UNIDAD 7.
DOM

Desarrollo web en entorno cliente
CFGS DAW

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 Importante

 Atención

 Interesante

ÍNDICE DE CONTENIDO

UD07. DOM

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 mas 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 el sub-árbol formado por el elemento en sí y sus hijos.

3.1 getElementById(identificador)

Esta función devuelve un elemento DOM del sub-árbol cuya 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 una 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 una array con todos los elementos DOM del sub-árbol cuya 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;
        }
}
```

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);
```

5. 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

6. BIBLIOGRAFÍA

[1] Referencia Javascript

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