Desarrollo Web En Entorno Cliente

TEMA 5

Interacción con el usuario y gestión de eventos

Daniel López Lozano

Escuela Arte Granada

Índice

- Acceso al HTML desde JavaScript.
- Manejo de eventos en JavaScript.
- Introducción a Jquery.
- Validación básica de formularios.

HTML desde JavaScript

Daniel López Lozano

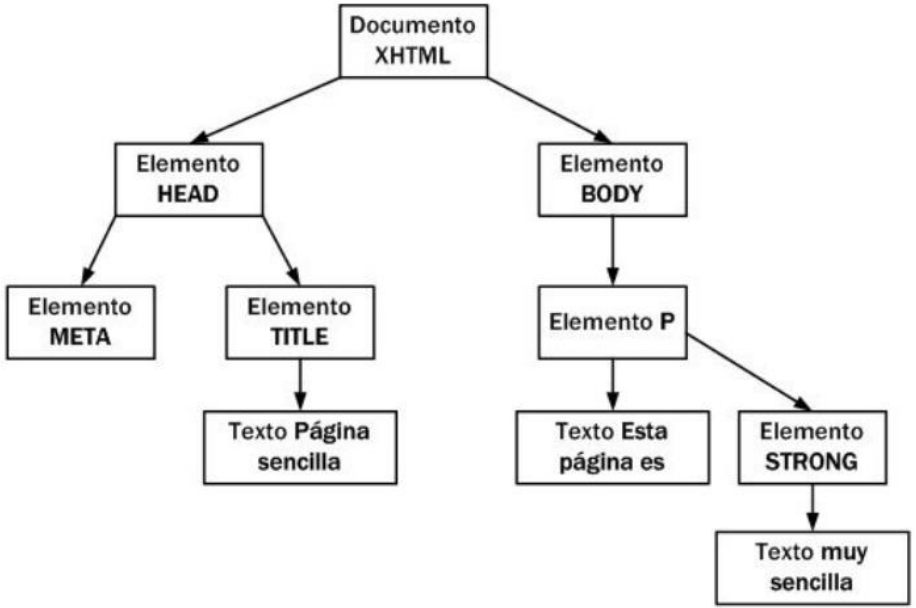
Escuela Arte Granada

- El DOM es la herramienta que más ha influido en el desarrollo de páginas webs dinámicas y aplicaciones web complejas.
- Hasta ahora, tan solo podíamos añadir nuevos elementos al comienzo de la página.
- Eso ensuciaba y complicaba en exceso el código.
- El DOM permite acceder y manipular páginas HTML.
- De este modo podemos modificar o eliminar una parte concreta de la Web o añadir nuevos elementos.

- Una de las tareas que más se realizan con JS es la de manipular la apariencia de una página Web, es decir, recuperar y modificar el valor de los atributos CSS o HTML.
- Es decir, el DOM es un estándar para obtener, modificar, añadir, borrar elementos de un documento HTML.
- El DOM transforma un documento HTML en una serie de nodos interconectados que representan el contenido de la página y sus relaciones. Estos nodos se organizan como un árbol formando el llamado árbol de nodos.

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
        <meta harset=UTF-8" />
        <title>Página sencilla</title>
</head>
<body>
<p>Esta página es <strong>muy sencilla</strong></p>
</body>
</html>
```



- La especificación completa de DOM define 12 tipos de nodos, aunque nosotros vamos a trabajar solamente con unos pocos:
- Document: nodo raíz que derivan todos los demás nodos del árbol.
- Element: representa cada una de las etiquetas HTML. Este es el único tipo de nodo que contiene atributos y del que pueden derivar otros nodos.
- Attr: representa cada uno de los atributos de las etiquetas HTML, es decir, uno por cada atributo=valor.

- Text: nodo que contiene el texto de una etiqueta HTML.
- Las páginas HTML producen árboles con miles de nodos.
- El DOM proporciona una serie de funciones que permiten acceder a cualquier nodo de la página de forma sencilla e inmediata.
- En las ultimas revisiones de JavaScript se ha evolucionado de manera que solamente son necesarias dos.
- querySelector y querySelectorAll.

- Para poder acceder a los nodos, la página debe haber sido cargada por completo, para que el árbol de nodos se haya construido.
- Una de las opciones para asegurarse de que el navegador ha cargado, es poner el script después de la etiqueta body (no siempre funciona).
- Pero la opción más recomendable es utilizar el evento onload.

```
window.onload=Inicio;

function Inicio()
{
    //Codigo para acceder a los elementos de la página
}
```

- Las funciones para acceder directamente a un nodo o nodos son:
- querySelector() // Busca usando selectores CSS un solo elemento de la página. Devuelve un objeto de tipo Element.
- querySelectorAll() //Busca usando selectores CSS todos los elementos que coincidan con ese selector. Devuelve un Array de tipo Element, aunque solo coincida un solo elemento.
- Son funciones del objeto **document**.

- El código siguiente guarda en la variable enlaces todos los nodos de la página que sean enlaces.

```
var enlaces = document.querySelectorAll("a");
var primer= enlaces[0]; // guarda el primer enlace de la página
for(i=0; i<enlaces.length; i++)
{
    alert(enlaces[i].href+"<br>"); //muestra el href de cada enlace
}
```

- La función querySelector también funcionaria, pero solo devolvería solo el primer enlace.

- **EJERCICIO 1:** Realizar un script para la página pagina.html que muestre el número de párrafos de la página y el número de enlaces de la página. Después mostrará a donde apunta el primer y el último enlace.
- El funcionamiento del script será el siguiente: Una vez se haya cargado la página por completo se mostrarán mediante alert los siguientes 3 mensajes:

Resumen: X párrafos y X enlaces

El primer enlace apunta a: enlace

El último enlace apunta a: enlace

- Podemos seleccionar por id.
- Esto lleva implícito que solo debe haber un elemento con un id.
- Para ello usamos querySelector() que es el indicado cuando solo queremos seleccionar un elemento.

```
var cabecera = document.querySelector("#cabecera");
alert(cabacera);
```

- **EJERCICIO 2:** Realizar un script para pagina.html muestre las etiquetas con los siguientes id:
- **buscador**
- **noticias**
- **menu_principal**
- **secundario**
- **pie**
- ¿Cuál de ellos esta aplicada una etiqueta distinta?

- Si queremos seleccionar elementos de una clase debemos usar **querySelectAll().**
- **EJERCICIO 3:** Realizar un script para la página pagina.html que muestre cuántos elementos tienen aplicada cada una de las siguientes clases y de qué tipo de etiqueta son:
- **clear**
- **fecha**
- **artículo**

- Uno de los atributos del objeto Element es innerHTML.
- Este atributo es muy potente y hace referencia a todo el contenido de un elemento incluido el código que pueda contener.
- Podemos consultar, modificar, editar o borrar el contenido de cualquier etiqueta HTML de la página.

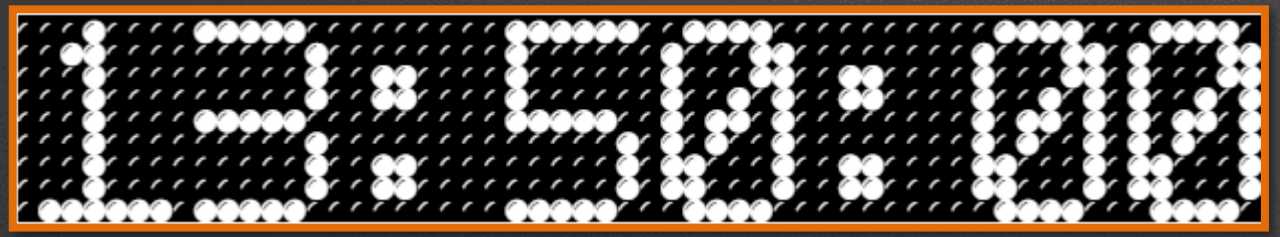
```
var parrafo=document.querySelector("titular"); //selecciona el elemento
alert(parrafo.innerHTML); //muestra Noticia <strong> Destacada</strong>

parrafo.innerHTML=parrafo.innerHTML + "del día";
alert(parrafo.innerHTML);// muestra Noticia <strong> Destacada</strong> del día

parrafo.innerHTML= "Nueva Noticia" //borra el contenido y guarda Nueva Noticia
alert(parrafo.innerHTML);
```

- Otros selectores posibles:

```
var enlaces=document.querySelectorAll("#principal a");

var lista=document.querySelectorAll("header .cursiva");

var tabla=document.querySelectorAll("table .subrayado");

var articulos=document.querySelectorAll("div.articulo");

var h2_derecha=document.querySelectorAll("h2[align='right']");

var etiquetasvarias=document.querySelectorAll("p,h1,span");

var grandeImportante=document.querySelectorAll(".grande.importante");
```

- **EJERCICIO 4:** a partir del fichero pagina.html mostrar en un solo alert la siguiente información:
- - **Nº de enlaces de la página**
- **- Nº Enlaces dentro del id noticias**
- **- Nº Elementos con atributo alt en la página**
- **- Nº Imágenes de la página**
- **- Nº Capas con la clase artículo**
- **- Nº Enlaces dentro de clases artículo**

- **EJERCICIO 5:** Usando el objeto Date, setInterval y con DOM crear una web (dentro de un div) que muestre un reloj digital y que se actualice cada segundo.



- Tras acceder a un nodo, el siguiente paso es acceder y/o modificar sus atributos y propiedades.
- Mediante DOM, podemos acceder a todos los atributos HTML y propiedades CSS de la página.
- Los atributos HTML se transforman automáticamente en propiedades de los nodos.
- Para acceder a ellos, simplemente se indica el nombre del atributo HTML.
- El atributo class es especial ya que cambia su nombre por className

- document.title hace referencia al título de la página, es decir permite consultar o modificar el atributo title de la página
- Existen métodos para consultar modificar y borrar atributos. getElement, setAttribute y removeAttribute
- Aunque podemos usar ambas formas nosotros utilizaremos métodos y no referirnos a los atributos por nombre directamente.
- Exceptuando innerHTML que es de uso muy común

- La principal ventaja que tiene el uso de los métodos es que podemos definir nuevos atributos a las etiquetas.

```
var enlace=document.querySelector("#principal"); //recuperamos el enlace con id principal

alert(enlace.href); //muestra el vínculo del enlace
alert(enlace.getAttribute("href")); // muestra el vínculo del enlace

enlace.href="www.google.es" //cambia el vínculo del enlace
enlace.setAttribute("href", "www.google.es") //cambia el vínculo del enlace

enlace.className=""; //borra las clases del enlace
enlace.removeAttribute("class") //borra las clases del enlace

enlace.nivel=2; //este daria error
enlace.setAttribute("nivel",2);//este si funcionaria
```

- **EJERCICIO 6:** Crea un script que modifique pagina.html de forma que se modifiquen el segundo, tercer y cuarto enlaces del con id noticias para que estos apunten respectivamente a:
- www.google.es
- www.escuelaartegranada.com
- www.w3schools.com

- Las propiedades CSS no son tan fáciles de obtener como los atributos HTML.
- Para obtener el valor una propiedad CSS, se debe utilizar el atributo style.
- Para acceder a estas propiedades, se elimina el guión y se cambia a mayúsculas la inicial de cada palabra.
- margin-top //valor css
- marginTop // nombre en el DOM

```
var capa = document.querySelector("#secundario");
capa.style.backgroundColor="#F0C0C0";
capa.style.borderTopWidth="20px";
alert(capa.style.backgroundColor);
alert(capa.style.borderTopWidth);
```

- **EJERCICIO 6:** Modifica los encabezados de los div artículo de la página para que aparezcan de color azul, subrayados y con tamaño de texto 22px.

Eventos en JavaScript

Daniel López Lozano

Escuela Arte Granada

- Hasta ahora, todos los scripts que hemos desarrollado no se ejecutaban hasta que la página se cargase o tras aplicar un intervalo de tiempo.
- Este tipo de scripts no son demasiado útiles debido a que no se ejerce una interacción con el usuario que utiliza nuestra aplicación.
- Lo que más caracteriza a Javascript es en el modelo de programación basada en eventos.
- Este tipo de scripts, esperan a que el usuario “haga algo” (pulsar una tecla, mover el ratón, pulsar un botón del raton…)

- Responde al usuario de alguna manera al realizar una de estas acciones.
- Los eventos permiten que los usuarios transmitan información a los programas, Javascript define numerosos eventos dentro de una web.
- Javascript permite asignar una función a cada uno de los eventos que se produce en la web como respuesta.
- De esta forma, cuando se produce cualquier evento, Javascript ejecuta su función asociada a dicha acción.

- Una de las principales incompatibilidades entre navegadores se producen en la gestión de eventos. Existen distintos modelos:
- Modelo tradicional de eventos: Desarrollado para la versión 4 de HTML, es el único modelo compatible con todos los navegadores. (Atributos y semánticos)
- Modelo de eventos estándar W3C: Introducido con la segunda versión del DOM (DOM2). Todos los navegadores lo incluyen, salvo Internet Explorer.
- Modelo de eventos de Internet Explorer: con el modelo estándar.

Manejadores como atributos HTML

```html
<input type="button" value="Pinchame y verás" onclick="alert('Gracias por pinchar');" />

<div style="background:tomato;height: 300px;text-align: center; font-size: 2em"
     onclick="alert('Has pinchado con el ratón');"
     onmouseover="alert('Has pasado por encima con el ratón');">
     Pasa por encima o pincha
</div>
```

```html
<input type="button" value="Pinchame y verás"
       onclick="SaludoBoton();" />

<div style="background:tomato;height:300px;
            text-align:center; font-size: 2em"
     onclick="SaludoDiv()"
     onmouseover="PasarPorEncima()">
     Pasa por encima o pincha
</div>
```

```javascript
function SaludoBoton()
{
   alert('Gracias por pinchar');
}

function SaludoDiv()
{
    alert('Has pinchado con el ratón');
}

function PasarPorEncima()
{
    alert('Has pasado por encima con el ratón');
}
```

Manejadores semánticos

- Los eventos semánticos son una evolución del método de funciones externas y se basa en utilizar las propiedades del DOM de los elementos HTML para asignar todas las funciones.

```
window.onload=Inicio;
function Inicio()
{
    var div_eventos=document.querySelector("#eventos");
    div_eventos.onclick=SaludoDiv;
    div_eventos.onmouseover=PasarPorEncima;
}
```

Manejadores modelo estándar de la W3C

- El funcionamiento de los eventos estándar W3C es muy parecido a los manejadores de eventos tradicionales en su versión semántica:

```
window.onload=Inicio;
function Inicio()
{
    var div_eventos=document.querySelector("#eventos");
    div_eventos.addEventListener("click",SaludoDiv);
    div_eventos.addEventListener("mouseover",PasarPorEncima);
}
```

- Lo interesante de los manejadores de eventos semánticos y de la W3C es que también se pueden eliminar los manejadores.

```
div_eventos.onclick=null; //elimina la función asignada
                          //para ese elemento y ese evento.
div_eventos.removeEventListener("click", SaludoDiv);
```

- Nosotros usaremos la versión W3C (addEventListener)

- **EJERCICIO 1 (Eventos):** Con un div con las mismas características del que hemos usado hasta ahora, asignarle manejadores de eventos para el click, el doble click, botón derecho, entrar en el div y salir del div.
- Cada función manejadora modificara el texto del div, por ejemplo si el evento es doble click el texto debe ser “Has hecho doble click”. Usar la propiedad innerHTML.
- **EJERCICIO 2:** Repetir el ejercicio anterior pero además hacer que cada evento cambie alguna propiedad CSS (cada evento una propiedad distinta). Usar la propiedad style.

- **EJERCICIO 3:** Crear una pagina con 4 párrafos distintos con atributos id distintos. Cuando se haga click sobre algún párrafo borradlo. (Propiedad CSS display)
- **EJERCICIO 4:** Añadir al anterior un botón por cada párrafo con un atributo id. El texto del botón por defecto será “Ocultar”. Al hacer click en el botón desaparecerá el párrafo correspondiente y el texto del botón cambiara a “Mostrar”

Variable this

- Al igual que en otros lenguajes de programación, en JavaScript podemos acceder a la variable this.
- Referencia al elemento u objeto que está invocando a un evento.
- Para el manejo de eventos es muy útil ya que permite acceder al elemento que lanzó el evento sin tener que recuperarlo de nuevo.
- Esto evita repetir código de manera innecesaria.

- Para resolver el ejercicio 3 podemos hacerlo de la siguiente manera, usando un solo manejador.

```
var parrafo1=document.querySelector("#p1");
parrafo1.addEventListener("click",Ocultar);
//resto de parrafos igual
function Ocultar()
{
    this.style.display="none";
}
```

- Al ser el manejador el mismo podemos simplificar todavía mas el código sin tener que usar id.
- Podemos usar selector por etiqueta o por clase y asignar manejadores por bucle.

```
var parrafos=document.querySelectorAll("p");
for(var i=0;i<parrafos.length;i++)
{
    parrafos[i].addEventListener("click",Ocultar);
}
```

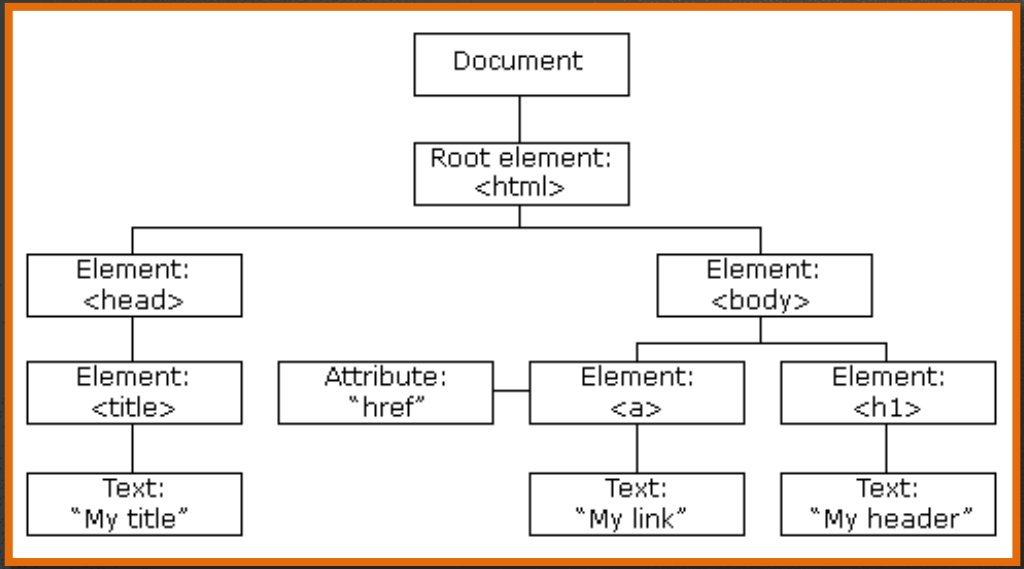
- Si tuviéramos por ejemplo, varios enlaces distintos con distintos destinos con la variable this podríamos recuperarlos sin usar id y solo una función manejadora.
- Como veremos más adelante esto es realmente útil.

```
<a href="destino1.html">Enlace a destino1</a>
<a href="destino2.html">Enlace a destino2</a>
<a href="destino3.html">Enlace a destino3</a>
<a href="destino4.html">Enlace a destino4</a>
```

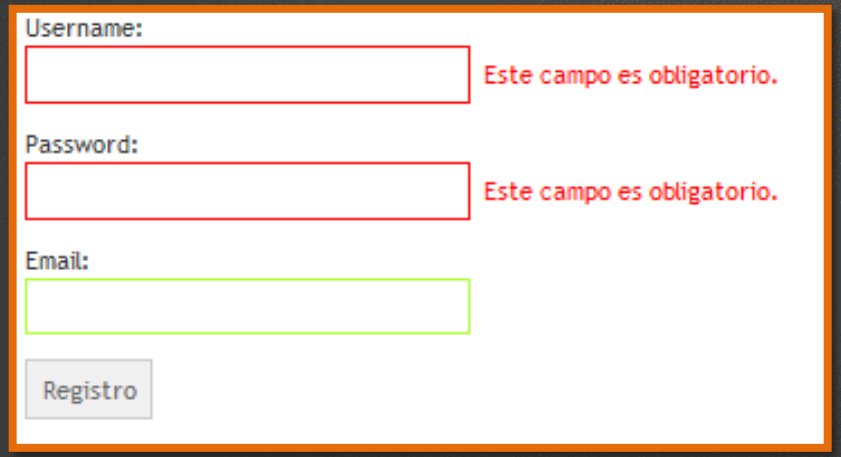
```
var anclas=document.querySelectorAll("a");
for(var i=0;i<anclas.length;i++)
{
 anclas[i].addEventListener("click",VerDestino);
}

function VerDestino()
{
    alert(this.href)
}
```

Propiedad nextSibiling

- Dado un elemento este nos devuelve el que esta justamente a su derecha en el árbol de nodos.
- Sibling significa hermano.



- Esto es muy importante cuando queremos que un evento modifique el estado o aspecto de un elemento cercano a el sin usar id y tener que usar de nuevo un querySelector



- Importante que en el código HTML no haya espacios entre los elementos hermanos.

```
<input type="button" name="Nombre" value="Ocultar"><p id="p1">
   Parrafo1Parrafo1Parrafo1Parrafo1Parrafo1Parrafo1Parrafo1Parrafo1Parrafo1
</p>
```

```
var parrafo=document.querySelector("#p1");
parrafo.addEventListener("click",VerHermano);

function VerHermano()
{
    var hermano=this.nextSibling;
    alert(parrafo.innerHTML);
}
```

- Existe también la propiedad previousSibling que nos da acceso al hermano de la izquierda (previo) al actual nodo
- **EJERCICIO 5:** Resolver el ejercicio 4 usando la variable this, nextSibling y sin depender de atributo id para simplificar el código enormente.

Eventos de teclado y ratón

- Para recuperar información sobre este tipo de eventos(posición del raton, tecla o botón que se ha pulsado) es necesario el objeto event.
- El principal inconveniente es que los distintos navegadores no tratan igual los eventos.
- Principalmente Internet Explorer es diferente a los demás, aunque también Firefox presenta diferencias.
- El objeto event dependiendo del navegador se recoge de una manera u otra.

- Cuando queremos detectar eventos de teclado, no sólo saber que se ha pulsado una tecla (onkeypress) también necesitamos saber que tecla se ha pulsado.
- Esta tarea depende del tipo de navegador, pero con el siguiente código podemos obtenerla de forma genérica:

```
function manejador(info_evento) {
    var evento = info_evento || window.event; // recupero el evento según el navegador
    var caracter = evento.charCode || evento.keyCode; // recupero la tecla según nav.
    var letra=String.fromCharCode(caracter); //transformo de ascii a texto
    alert("El carácter pulsado es: " +letra); //la utilizo para lo que necesite
}
document.addEventListener("keypress",manejador);
//Tambien valdria
document.onkeypress=manejador;
```

- **EJERCICIO 6:** Realizar una página web que muestre la tecla que se pulsa. Desde JavaScript capturar las teclas que se pulsen en el teclado y poner en un div un mensaje Se ha pulsado la tecla...
- Cuando queremos detectar pulsaciones del ratón, otra de las informaciones que es interesante conocer es en que parte de la pantalla se encuentra el ratón.
- Como siempre, las coordenadas se calculan desde la esquina superior izquierda del navegador:

```
function muestraInformacion(info_evento) {
    var evento = info_evento || window.event; //recupera el evento según navegador
    var coordenadaX = evento.clientX; //coordenada x
    var coordenadaY = evento.clientY; //corrdenada y
    alert("Has pulsado el ratón en la posición: " + coordenadaX + ", " + coordenadaY);
}
document.addEventListener("keypress",muestraInformacion);
//Tambien valdria
document.onclick = muestraInformacion;
```

- **EJERCICIO 7:** Mostrar en un div con id “info” la posición del ratón. La posición del ratón debe actualizarse cada vez que se mueva el ratón.

- El objeto event proporciona una función muy interesante que se llama preventDefault() que evita que se ejecuta la acción por defecto.
- Por ejemplo al pulsar un enlace automáticamente va a su destino o cuando pulsamos un botón submit envía automáticamente los datos.
- preventDefault() cancela dichos comportamientos por ejemplo cuando algún datos no cumple con los requisitos (DNI o email erróneo, etc)
- Lo utilizaremos en formularios sobre todo

- Enlace a google que no funciona y hace algo en su lugar , comprobamos que no hace lo que programamos por eso prevent default
- **EJERCICIO 7:** Dado un enlace con un destino, que aparezca un dialogo modal que pregunte antes si quiere abandonar la página actual(confirm)

Bibliografía

- Gauchat, Juan Diego: **“El gran libro de HTML5, CSS3 y Javascript”**.Editorial Marcombo. 2012
- Vara, J.M. y otros: **“Desarrollo Web en Entorno Cliente. CFGS”**.Editorial Ra-Ma. 2012
- **“Programación en Javascript”.** Colección de artículos disponibles en la url. http://www.desarrolloweb.com/manuales/ Última visita: Septiembe 2017.
- **W3SCHOOL “Manual de referencia y Tutoriales”** http://www.w3schools.com/ Última visita Septiembre 2017
- **Comesaña, J.L. : Técnico en Desarrollo de Aplicaciones Web.** http://www.sitiolibre.com/daw.php Última visita Septiembre 2017