UD8-DOM

Contenido

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 queforman la página, pudiendo manipular sus atributos e incluso crear nuevos elementos.

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 paramodificar 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 por 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)

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”.

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”.

Ejemplo:

```
var x = document.getElementsByName("alumnos ");
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.

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.

Ejemplo:

```
// Creo un nodo de tipo LI
var nuevoNodo = document.createElement("LI");
// Al nodo LI le asocio un texto
nuevoNodo.textContent = ”Agua”;
// var nodoTexto = document.createTextNode("Agua"); //otra forma de hacerlo
// nuevoNodo.appendChild(nodoTexto);
// A miLista, lista ya existente, le añado el elemento creado
document.getElementById("miLista").appendChild(nuevoNodo);
```