Unidad de Trabajo 7: JQuery

1. Introducción a JQuery

jQuery es una librería de JavaScript que simplifica significativamente la programación. Al utilizar JavaScript, es necesario garantizar la compatibilidad con varios navegadores, lo que implica escribir código compatible. jQuery aborda estos desafíos al proporcionar la infraestructura necesaria para desarrollar aplicaciones complejas en el lado del cliente. Basado en el principio de "escribe menos y produce más", éste ofrece soporte para la creación de interfaces de usuario, efectos dinámicos, AJAX, acceso al DOM, eventos, entre otras funcionalidades.

Además, jQuery cuenta con una amplia variedad de plugins que facilitan la creación de presentaciones con imágenes, validaciones de formularios, menús dinámicos, funcionalidades de arrastrar y soltar, entre otras. Esta librería es gratuita, cuenta con una licencia que permite su uso en cualquier tipo de plataforma, ya sea con fines personales o comerciales. El tamaño del archivo es de aproximadamente 31 KB y su carga es rápida. Una vez cargada, la librería queda almacenada en la caché del navegador, lo que significa que otras páginas que utilicen la misma librería no necesitarán cargarla nuevamente desde el servidor. Aquí te dejo la página principal de JQuery:

https://jquery.com/

¿Cómo incluir JQuery en tu página?

En primer lugar, para programar con JQuery se debe incorporar la librería a nuestro proyecto. Ésto se puede hacer de dos modos:

Cargando la librería directamente desde la propia web de jQuery con la siguiente instrucción:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js">
```

De esta forma, siempre nos estaremos descargando la versión más actualizada de la librería. El único inconveniente, es que necesitamos estar conectados a Internet para que la librería pueda descargarse. En la siguiente página puedes encontrar distribuciones de JQuery: https://developers.google.com/speed/libraries?hl=es-419#jquery

Cargando la librería desde nuestro propio servidor:

```
<script type="text/javascript" src="jquery.js"></script>
```

De este modo, el archivo de la librería se guarda como parte integral de nuestra aplicación, lo que significa que no necesitaremos una conexión a Internet (si estamos trabajando

localmente) para utilizar la librería. Para adoptar este enfoque, debemos descargar el archivo de la librería desde la página de jQuery (jquery.com). Hay dos versiones disponibles para descarga: la versión de producción (comprimida para ocupar menos espacio) y la versión de desarrollo (sin comprimir). En general, se descargará la versión de producción, ya que ocupa menos espacio. La versión de desarrollo únicamente ofrece la ventaja de permitir una lectura más clara del código fuente de la librería, en caso de que estemos interesados en realizar modificaciones.

La clave principal para el uso de jQuery radica en el uso de la función **$()**, que es un alias de jQuery() y se denomina, a su vez **función factoría**. 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().**

La función $("selector") toma como parámetro una cadena de texto que representa un selector CSS. También puede aceptar un segundo parámetro que define el contexto en el que se realizará la búsqueda del selector mencionado. Otra aplicación de la función puede ser $(function){..}); equivalente a la instrucción $(document).ready (function() {...});, que nos permite detectar cuando el DOM ha sido completamente cargado.

A continuación vamos a ver el mismo ejemplo desarrollado en primer lugar mediante las técnicas habituales de JavaScript y a continuación mediante el uso de JQuery, de modo que puedas comparar. La función del script en este caso es modificar el estilo de las filas en una tabla para generar una tabla con colores alternos:

Ejemplo

Método sin JQuery

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Ejemplo sin jQuery</title>
<style>
   .colorido{
      background-color:#99FF33;
   }
</style>
<script>
document.addEventListener('DOMContentLoaded',function(){
   let tabla = document.getElementById("mitabla"); // Seleccionamos la tabla.
```

```
  let filas= tabla.getElementsByTagName("tr"); // Seleccionamos las filas de la tabla.
  for (var i=0; i<filas.length; i++){
     if (i%2 == 1){// Es una fila impar aplicamos la clase .colorido a esas filas.
        filas[i].setAttribute('class','colorido');
     }
  }
});
</script>
</head>
<body>
<table width="200" border="1" align="center" id="mitabla">
  <tr>
     <td><div align="center"><strong>País</strong></div></td>
     <td><div align="center"><strong>Habitantes</strong></div></td>
     <td><div align="center"><strong>Renta</strong></div></td>
  </tr>
  <tr>
     <td>España</td>
     <td>15600000</td>
     <td>25000</td>
  </tr>
  <tr>
     <td>Italia</td>
     <td>45105500</td>
     <td>45000</td>
  </tr>
  <tr>
     <td>Francia</td>
     <td>58454545</td>
     <td>45645</td>
  </tr>
  <tr>
     <td>Uk</td>
     <td>78799788</td>
     <td>88547</td>
  </tr>
  <tr>
     <td>USA</td>
     <td>98878787</td>
     <td>45124</td>
  </tr>
```

```
</table>
</body>
</html>
```

Método con JQuery

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Ejemplo con jQuery</title>
<style>
  .colorido{
    background-color:#99FF33;
  }
</style>
<!--Primero debemos incluir la librería en nuestro código-->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
<script>
  // Cuando el documento esté preparado se ejecuta esta función:
  $(document).ready(function(){
      // Seleccionamos las filas impares contenidas dentro de mitabla y le aplicamos la clase colorido.
    $("#mitabla tr:nth-child(even)").addClass("colorido");
  });
</script>
</head>
<body>
<table width="200" border="1" align="center" id="mitabla">
  <tr>
    <td><div align="center"><strong>País</strong></div></td>
    <td><div align="center"><strong>Habitantes</strong></div></td>
    <td><div align="center"><strong>Renta</strong></div></td>
  </tr>
  <tr>
    <td>España</td>
    <td>15600000</td>
    <td>25000</td>
  </tr>
  <tr>
    <td>Italia</td>
    <td>45105500</td>
```

```
    <td>45000</td>
  </tr>
  <tr>
    <td>Francia</td>
    <td>58454545</td>
    <td>45645</td>
  </tr>
  <tr>
    <td>Uk</td>
    <td>78799788</td>
    <td>88547</td>
  </tr>
  <tr>
    <td>USA</td>
    <td>98878787</td>
    <td>45124</td>
  </tr>
</table>
</body>
</html>
```

2. ¿Qué ventajas aporta usar JQuery?

JQuery presenta diversas ventajas que facilitan el desarrollo web:

- **Cross-Browser Compatibility:** Programar en JQuery garantiza compatibilidad con diferentes navegadores. La librería detecta automáticamente el navegador y adapta el comportamiento para lograr resultados independientes de la plataforma.
- **Acceso y Modificación Eficiente del DOM:** JQuery ofrece un sólido mecanismo de selección que permite acceder y modificar elementos en un documento de manera eficiente. Facilita cambios estructurales complejos, como la reorganización de una lista no ordenada, con un código más conciso.
- **Modificación de la Apariencia con Facilidad:** Aunque CSS es poderoso, JQuery resuelve pequeñas incompatibilidades entre navegadores. Permite cambiar clases o propiedades de estilo incluso después de que la página se haya cargado. Conoce CSS, ya que utiliza el mismo mecanismo de selectores para acceder a partes del documento.

- **Manejo Elegante de Eventos de Usuario:** JQuery proporciona una forma elegante de interceptar una amplia variedad de eventos sin afectar al código HTML. Al ser una librería cross-browser, adapta los eventos según el navegador, eliminando preocupaciones sobre el uso del objeto event y sus propiedades.
- **Animaciones y Efectos Incorporados:** La librería incluye una serie de animaciones y efectos (ocultar, deslizar, desvanecer, etc.) que mejoran la interactividad de la página, proporcionando una interfaz más atractiva para los usuarios.
- **Soporte Nativo para AJAX:** JQuery dispone de sus propios métodos para realizar peticiones AJAX, eliminando la necesidad de configurar manualmente objetos XMLHttpRequest. Con funciones como $ajax{...}, la llamada queda correctamente configurada independientemente del navegador.
- **Simplificación de Tareas JavaScript Comunes:** Además de sus características específicas, JQuery mejora construcciones básicas de JavaScript, como la iteración y manipulación de tablas. La librería implementa bucles automáticos en muchos casos, simplificando tareas comunes, como ocultar elementos con la clase CSS "rotulo" en una sola línea de código.

3. Trabajar con selectores CSS3

JQuery ha ganado popularidad al simplificar el acceso al DOM desde JavaScript, evitando la complejidad asociada. En el siguiente código, ilustramos cómo asignar color rojo a todos los enlaces (etiquetas "a") de la página. Utilizamos el método JQuery css, que toma el nombre de la propiedad CSS y su valor:

Mediante JavaScript y DOM

```
var enlaces = document.getElementsByTagName("a");
for (var i = 0; i < enlaces.length; i++) {
   enlaces[i].style.color = "red";
}
```

Mediante JQuery

```
$("a").css("color", "red");
```

Ambos enfoques logran el mismo resultado, pero JQuery simplifica la tarea, evitando la necesidad de un bucle explícito. Las selecciones en JQuery pueden utilizar cualquier selector CSS, lo que facilita la manipulación y modificación de elementos en la página. JQuery crea objetos que representan colecciones de elementos seleccionados. La propiedad length se utiliza para determinar si la colección tiene elementos. Recordando que JQuery es

esencialmente JavaScript, cualquier selección puede guardarse en una variable. Por ejemplo, al buscar celdas que contengan la palabra "Enrique" en una tabla, podemos utilizar la propiedad length para obtener la cantidad de elementos encontrados:

```
let celdasEnrique = $("td:contains('Enrique')");
alert("Hay " + celdasEnrique.length + " celdas con la palabra Enrique");
```

Cabe destacar que JQuery tiene disponibles todos los selectores CSS3, además de algunos propios e incluye, además, la posibilidad de usar selectores personalizados. Por ejemplo:

```
$("p:first") // selecciona el primer párrafo
$("p:eq(0)") // selecciona el primer párrafo, aquél cuya posición es igual a cero (el primero).
$("p:odd") // selecciona todos los párrafos en posiciones impares.
$("td:contains("Enrique")") // selecciona todas las celdas de tabla que contienen "Enrique"
```

Asimismo, en JQuery se pueden emplear los llamados **filtros.** Los filtros son **métodos** que refinan selecciones. Por ejemplo:

```
$("p:first") // Selector personalizado
$("p").first() // Filtro
```

3.1. Selectores para formularios

Selector	Descripción
:button	Selecciona los elementos <button> y los <input> con type="button"
:checkbox	Selecciona todos los elementos <input> con type="checkbox"
:checked	Selecciona todos los <input> con estado checked
:disabled	Selecciona todos los elementos de formulario con estado disabled
:enabled	Selecciona todos los elementos de formulario sin estado disabled
:file	Selecciona los <input> con type="file"
:image	Selecciona los <input> con type="image"
:input	Selecciona los <input>, <textarea>, y <select>
:password	Selecciona los <input> con type="password"
:radio	Selecciona los <input> con type="radio"
:reset	Selecciona los <input> con type="reset"

:submit	Selecciona los <input> con type="submit"
:selected	Selecciona todos los <option> cuyo estado es "selected"
:text	Selecciona los <input> con type="text"

3.2. Selectores de atributos

JQuery permite también seleccionar elementos por sus atributos. En la siguiente tabla dispones de múltiples ejemplos:

Selector	Descripción
$('input[type="password"]')	Selecciona todos los <input> de tipo password.
$('input[type!="password"]')	Selecciona todos los <input> que NO son de tipo password.
$('a[href^="http://"]')	Selecciona todos los enlaces cuyo atributo href comienza por "http://".
$('a[href$=".com"]')	Selecciona todos los enlaces cuyo atributo href termina por ".com".
$('img[src*="jquery"]')	Selecciona todas las imágenes cuya URL contiene la palabra "jquery".
$("form[encoding]")	Selecciona todos los formularios que tienen un atributo encoding (sin importar su valor).

4. Iteración en JQuery

Tal y como hemos visto, JQuery incorpora los bucles implícitos. Sin embargo, en ocasiones, es necesario aplicar un tratamiento más complejo a cada elemento de la selección, accediendo a cada nodo seleccionado de manera personalizada. Para lograr esto, se pueden emplear las estructuras iterativas ya vistas en JavaScript, incluyendo la construcción .each(), que nos permite acceder a cada nodo individualmente.

5. Encadenamientos

JQuery presenta, tal y como hemos visto en cuanto a los métodos de selección del DOM, la propiedad de permitir encadenar los selectores, de modo que se simplifica notablemente el código. Por ejemplo:

```
$("table").find("td:contains("Enrique")").eq(3).addClass(".rotulo").show(); // busca las tablas y después encuentra las celdas con la palabra Enrique. De éstas coge la cuarta y le agrega la clase rótulo y por último la muestra.
```

Tal y como puedes observar, no es necesario que apliques cada selección o método paso por paso sino que puedes simplificar agrupando los mismos.

6. Getters y Setters

JQuery ofrece métodos que pueden actuar tanto como getters (para obtener valores específicos) como setters (para cambiar valores), utilizando el mismo método en realidad. El proceso es sencillo: si se invocan sin parámetros, se utiliza el método como un getter para leer una propiedad. En cambio, si se invocan con parámetros, se modifica la propiedad y el método actúa como un setter. En resumen, el mismo método puede ser empleado tanto para obtener como para cambiar valores, dependiendo de si se le proporcionan parámetros o no. Un ejemplo, es el método **html()**:

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="UTF-8">
 <title>Ejemplo de JQuery</title>
 <script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>
 <script>
  $(document).ready(function() {
   alert("Lea detenidamente: " + $('#miformulario p').html()); // html como getter
   $('#miformulario p').html('Gracias por seguir las indicaciones'); // html como setter
  });
 </script>
</head>
<body>

<form id="miformulario">
 <label for="nombre">Nombre:</label>
 <input type="text" id="nombre" value="Escriba aquí su nombre" size="30" />
 <p> En el campo nombre introduzca su nombre y también sus apellidos </p>
</form>

</body>
</html>
```

Otros métodos que se comportan como setters y como getters son, por ejemplo: **val()**, que devuelve el valor o permite modificarlo; **attr(atributo, [valor])**; **text()** y el método **css()**. Un ejemplo de este último:

```
$(document).ready( function() {
alert("Color de letra: " + $('#nombre').css('color')); // css como getter
$('#nombre').css('color', 'rgba(255,0,0,.5)'); // css como setter (rojo con semitransparencia)
$('#miformulario p').css('background-color', 'rgba(255,0,0,.5)');
});
```

En el caso de querer modificar múltiples propiedades simultáneamente, se debe utilizar una sintaxis similar a las reglas CSS tal y como se muestra en el ejemplo:

```
$('#nombre').css({'font-size':'24px','color':'rgba(255,0,0,.5)'});
```

Cabe destacar que, cuando se manipulan las propiedades de estilo mediante JQuery es útil no sólo utilizar el método CSS, el cual permite añadir propiedades o modificarlas sino también métodos que permiten agregar o eliminar clases de estilo CSS3 a los elementos seleccionados:

Método	Descripción
addClass(clase_CSS)	Agrega una clase CSS previamente definida a un elemento.
removeClass(clase_CSS)	Elimina una clase CSS de un elemento.
toggleClass(clase_CSS)	Alterna la aplicación y eliminación de una clase CSS en cada llamada al método.
hasClass(clase_CSS)	Método booleano que indica si el elemento tiene o no la clase CSS especificada.

Igualmente, existen multitud de métodos que permiten trabajar con los estilos tales como width(), height(), innerWidth(), outerWidth(), position(), scrollTop(), etc. Es recomendable, por tanto, que revises la documentación de la API cuando estés trabajando con la misma puesto que sus posibilidades son muy extensas:

https://api.jquery.com/

7. Recorrido del DOM mediante JQuery

JQuery dispone de múltiples y poderosos métodos para trabajar con el DOM. Algunos de los más empleados se especifican en las tablas a continuación:

Método	Descripción
Recorriendo el DOM de forma descendente:	
.children(selector)	Desciende un nivel en el DOM desde el elemento actual, seleccionando los hijos que cumplen con el selector especificado.
.find(selector)	Desciende varios niveles en el DOM desde el elemento actual, seleccionando los descendientes que cumplen con el selector especificado.
.has(selector)	Reduce la selección actual a aquellos nodos que tienen al menos un descendiente que cumple con el selector especificado.
Recorriendo el DOM de forma ascendente:	
.parent()	Sube un nivel en el DOM desde el elemento actual, seleccionando el padre del elemento.
.parents(selector)	Sube varios niveles en el DOM desde el elemento actual, seleccionando todos los ancestros que cumplen con el selector especificado.
.closest(selector)	Sube varios niveles en el DOM desde el elemento actual, seleccionando el primer ancestro (el más próximo) que cumple con el selector, incluyendo el propio objeto si cumple con el selector.
Filtrando selecciones:	
.first()	Filtra la selección para obtener solo el primer objeto.
.last()	Filtra la selección para obtener solo el último objeto.
Recorriendo hermanos siguientes:	
.next()	Selecciona el siguiente hermano en el DOM.
.next(selector)	Selecciona el siguiente hermano que cumple con el selector especificado.
.nextAll()	Selecciona todos los hermanos posteriores en el DOM.
.nextAll(selector)	Selecciona todos los hermanos posteriores que cumplen con el selector especificado.
.nextUntil(selector)	Selecciona todos los hermanos posteriores hasta que se cumple con un selector.

.nextUntil(selector)	Selecciona todos los hermanos posteriores hasta que se cumple con un selector.
Recorriendo hermanos anteriores:	
.prev()	Selecciona el hermano anterior en el DOM.
.prevAll()	Selecciona todos los hermanos anteriores en el DOM.
.prevUntil(selector)	Selecciona todos los hermanos anteriores hasta que se cumple con un selector.

8. Manipulación del DOM mediante JQuery

Asimismo, JQuery no sólo nos permite recorrer el árbol de nodos sino también modificarlo. Algunos de los métodos para ello son los siguientes:

Método	Descripción
Métodos para Insertar	
.append(content)	Inserta contenido al final del contenido de cada elemento en la selección.
.prepend(content)	Inserta contenido al inicio del contenido de cada elemento en la selección.
.before(content)	Inserta contenido antes de cada elemento en la selección, al mismo nivel.
.after(content)	Inserta contenido después de cada elemento en la selección, al mismo nivel.
.appendTo(target)	Inserta los elementos seleccionados al final del contenido del objetivo especificado.
.prependTo(target)	Inserta los elementos seleccionados al inicio del contenido del objetivo especificado.
.insertBefore(target)	Inserta los elementos seleccionados antes del objetivo especificado, al mismo nivel.
.insertAfter(target)	Inserta los elementos seleccionados después del objetivo especificado, al mismo nivel.
Método para clonar	
.clone()	Hace una copia del elemento y de su contenido. No obstante, no copia los eventos asignados a dicho elemento. En caso de querer clonar también los eventos debemos usar **.clone(true)**

Método para eliminar nodos o su contenido	
.remove(selector)	Elimina un nodo del DOM. Puede aceptar un selector para eliminaciones selectivas, eliminando solo los nodos que cumplen con el selector especificado.
.empty()	Vacía el contenido de un nodo, eliminando todos sus hijos.

9. Eventos en JQuery

En JQuery, contamos con métodos para aplicar eventos de manera sencilla con un solo paso. Esto incluye métodos como .click(), .focus(), .blur(), .change(), entre otros. Estos métodos se destacan por su simplicidad y reciben una función de **callback** que se ejecutará al ocurrir el evento.

Por ejemplo, al utilizar $('div.noticia').click(masInformacion);, cada vez que se haga clic en un div con la clase "noticia", se activará la función masInformacion(). También podemos emplear una **función anónima** de la siguiente manera:

```
$('div.noticia').click(function() {
  // 'this' se refiere al objeto DOM que disparó el evento
});
```

En cuanto al modelo de eventos, en JQuery se mantiene la validez de los conceptos aprendidos en JavaScript, con la salvedad de que siempre se aplica el **modelo de burbujeo** y los eventos son compatibles con todos los navegadores (Cross-Browser), eliminando la necesidad de verificar versiones específicas. Esto significa que el objeto evento tiene propiedades que funcionan uniformemente en todos los navegadores. Asimismo, dentro de las funciones de eventos, podemos usar **this** para referirnos al objeto que provocó el evento, y la función recibe automáticamente un parámetro representando el evento.

Cabe destacar que, a pesar de contar con métodos específicos para eventos comunes como **click, focus, blur, load o change**, se aconseja emplear una notación más **general** utilizando el método **bind**. El método bind ofrece la conveniencia de asignar **múltiples eventos** a un mismo objeto en una sola línea de código, así como la capacidad de enviar información a la función que maneja el evento. Esta función acepta tres parámetros: el primero es el evento, el segundo es un array JSON con propiedades y valores, y el tercero es la función manejadora del evento. Por ejemplo, si queremos asignar varios eventos al mismo elemento:

```
$('#lienzo').bind({
 'click': function(e) { alert('Usted hizo click en ' + e.pageX + "," + e.pageY)},
 'mouseover': function() { alert('El ratón está sobre el lienzo")},
 'mouseout': function() { alert('El ratón ha salido del lienzo)}
})
```

Un ejemplo de cómo pasar parámetros:

```
// Definimos la función manejadora del evento con parámetros
function handleClick(event, mensaje) {
 alert("Se hizo clic en el botón. Mensaje adicional: " + mensaje);
}

// Asignamos el evento de clic al botón y pasamos parámetros usando bind
$('#miBoton').bind('click', { mensaje: 'Hola desde el evento clic' }, handleClick);
```

En este ejemplo, al hacer clic en el botón con el id "miBoton", se ejecutará la función handleClick. La función bind se utiliza para asignar el evento de clic y también para pasar un objeto con datos adicionales ({ mensaje: 'Hola desde el evento clic' }) que estarán disponibles en la función manejadora a través del objeto event.data.

Adicionalmente, bind permite desactivar eventos en cualquier momento mediante **unbind**. Podemos desactivar un evento específico o todos los eventos asociados a un objeto:

```
$('input').unbind("click", imprimirBalance); // Desactivamos solo este listener
$('input').unbind("click"); // Desactivamos TODOS los listener para el evento click
```

Además, JQuery cuenta con un mecanismo interesante llamado **one()**. Este método habilita un evento de manera que solo puede dispararse una única vez. En las subsiguientes ocasiones, el evento no se activará. Es útil cuando se desea asegurar que un evento solo ocurra una vez durante la interacción con un elemento.

Por otro lado, en las últimas actualizaciones de JQuery el método bind() fue mejorado para incorporar el método **on()**. Este método on() recibe los mismos parámetros que en el caso de bind() pero tiene la ventaja de permitir lo que se denomina **delegación de eventos**, método por el cual asignamos un evento a un contenedor padre y éste asignará dichos eventos a los elementos hijos, de modo que cuando se produzca el mismo en cada un de ellos se ejecute. De este modo, se simplifica aún más el código. Por ejemplo, si quisiéramos asignar la misma función a una serie de botones podríamos hacerlo uno a uno como:

```
$('button').on('click', function() {
// DO ACTION
});
```

Sin embargo, el método on() nos da la facilidad de asignar dicho evento en un contenedor padre como:

```
$('.container').on('click', 'button', function() {
// DO ACTION
});
```

Cosa que con el método bind() no estaba disponible. Finalmente, para eliminar eventos delegados en JQuery utilizando el método on(), se utiliza el método **off()** cuya sintaxis es análoga a la del método on().

10. Efectos y animaciones en JQuery

Con jQuery, agregar efectos a una página es muy fácil. Estos efectos poseen una **configuración predeterminada** pero también es posible proveerles parámetros personalizados. Además es posible crear animaciones personalizadas estableciendo valores de propiedades CSS. Algunos de los efectos predefinidos más habituales son los siguientes:

Método	Descripción
show()	Muestra el elemento seleccionado.
hide()	Oculta el elemento seleccionado.
fadeIn()	De forma animada, cambia la opacidad del elemento seleccionado al 100%.
fadeOut()	De forma animada, cambia la opacidad del elemento seleccionado al 0.
slideDown()	Muestra el elemento seleccionado con un movimiento de deslizamiento vertical.
slideUp()	Oculta el elemento seleccionado con un movimiento de deslizamiento vertical.
slideToggle()	Muestra o oculta el elemento seleccionado con un movimiento de deslizamiento vertical, dependiendo de su visibilidad actual.

Por ejemplo, para asignar un efecto basta con aplicarlo al selector JQuery:

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

Incluso, se puede configurar la duración de esos efectos. A excepción de show() y hide(), todos los métodos tienen una duración predeterminada de la animación en **400ms**. Dicho valor se puede cambiar asignando la duración en el método como un valor numérico (en ms) o mediante “slow”, “normal” o “fast”:

```
$('h1').fadeIn(300);     // desvanecimiento en 300ms
$('h1').fadeOut('slow');  // utilizar una definición de velocidad interna
```

Asimismo, disponemos de animaciones **personalizadas** mediante el uso del método **animate()**. Dicho método permite realizar una animación estableciendo valores a propiedades CSS o cambiando sus valores actuales. Su sintaxis es la siguiente:

```
$(selector).animate({props}, duración, [easing], [callback]);
```

- **props:** Un objeto que define las propiedades CSS y los valores a los que se animarán.
- **duración**: La duración de la animación en milisegundos.
- **easing (opcional):** La función de aceleración que especifica cómo se debe realizar la animación, es decir, permite definir si la velocidad de animación es constante o no.
- **callback (opcional):** Una función que se ejecutará una vez que la animación haya terminado.

Por ejemplo:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>jQuery Animate Example</title>
 <script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>
 <style>
  #animated-box {
    width: 100px;
    height: 100px;
    background-color: blue;
    opacity: 1;
  }
 </style>
</head>
```

```
<body>

<div id="animated-box"></div>

<script>
 $(document).ready(function() {
  // Seleccionar el elemento con el id "animated-box"
  let $box = $('#animated-box');

  // Animar el cambio de ancho y la opacidad
  $box.animate({
   width: '200px',
   opacity: 0.5
  }, 1000, 'swing', function() {
   // Callback: Se ejecuta después de completar la animación
   console.log('Animación completada');
  });
 });
</script>

</body>
</html>
```

Por último, JQuery provee varias herramientas para el manejo de animaciones tales como el método stop(), el cual Detiene las animaciones que se están ejecutando en el elemento seleccionado, y delay(), que espera un tiempo determinado antes de ejecutar la próxima animación. Por ejemplo:

```
$('h1').show(300).delay(1000).hide(300);
```

11. Bibliografía y Webgrafía

[1] A. Garro, “JavaScript”, *Arkaitzgarro.com*, 01-ago-2014. [En línea]. Disponible en: https://www.arkaitzgarro.com/javascript/index.html [Consultado: 12-nov-2023].

[2] “Introducción a JavaScript”, Uniwebsidad.com. [En línea]. Disponible en: https://uniwebsidad.com/libros/javascript [Consultado: 14-nov-2023].

[3] A. Garro, “jQuery”, Arkaitzgarro.com, 14-ene-2014. [En línea]. Disponible en: https://www.arkaitzgarro.com/jquery/index.html [Consultado: 14-nov-2023].