MEDAC
Instituto Oficial de Formación Profesional

DISEÑO DE INTERFACES WEB

TÉCNICO EN DESARROLLO DE APLICACIONES WEB

Integración de contenido interactivo

09

ÍNDICE

OBJETIVOS

Identificar las tecnologías asociadas con la inclusión de contenidos multimedia e interactivo.

Configuración de navegadores.

Elementos interactivos básicos y avanzados.

Comportamientos interactivos de los elementos.

Modificación de los comportamientos.

Cambio de las propiedades de un elemento.

Ejecución de secuencias de comandos.

/ 1. Introducción y contextualización práctica

Los elementos interactivos permiten dotar a un sitio web de un comportamiento "más activo", lo que suele proporcionar una mejora en la experiencia del usuario dentro del sitio. Ahora bien, no conviene excedernos con la cantidad de elementos interactivos, puesto que se puede conseguir el efecto contrario, recargando tanto la página web que el usuario quiera salir de ella y no regresar nunca.

En este capítulo se trabajará con una librería muy importante jQuery, la cual permite la implementación de comportamiento interactivo a través del acceso a los distintos elementos y espacios definidos en el DOM. Se trata de una librería de JavaScript.

Nos detendremos en el tratamiento de eventos que realiza esta librería, así como en algunas de sus funciones más importantes como *CSS() o ReplaceWith()*.

En el audio siguiente se propone una reflexión sobre la adecuación del tipo de reproducción de un audio o video en función del tipo de sitio web.

Al final de la unidad, encontrarás su resolución.

Fig. 1. Imagen de jQuery

Audio Intro. "JavaScript y jQuery"
https://bit.ly/35vhHiM

/ 2. Comportamiento interactivo con jQuery

Cómo se comenzó a ver en el capítulo anterior, JavaScript a través del uso de su **librería jQuery** permite la implementación de cierto comportamiento interactivo sobre las interfaces web. Esta librería de código abierto se basa en **facilitar la realización de ciertas tareas como el acceso al DOM, la manipulación de las hojas de estilo CSS o la inclusión de elementos que proporcionan interactividad a una interfaz web. De ahí que su eslogan sea *"Write less, do more"*.**

Para su descarga y puesta en funcionamiento, en primer lugar, es necesario **configurar el entorno** para poder utilizar el contenido de la librería jQuery. Encontramos dos opciones:

1. **Acceso local.** En este caso, en primer lugar, se realiza la descarga de la librería desde el sitio web (enlace), se guarda en una ruta concreta y finalmente, se añaden las siguientes etiquetas en el fichero HTML. El atributo ***src*** se utiliza para indicar la ruta en la que ha sigo almacenado.

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

Código 1. Sintaxis inclusión jquery.js acceso local

2. **Acceso desde CND.** La segunda opción consiste en utilizar un CND *(Red de Distribución de Contenidos)*. Se trata de realizar la carga de la librería de forma directa apuntando a alguno de los repositorios que permiten el acceso a estas librerías, la desventaja de esta forma es que solo funciona si hay conexión a Internet.

```
<script
src="http://ajax.googleapis.com/ajax/libs/
jquery/1.8.3/jquery.min.js"
type="text/javascript">
```

Código 2. Sintaxis inclusión jquery.js acceso desde CND

Tras la inclusión de estas etiquetas, de la forma que se considere más conveniente en cada diseño, será posible realizar la implementación de cualquier tipo de función en JavaScript usando la librería jQuery. Por ejemplo, de la siguiente forma:

```
<script
src="https://ajax.googleapis.com/ajax/libs/
jquery/3.2.1/jquery.min.js"></script>
<script>
 $(document).ready(function(){
   $("#b1").click(function() {
     ...
   }
 }
 $("b1").click(function() { alert( 'Hola
 jQuery!');});
</script>
```

Código 3. Sintaxis uso de funciones jQuery

2.1. Revisión del DOM y ejecución del código jQuery.

En primer lugar, conviene recordar la estructura principal de una página HTML, esto es, de aquellos elementos que definen una interfaz, puesto que jQuery se basa en el acceso y manejo de estos elementos.

Una de las características que se requieren para poder actuar de forma dinámica sobre los elementos de una página web es el **DOM** *(Document Objet Model),* que permite acceder a los diferentes elementos de una página web,

Este modelo permite diferenciar entre **varios tipos de elementos denominados nodos bajo el modelo DOM.**

- ***Document.*** Se trata del nodo raíz del que derivan el resto.
- ***Element.*** Representa las etiquetas utilizadas en HTML, puede contener atributos, los cuales coinciden con los utilizados en cada etiqueta.
- ***Attr.*** Representa los atributos de las etiquetas HTML.
- ***Text.*** Representa el texto que se encuentra entre los delimitados de una etiqueta HTML.
- ***Comment.*** Se utilizan para representar los comentarios en una página HTML.

Como se ha descrito, este modelo utiliza un sistema de nodos que quedan identificados de diferentes formas, por ejemplo, a través del nombre del elemento, de un identificador, de un atributo...

Para la implementación del comportamiento interactivo utilizando el **código jQuery,** es necesario utilizar una instrucción que permite el acceso al DOM que se produzca la ejecución del código contenido se produzca ante la ocurrencia de un determinado evento.

```
$(document).ready(function(){. . .}
```

Código 4. Sintaxis ejecución de función

El ámbito de aplicación de una acción se realiza seleccionando un elemento concreto del DOM, para ello se utilizará la **función $(),** la cual nos permite poner el foco en practicasen cualquier elemento del DOM.

```
$selector.accion()
```

Código 5. Sintaxis de función $()

La acción permite determinar qué es lo que se va a realizar sobre el elemento seleccionado. Es posible llevar a cabo esta selección de múltiples formas distintas, algunas de las más utilizadas son:

- La selección de elementos con **identificador:** $("#ID").
- La selección de elementos con **etiqueta:** $("p").
- Atendiendo a su **clase:** $(".nombreClase").
- La selección de **atributos** de un elemento: $("a[rel]").

2.2. Funciones DOM.

A continuación, veremos las funciones que permiten acceder a cada uno de estos elementos para consultar o modificar su contenido

a. **Extracción de valores:**

- ***getElementsByTagName* (Etiqueta):** Obtiene todos los elementos de un documento HTML cuya etiqueta sea indicada por parámetro. Esta función se aplica al elemento "document", es decir, a todo el documento HTML.

```
var elemP = document.getElementsByTagName("p");
```

Codigo 6. Función acceso por etiqueta a elemento

- ***getElementsByName* (Etiqueta):** Obtiene todos los elementos de un documento HTML cuyo nombre (atributo name) coincida con el que se indica por parámetro. Esta función se aplica al elemento *"document",* es decir, a todo el documento HTML.

```
var elemName = document.getElementsByName("prueba");
```

Codigo 7. Función acceso por atributo a elemento

- ***getElementById* (Etiqueta):** Obtiene todos los elementos de un documento HTML cuyo identificador coincida con el que se pasa por parámetro. Esta función se aplica al elemento "document", es decir, a todo el documento HTML.

```
var elemName = document.getElementById("id");
```

Codigo 8. Función acceso por id a elemento

b. **Inserción de elementos**

Además del acceso a los nodos ya creados y la modificación de su valor, es posible crear nuevo contenido utilizando las funciones dedicadas a tal fin:

- ***document.createElement (Etiqueta):*** Crea un elemento del tipo especificado por parámetro, es decir, del tipo de etiqueta indicado.
- ***document.createTextNode ("cadena de texto"):*** Crea un elemento de texto con el contenido especificado por parámetro.
- ***element.appendChild (elemento_hijo):*** Esta función realiza la inserción de un nuevo elemento "elemento_hijo" que dependerá del elemento *"element"* sobre el que se define acción. Para poder utilizar esta función se debe haber creado previamente el elemento del que depende el elemento hijo pasado por parámetro.

Para concluir la inclusión de un nuevo nodo, será necesario añadirlo al resto de la página, hasta ahora teníamos el nodo creado, pero no estaba insertado como tal en el documento HTML.

/ 3. Caso práctico 1: “Diseña un fichero HTML con JavaScript y jQuery”

Planteamiento: En este primer caso práctico se pide diseñar el código HTML necesario para importar de forma adecuada la librería jQuery que nos permitirá usar sus funciones. Además, los elementos que debe incluir se detallan a continuación:

- Etiqueta de texto.
- Botón.

Cuando el botón sea pulsado aparecerá en la etiqueta de texto un mensaje cualquiera.

```
<!DOCTYPE html >
<html><head>
<meta charset=”utf-8”>
  <script
  src=””https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js”>
  <script type=”text/javascript”>
    $(document).ready(function () {
      var texto = “Hola mundo”;
    $(“#botonSaludar”).click(function () {
      $(“#mensajeSalida”).append(texto);
    });
  });
  </script>
</head>
<body>
<div id=”mensajeSalida”></div>
<div id=”botonSaludar”> SALÚDAME </div>
</body>
</html>
```

Código 9. Código Caso práctico 1

Nudo: La realización de este ejercicio nos ayudará a comprender el funcionamiento de la librería jQuery. Además, se les está dotando de comportamiento interactivo al sitio web. Si cuando se detecte un evento de ratón “pulsación del botón” la página web nos devuelve un mensaje, podemos decir que tenemos una comunicación, por lo que ya no estamos ante una página estática e invariable.

Desenlace: En el siguiente código se añade tanto el código HTML que modela los elementos de la página web, como el código desarrollado en jQuery y JavaScript, contenido en head y entre las etiquetas script.

/ 4. Eventos con jQuery: Teclado

Los eventos permiten que mediante la interacción de una determinada acción del usuario sobre los elementos del sitio web se produzcan ciertos cambios en la interfaz de usuario de una página web. Ahora bien, jQuery también incorpora un conjunto de métodos que permiten la detección de eventos.

Como hemos visto en el apartado anterior, la sintaxis de uso se basa en la detección de una acción aplicada sobre un determinado elemento.

```
$(elemento).evento(accion)
```

Código 10. Sintaxis detección de eventos

En este caso, cuándo se produzca el **evento** sobre el **elemento** indicado, se llevará a cabo la acción implementada entre paréntesis.

Los eventos de teclado ocurren **cuando se detecta algún tipo de acción producida por el uso de las teclas de un teclado.** Este tipo de eventos no suelen estar asociados a un elemento concreto, sino a todo el documento, identificado con el nodo denominado **document.**

```
$(document).keyXXX(funcion () {

 ...

}
```

Código 11. Sintaxis detección evento de teclado

FUNCIÓN	DESCRIPCIÓN
keypress()	Este evento se produce mientras está pulsada una tecla. Se producirá tantas veces como tiempo se mantenga pulsada.
keyup()	Este evento se produce cuando se deja pulsar una tecla.
keydown()	Este evento se produce cuando se pulsa una tecla, a diferencia de keypress solo se detecta como evento la pulsación no mantenerla pulsada.

Tabla 1. Funciones eventos de teclado

En el siguiente fragmento de código se detecta cuando una tecla ha sido pulsada, para ello r a través de la función función *keydown()* se detecta la pulsación.

```
$(document).ready(function(){
  $("#tecla").keydown(function(event){
    $("#salida").text("La tecla pulsada es: " + event.which);
    $("#tecla").val("");
  });
});
```

Código 12. Ejemplo uso keydown

/ 5. Eventos con jQuery: Ventana

Este tipo de eventos ocurren **cuando se detecta algún tipo cambio sobre la ventana en la que se visualiza la interfaz.** Estos eventos quedan vinculados al documento y no un elemento concreto.

FUNCIÓN	DESCRIPCIÓN
load()	Este evento se produce cuando quedan cargados todos los elementos de una página web.
unload()	Este evento se produce cuando se realiza cualquier acción que implica "salir" de la página web en la que nos encontrábamos (cerrar la ventana, cambiar de página web a través de un vínculo...).
scroll()	Este evento se produce cuando existe un desplazamiento por la ventana de la página web utilizando al barra de desplazamiento, el scroll.
resize()	Este evento se produce cuando se redimensiona el tamaño de una ventana.

Tabla 2. Funciones eventos de ventana

Veamos algunos ejemplos de uso del tratamiento de eventos de ventana. En este código cuando se realiza *scroll* sobre un elemento concreto llamado "cajaAuxiliar" se incrementa un contador que posteriormente se muestra en la página web.

```
 var $contador = 0;
$( "#cajaAuxiliar" ).scroll(function() {
  $contadorScroll++;
  $("#numeroScroll").prepend( "<div>Scroll realizado:
  "+$contadorScroll+"</div>" );
});
```

Código 13. Ejemplo uso scroll

En este otro fragmento de código cuando se redimensiona el tamaño de la ventana, se detecta como la ocurrencia de un evento. Al igual que en el caso anterior se imprime en la pantalla un valor, concretamente el tamaño del ancho de la pantalla al que se accede a través de la función *width()*.

```
$( window ).resize(function() {
  $( "body" ).prepend( "<div>" + $( window ).width() + "</div>" );
});
```

Código 14. Ejemplo uso resize

Vídeo 1. "Análisis uso jQuery"
https://bit.ly/3kvQ67p

/ 6. Eventos con jQuery: Ratón

Es tipo de eventos ocurren **cuando se detecta algún tipo acción producida por el puntero del ratón.**

FUNCIÓN	DESCRIPCIÓN
click()	Este evento se produce cuando se pulsa una vez sobre el elemento con el ratón.
dblclick()	Este evento se produce cuando se pulsa dos veces seguidas sobre el elemento con el ratón.
hover()	Este evento se produce cuando el puntero del ratón pasa sobre el elemento indicado.
mousedown()	Este evento se produce cuando se realiza una pulsación sobre el elemento. La diferencia entre este evento y click(), es que en este caso no es necesario "soltar" la pulsación para que el evento sea detectado.
mouseup()	Este evento se produce si se suelta el botón del ratón tras realizar la pulsación sobre el elemento.
mouseleave()	Este evento se produce cuando el puntero sale de encima del elemento.
mouseenter()	Este evento se produce cuando el puntero se coloca justo encima del elemento.

Tabla 3. Funciones eventos de ratón

En el siguiente fragmento de código se detecta tanto cuando se presiona el botón del ratón, como cuando este es soltado.

```
$(document).ready(function(){
  $("#botonPulsado").mouseup(function(){
    $('#salida').text("Has soltado el botón del ratón");
  });
  $("#botonPulsado").mousedown(function(){
    $('#mensaje').text("Has pulsado el botón del ratón");
  });
});
```

Código 15. Ejemplo uso mouseup y mousedown

/ 7. Funciones interactivas con jQuery: replaceWith()

Esta función **permite que una imagen se modifique por otra ante la ocurrencia de un determinado evento,** por ejemplo, la pulsación sobre un botón.

```
selector.replaceWith(nueva imagen)
```

Código 16. Sintaxis función ReplaceWith() jquery.js

En primer lugar, se indica el elemento sobre el que se va a producir el cambio, es decir, aquel bloque en el que esté insertada la imagen inicial. Entre paréntesis se coloca el nuevo código que va a sustituir al que define la inserción de la imagen inicial. Lo vemos en el siguiente ejemplo:

Se define un elemento de tipo imagen que quedará identificado con el nombre **"imagenInicial"** y cuyo contenido queda apuntado por el atributo ***src.***

```
<button id="botonCambiar"> PULSAR PARA CAMBIAR </button>
<img id="imagen" src="imagenInicial.png" width="500"
height=150">
```

Código 17. Definición de botón e imagen HTML para ejemplo Código 17

Ahora bien, para realizar el cambio de imagen utilizando la función ***replaceWith(),*** en primer lugar, se utiliza la instrucción principal para el uso de jQuery ***($(document).ready(function() {}.*** Será dentro de estas llaves donde se implemente el resto del código.

En este caso vamos a hacer que al pulsar un botón se realiza la llamada a una función que produce el cambio.

```
<script>
 $(document).ready(function() {
   $('#botonCambiar').click(function(){
     $('#imagen').replaceWith('<img id="imagen" src="imagenFinal.
     png" width="1000" height="500">');
   });
 });
</script>
```

Código 18. Sintaxis ejemplo uso de replace With

Audio 1. "Ampliación de funciones jQuery. Función each()"
https://bit.ly/3ot9ynH

/ 8. Funciones interactivas con jQuery: css()

Finalmente, resulta conveniente conocer la función css() de jQuery. Esta función **permite acceder y/o modificar las propiedades implementadas para un determinado evento.**

Para extraer el valor de una determinada propiedad de un elemento concreto se utiliza la sintaxis que se muestra a continuación:

```
selector.css('nombre propiedad')
```

Código 19. Sintaxis función css()

Por ejemplo, en el siguiente fragmento de código se almacena en una variable las dimensiones de un bloque identificado con el id *clase1*.

```
var dim=$('#clase1'.css('width')
```

Código 20. Extracción y almacenamiento valor propiedad with con css()

La función *css()* también permite modificar el valor de las propiedades definidas en la hoja de estilo, para ello esta función recibe por parámetro el nombre de la propiedad que va a modificar su aspecto y también el nuevo valor para esta propiedad. Al igual que en el caso anterior, se aplica sobre un elemento concreto.

```
selector.css('nombre propiedad','nuevo valor propiedad')
```

Código 21. Sintaxis 2 función css()

Por último, otras de las ventajas del uso de esta función es que permite modificar el valor de varias propiedades al mismo tiempo, siempre sobre el mismo elemento.

```
selector.css({
    'propiedad 1' : 'nuevo valor propiedad 1',
    'propiedad 2' : 'nuevo valor propiedad 2',
  ...,
  'propiedad N' : 'nuevo valor propiedad N',
});
```

Código 22. Sintaxis múltiple de Css()

Por ejemplo, en el siguiente fragmento de código se modifica el valor de las tres propiedades indicadas sobre un bloque identificado de nuevo como clase1.

```
$('#clase1').css({
  'width' : '50%',
  'background-color' : '#111111',
  'height' : '100px',
});
```

Código 23. Ejemplo de uso de css()

Vídeo 2. "Análisis uso de funciones con jQuery"
https://bit.ly/35PWOPF

/ 9. Caso práctico 2: "Detecta la tecla pulsada con jQuery"

Planteamiento: Se pide crear una aplicación que permita detectar qué tecla ha sido pulsada. Para ello se implementa una caja de texto sobre la que se situará el foco y se implementa el evento de escucha de teclado.

Además de mostrarse en la caja de texto se devolverá un mensaje con el valor de la letra seleccionada en cada caso, algo así ocurre cuando se introduce una contraseña y en la caja de texto se colocan * y por unos instantes, en otra capa, se muestra la tecla exacta seleccionada.

Nudo: Para identificar qué tecla ha sido pulsada se utiliza la propiedad *which* del objeto evento de jQuery, esta propiedad puede ser invocada siempre y cuando se haya implementado una escucha de evento de teclado, el valor devuelto es un número entero con el código Unicode de la tecla pulsada.

Desenlace: En el siguiente código se añade tanto el código HTML que modela los elementos de la página web, como el código desarrollado en jQuery y JavaScript, contenido en *head* y entre las etiquetas *script*.

```
<html>
<head>
  <title>...</title>
  <script src="../jquery-1.4.1.min.js"></script>
  <script>
    $(document).ready(function(){
      $("#cajaTexto").keypress(function(e){
        e.preventDefault();
        $("#teclaPulsada").html(e.which + ": " + String.
        fromCharCode(e.which))
      });
    })
  </script>
</head>
<body>
  <h1>Detección teclas</h1>
  <form>
    <textarea cols=220 rows=2 id="cajaTexto">Introduce una cadena
    de texto</textarea>
    <br>
    <b>Se ha pulsado la tecla</b>
    <div id="teclaPulsada"></div>
  </form>
</body>
</html>
```

Código 24. Código Caso Práctico 2

/ 10. Resumen y resolución del caso práctico de la unidad

En este capítulo hemos trabajado sobre el **comportamiento interactivo** a través de la librería **jQuery,** esta librería de código abierto se basa en facilitar la realización de ciertas tareas como el acceso al DOM, la manipulación de las hojas de estilo CSS o la inclusión de elementos que proporcionan interactividad a una interfaz web.

Para la implementación del comportamiento interactivo utilizando el código jQuery, es necesario utilizar una instrucción que permite el acceso al **DOM.**

El ámbito de aplicación de una acción se realiza seleccionando un elemento concreto del DOM, para ello se utilizará la **función $(),** la cual nos permite poner el foco en practicasen cualquier elemento del DOM.

Otro de los elementos importantes que se han revisado en este capítulo, son los **eventos** puesto que permiten que ante la interacción del usuario con un determinado elemento se produzca una acción que tiene como consecuencia ciertos cambios en la interfaz de usuario de una página web.

Finalmente, para concluir se han revisado algunas funciones muy útiles como ***Css()*** o ***replaceWith(),*** las cuáles pueden agilizar ciertos aspectos de desarrollo.

Resolución del caso práctico inicial

Como se ha visto a lo largo del tema, el uso de jQuery para la implementación y desarrollo de funciones que permiten dotar de interacción a un sitio web es clave cuando hablamos de comportamiento interactivo.

Como respuesta a las preguntas planteadas al inicio de este capítulo, podemos concluir que jQuery y JavaScript son "lo mismo" en cuanto a que jQuery no es más que un conjunto de librerías JavaScript que permiten el acceso de una forma ágil a todos los elementos de una página web, utilizando para ello el conocido DOM.

Fig. 2. Tecnologías desarrollo web

Por otro lado, JavaScript es en sí un lenguaje de programación que puede ser utilizado en un navegador web, por ejemplo, en la inclusión de imágenes, creación de funciones, interacciones con AJAX para el modelado de los sistemas petición-respuestas con un servidor...

Es decir, será la combinación de JavaScript y jQuery en el marco de un sitio HTML y CSS, lo que permiten modelar casi cualquier tipo de comportamiento interactivo en un sitio web.

/ 11. Bibliografía

García-Miguel, D. (2019). *Diseño de Interfaces Web (1.a ed.).* Madrid, España: Síntesis.

Gauchat, J. (2017). *El gran libro de HTML5, CSS3 y JavaScript (3.a ed.).* Barcelona, España: Marcombo.