UD 04. DOM Y EVENTOS.

Índice de contenido

TEMA 4 DOM Y EVENTOS.

1. INTRODUCCIÓN.

DOM (Document Object Model) es un modelo que permite tratar un documento Web XHTML como si fuera XML, navegando por los nodos existentes que forman la página, pudiendo manipular sus atributos e incluso crear nuevos elementos.

Para más información general de DOM:

- https://es.wikipedia.org/wiki/Document_Object_Model
- http://www.w3schools.com/js/js_htmldom.asp

Usando Javascript para navegar en el DOM podemos acceder a todos los elementos XHTML de una página. Esto nos permite cambiar dinámicamente el aspecto de nuestras páginas Web.

En este tema vamos a estudiar las principales funciones de Javascript para modificar el DOM.

2. ATRIBUTOS DISPONIBLES PARA MODIFICAR

Cuando obtengamos algún elemento con las funciones que estudiaremos más adelante, podemos manipular los atributos de dicho elemento de la siguiente forma.

```
var elemento=document.getElementById(“miElemento”);
elemento.innerHTML=”El html interno a cambiar de ese elemento”;
```

Los atributos disponibles a modificar pueden depender de cada elemento.

3. FUNCIONES JAVASCRIPT PARA LOCALIZAR ELEMENTOS EN DOM

Las funciones aquí estudiadas normalmente se usan sobre el elemento “document”, ya que así se aplican a todo el documento.

Aun así, pueden usarse en cualquier nodo XHTML, entonces la búsqueda se realizaría no en todo en el documento, sino en al sub-árbol formado por el elemento en sí y sus hijos.

3.1 getElementById(identificador)

Esta función devuelve un elemento DOM del sub-árbol cuyo identificador sea el indicado en la cadena “identificador”.

http://www.w3schools.com/jsref/met_document_getelementbyid.asp

Ejemplo:

```
var myDiv = document.getElementById("miDiv");
alert(“El html de miDiv es “+myDiv.innerHTML);
```

3.2 getElementsByTagName(etiqueta)

Esta función devuelve un array con todos los elementos DOM del sub-árbol cuya etiqueta XHTML sea la indicada en la cadena “etiqueta”.

http://www.w3schools.com/jsref/met_document_getelementsbytagname.asp

Ejemplo:

```
var myDiv = document.getElementById("miDiv")
var losP = myDiv.getElementsByTagName("p");
var num = losP.length;
alert("Hay " + num + " <p> elementos en el elemento miDiv”);
```

```
alert("En el primer P el HTML asociado es ”+losP[0].innerHTML);
```

3.3 getElementsByName(nombre)

Esta función devuelve un array con todos los elementos DOM del sub-árbol cuyo atributo name sea el indicado en la cadena “nombre”.

http://www.w3schools.com/jsref/met_doc_getelementsbyname.asp

Ejemplo:

```
var x = document.getElementsByName("name");
var i;
// Todos los textbox que tengan de name alumnos, los marcamos
for (i = 0; i < x.length; i++) {
        if (x[i].type == "checkbox") {
                x[i].checked = true;
        }
}
```

3.4 getElementsByClassName (clase)

Esta función devuelve un array con todos los elementos DOM del sub-árbol cuya clase sea el indicado en la cadena “clase”.

https://www.w3schools.com/jsref/met_document_getelementsbyclassname.asp

4. FUNCIONES PARA CREAR/ELIMINAR NODOS

En esta parte veremos las funciones básicas para crear y eliminar nodos XHTML.

4.1 removeChild(nodo)

Esta función se aplica a un nodo padre. La función recibe un nodo hijo suyo y lo borra. Es útil usarlo con el atributo “parentnode”, que devuelve el nodo padre del elemento que estamos manejando.

http://www.w3schools.com/jsref/met_node_removechild.asp

Ejemplo:

```
var parrafo=document.getElementById(“miParrafo”);
// Obtiene la referencia del padre, y al padre le aplica la función removeChild
parrafo.parentnode.removeChild(parrafo);
```

4.2 appendChild(nodo)

Esta función se aplica a un nodo padre. La función recibe un nodo y lo incluye como nodo hijo del padre. Se puede combinar con funciones como “createElement”, que permiten crear elementos XHTML.

http://www.w3schools.com/jsref/met_node_appendchild.asp

Ejemplo:

```
// Creo un nodo de tipo LI
var nuevoNodo = document.createElement("LI");
// Al nodo LI le asocio un texto (tambien podria asociarle XHTML con innerHTML)
```

```
var nodoTexto = document.createTextNode("Agua");
nuevoNodo.appendChild(nodoTexto);
// A miLista, lista ya existente, le añado el elemento creado
document.getElementById("miLista").appendChild(nuevoNodo);
```

4.3 setAttribute().

Establece el valor de un atributo en el elemento indicado. Si el atributo ya existe, el valor es actualizado, en caso contrario, el nuevo atributo es añadido con el nombre y valor indicado.

```
caja.setAttribute('class', 'caja naranja');
```

También se puede acceder a los atributos y modificar sus valores.

```
caja.className = 'caja naranja';
```

Para obtener el valor actual de un atributo, se utiliza getAttribute(); para eliminar un atributo, se llama a removeAttribute(), para reemplazarlo replaceChild(nuevoElemento, viejoElemento).

5. EVENTOS.

Los eventos son manejadores que nos proporciona el navegador para que cuando detecte que se produzca una acción (evento), se ejecute un código asociado a esa acción. En esta unidad trataremos los eventos existentes en Javascript y las distintas formas de manejarlos.

5.1 Principales eventos

A continuación, mostramos un listado de los **principales** eventos existentes en Javascript:

- onfocus: al obtener un foco.
- onblur: al salir del foco de un elemento.
- onchange: al hacer un cambio en un elemento.
- onclick: al hacer un click en el elemento.
- ondblclick: al hacer doble click en un elemento.
- onkeydown: al pulsar una tecla (sin soltarla).
- onkeyup: al soltar una tecla pulsada.
- onkeypress: al pulsar una tecla.
- onload: al cargarse una página.
- onunload: al descargarse una página (salir de ella).
- onmousedown: al hacer clic de ratón (sin soltarlo).
- onmouseup: al soltar el botón del ratón previamente pulsado.
- onmouseover: al entrar encima de un elemento con el ratón.
- onmouseout: al salir de encima de un elemento con el ratón.
- onsubmit: al enviar los datos de un formulario.

- onreset: al resetear los datos de un formulario.
- onselect: al seleccionar un texto.
- onresize: al modificar el tamaño de la página del navegador.

> El total de eventos disponibles está descrito en http://www.w3schools.com/jsref/dom_obj_event.asp

5.2 Manejo de eventos desde elementos XHTML

La forma más sencilla (aunque menos práctica para tener un código limpio y ordenado) de indicar que hay un evento asociado a un elemento XHTML es indicándolo en el propio código.

Ejemplo 1:

```
<input type="button" value="Boton Hola mundo" onclick="alert('Hola mundo');alert('Adios');" />
```

También en lugar de ejecutar una serie de instrucciones, es posible llamar a una función predefinida.

Ejemplo 2:

```
<input type="button" value="Boton miFuncion" onclick="miFuncion('cadenaParam1');" />
```

5.3 Uso del objeto this en gestión de eventos

Cuando ejecutas código dentro de un evento, existe un objeto llamado "this".

Este objeto es una referencia al elemento que se ha producido el evento. Ejemplo, si el evento se ha producido un evento al hacer clic a una imagen con id="miImagen", el objeto "this" será lo mismo que poner "document.getElementById("miImagen");

Ejemplo sin this:

```
<div id="cont" style="width:150px; height:60px; border:thin solid silver"
  onmouseover="document.getElementById('cont').style.borderColor='black';"
onmouseout="document.getElementById('cont').style.borderColor='red';">
  Contenidos
</div>
```

Equivalente con "this":

```
<div id="cont" style="width:150px; height:60px; border:thin solid silver"
  onmouseover="this.style.borderColor='black';" onmouseout="this.style.borderColor='red';">
  Contenidos
</div>
```

5.4 Asignación de eventos desde código a objetos XHTML

Podemos asignar/modificar mediante código el manejador de un evento predefinido en un objeto XHTML de una forma similar a esta. Supongamos que tenemos un objeto con id=”miObjeto” que posee el evento “onclick” sin asignar y la función “mostrarMensaje”.

Podemos referenciar al elemento XHTML como vimos las funciones, asignar la función como manejador del evento como vemos en este código

```
function mostrarMensaje(){
        alert(“Hola”);
}
document.getElementById(“miObjeto”).onclick=mostrarMensaje;
```

> Atención: fijaros que pone “mostrarMensaje”. Así asigna la función como manejadora del evento. Si ponemos “mostrarMensaje()” no funcionará, ya que ejecutará la función y asignará su resultado al manejador.

5.5 Obteniendo información del objeto event

Cuando se crea una función como manejador, al producirse el evento el navegador automáticamente manda como parámetro un objeto de tipo event.

```
function mostrarMensaje(evento) {
        alert(evento.type);
}
document.getElementById(“miObjeto”).onclick=mostrarMensaje;
```

Este objeto posee cierta información útil del evento que se ha producido.

Entre otros atributos:

- type: dice el tipo de evento que es (“click”, “mouseover”, etc...). Devuelve el nombre del evento tal cual, sin el “on”. Es útil para hacer una función que maneje varios eventos.
- keyCode: en eventos de teclado, almacena el código de tecla de la tecla afectada por el evento.
- clientX / clientY: en eventos del ratón, devuelve las coordenadas X e Y donde se encontraba el ratón, tomando como referencia al navegador.
- screenX / screenY: en eventos del ratón, devuelve las coordenadas X e Y donde se encontraba el ratón, tomando como referencia la pantalla del ordenador.

Ejemplo:

```
function mostrarMensaje(evento){
        if(evento.type==”keyup”){
                alert(evento.keyCode);
        }
        else if(evento.type==”click”){
                alert(evento.clientX+” “+evento.clientY);
        }
}
document.getElementById(“miObjeto”).onclick=mostrarMensaje;
document.onkeyup=mostrarMensaje;
```

5.6 Declarando eventos desde código.

Podemos declarar eventos mediante código usando addEventListener(evento,manejador).
Ejemplo:

```
function mostrarMensaje(evento){
        if(evento.type==”keyup”){
                alert(evento.keyCode);
        }
        else if(evento.type==”click”){
                alert(evento.clientX+” “+evento.clientY);
        }
}
document.getElementById(“miObjeto”).addEventListener(“click”,mostrarMensaje);
document.addEventListener(“keyup”,mostrarMensaje);
document.getElementById(“miObjeto”).addEventListener(“dblclick”,function (){
 alert(“Codigo metido directamente”);
});
```

6. DRAG AND DROP

Además de los eventos típicos tratados, existe un proceso (en el que entran en juego varios eventos) que es útil para el desarrollo de aplicaciones Web, el “Drag and Drop” (Arrastrar y soltar).

En W3Schools podéis obtener teoría y ejemplos de esta técnica https://www.w3schools.com/html/html5_draganddrop.asp

Hay muchos eventos que se utilizan y pueden ocurrir en las diferentes etapas de una operación de arrastrar y soltar:

Eventos disparados en el objetivo arrastrable (el elemento fuente):

- ondragstart: ocurre cuando el usuario comienza a arrastrar un elemento
- ondrag: ocurre cuando se arrastra un elemento
- ondragend: ocurre cuando el usuario ha terminado de arrastrar el elemento

Eventos disparados en el objetivo de caída:

- ondragenter: - ocurre cuando el elemento arrastrado ingresa al destino de la gota
- ondragover: ocurre cuando el elemento arrastrado está sobre el destino de colocación
- ondragleave: ocurre cuando el elemento arrastrado abandona el destino de colocación
- ondrop: ocurre cuando el elemento arrastrado se coloca en el destino de la gota.

Aquí un ejemplo completo:

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function allowDrop(ev) {
  ev.preventDefault();
}

function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<img id="drag1" src="img_logo.gif" draggable="true" ondragstart="drag(event)" width="336" height="69">

</body>
</html>
```

7. MATERIAL ADICIONAL

[1] Curso de Javascript en Udacity https://www.udacity.com/course/javascript-basics--ud804

[2] Enlaces Libros Web con multitud de ejemplos de DOM y Javascript
http://librosweb.es/libro/ajax/capitulo_4/html_y_dom.html

8. BIBLIOGRAFÍA

[1] Referencia Javascript

http://www.w3schools.com/jsref/

[2] Referencia DOM

https://developer.mozilla.org/es/docs/Referencia_DOM_de_Gecko/Introducci%C3%B3n

[3] Referencia eventos Javascript

http://www.w3schools.com/jsref/dom_obj_event.asp

[4] Libros Web: manejo de eventos

http://librosweb.es/libro/javascript/capitulo_6.html

[5] Libros Web: objeto event

http://librosweb.es/libro/javascript/capitulo_6/obteniendo_informacion_del_evento_objeto_event.html