de las etiquetas de nuestro ejemplo

```
$("#btn3").click(function(){
    $("#pa").css("color","red");
    $("#pa").text("I. He cambiado el texto del primer parrafo");
    $("#pa").css("text-transform"," uppercase");
})
```

En este ejemplo hemos cambiado el texto de la primera etiqueta p, así cambiamos el color, el texto de la misma y además el texto hacemos que aparezca en mayúsculas.

Una vez que hemos analizado la propiedad attr, nos faltaría mencionar también la propiedad prop, que se utiliza para modificar las propiedades de un elemento. Así en el caso de un campo input de tipo checkbox si quisiéramos desmarcarlo, tendríamos que realizar lo siguiente.

```
      $("#s_img").prop("checked", false);
```

En este ejemplo lo que haríamos sería desmarcar un campo checkbox, si quisiéramos marcalor tendríamos que cambiar false por true.

b. Asignar y quitar clases.

Otra cosa que se podría realizar con nuestra página sería añadir una clase a un objeto. Así en el siguiente ejemplo vamos a crear de forma dinámica 5 div y le vamos a asignar una clase denominada borde.

Añadimos a nuestro HTML un nuevo elemento:

```
  <div id="bloque_1"></div>
```

y a nuestro código jQuery las siguientes líneas:

```
        $("#bloque_1").addClass("enlinea");


        var x=0;
        for (x=0;x<5;x++){
            $("#bloque_1").append("<div></div>");
         }
         $("#bloque_1>div").addClass("borde");
```

Ha bloque_1 le hemos asignado la clase en línea

y posteriormente creamos cinco div y a cada uno de ellos le asignamos la clase borde.

```
.borde{
        border: solid black 2px;
        width:9vw;
        height: 15vh;
        margin:5px;
        display: inline-block;
    }

    .enlinea{

        border: solid blue 2px;
        width:90vw;
        height: 20vh;
        margin:5px;
        display: inline-block;
    }
```

Posteriormente eliminamos la clase asignada con el botón eliminar clase.

Para eliminar las clases a un elemento utilizamos el método removeClass.

```
$("#btn4").click(function(){
            $("#bloque_1>div").removeClass("enlinea");
```

c. Asignar y quitar clases.

En el ejemplo anterior hemos utilizado el método append para añadir nuevos elementos a nuestra página.

7. Efectos.

Para finalizar este pequeño tutorial vamos a tratar los efectos que podemos aplicar a nuestros objetos para que aparezcan y desaparezcan:

Así hemos insertado nuevos botones para cada uno de los efectos que vamos a trabajar:

- fadeOut
- fadeIn
- slideDown
- slideUp
- hide
- show

A continuación, añadimos nuevos botones a nuestro ejemplo, para utilizar cada uno de los primeros 4 efectos que se han citado anteriormente.

```
<button id="btn5">Efecto fadeOut</button>
<button id="btn6">Efecto fadeIn</button>
<button id="btn7">Efecto slideUp</button>
<button id="btn8">Efecto slideDown</button>
```

Para cada uno de los botones vamos a aplicar un efecto a una de nuestras imágenes.

```
$("#btn5").click(function(){
    $(".miniaturas:first").fadeOut(2000);
})

$("#btn6").click(function(){
    $(".miniaturas:first").fadeIn(2000);
})

$("#btn7").click(function(){
     origen=$(".miniaturas:last").attr("src");
     $(".miniaturas:last").slideUp(100,function()
       {
         $(".miniaturas:last").attr("src",origen);
        }
      );
 });

 $("#btn8").click(function(){
     $(".miniaturas:last").slideDown("slow");
 });
```

Como se puede ver en los diferentes ejemplos hemos trabajado diferentes selectores, para seleccionar la primera o última imagen de nuestro ejemplo.

En cuanto al tiempo que pueden durar los efectos puede venir expresado en milisegundos o con los términos fast, slow, etc.

Por último, insertamos un nuevo botón donde lo que realizamos es un cambio de imagen con los efectos slideUp y slideDown:

```
    <button id="btn9">Efecto slideUp-Down</button>
```

```
            $("#btn9").click(function(){
                    origen=$(".miniaturas:first").attr("src");

                    $(".miniaturas:last").slideUp(1000,function()
                       {
                           $(".miniaturas:last").attr("src",origen);
                       }
                    );
                    $(".miniaturas:last").slideDown("slow");
            });
```

Nota: Relacionado con javascript.

La propiedad naturalWidth y naturalHeight nos permiten conocer las dimensiones reales de una imagen.

Así en el siguiente ejemplo:

Variable= document.getElementsByClassName("img_seleccionada")[0].naturalWidth;

Almacenamos en la variable con nombre variable el ancho real del primer objeto de una colección o matriz de objetos cuya clase es img_seleccionada. Es decir, con getElementsByClassName, seleccionamos todos aquellos objetos que tienen asignada la clase “img_seleccionada”.

Otras propiedades que pueden interesar son:

- getElementById
- getElementsByTagName