INTRODUCCIÓN A LA MANIPULACIÓN DEL DOM

El llamado **DOM** (**D**ocument **O**bject **M**odel) 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.

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.

Con el fin de facilitar la realización de pequeños ejercicios, incluimos unos primeros conceptos de manipulación del **DOM** (**Document Object Model**). Conociendo la **id** de algún objeto HTML existente en la página podemos cambiar sus propiedades.

Cuando queremos cambiar una propiedad de un objeto presente en el documento, utilizamos su **ID** (nombre único que define a un elemento) y el método "**document.getElementById()**" para acceder al elemento en sí.

MODIFICANDO ATRIBUTOS

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

```
let elemento=document.getElementById("miElemento");
elemento.innerHTML="El html interno a cambiar de ese elemento";
```

Los atributos disponibles a modificar pueden depender de cada elemento.

FUNCIONES DE JAVASCRIPT PARA LOCALIZAR ELEMENTOS EN EL 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.

GetElementById (*identificador*)

Esta función devuelve un elemento DOM del subárbol cuya identificador sea el indicado en la cadena "identificador".

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

GetElementsByTagName (etiqueta)

Esta función devuelve una array con todos los elementos DOM del subárbol cuya etiqueta XHTML sea la indicada en la cadena "etiqueta".

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

Ejemplo:

```
let myDiv = document.getElementById("miDiv")
let losP = myDiv.getElementsByTagName("p");
let num = losP.length;
console.log("Hay " + num + " <p> elementos en el elemento miDiv");
console.log("En el primer P el HTML asociado es "+losP[0].innerHTML);
```

GetElementsByName (*nombre*)

Esta función devuelve una array con todos los elementos DOM del subárbol cuya atributo name sea el indicado en la cadena "nombre".

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

Ejemplo:

Para cambiar una imagen con **id 'matrix'**, accedemos al elemento y modificamos la propiedad src :

```
document.getElementById('matrix').src = "mt05.jpg";
```

Esto se puede usar *para cualquier propiedad existente en cualquier objeto HTML*.

Una función Javascript genérica para cambiar una imagen sería:

```
function cambiarImagen (id, rutaImagen) {
  document.getElementById(id).src=rutaImagen;
}
```

FUNCIONES PARA CREAR/ELIMINAR NODOS

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

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:

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

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
let nuevoNodo = document.createElement("LI");
// Al nodo LI le asocio un texto (también podría asociarse XHTML con innerHTML)
let nodoTexto = document.createTextNode("Agua");
nuevoNodo.appendChild(nodoTexto);
// A miLista, lista ya existente, le añado el elemento creado
document.getElementById("miLista").appendChild(nuevoNodo);
```

Document Object Model (DOM) – Ejercicios

Importante 1: no intentes copiar ejercicios ni tan siquiera "ver un poco" código de otros compañeros. Es el mayor error de quien empieza a programar, ya que luego no sabe resolver problemas por sí mismo y da una falsa sensación de aprendizaje.
Importante 2: si en programación algo no sale a la primera... es totalmente normal. Es parte del aprendizaje. ¿Cómo crees que aprendieron los mejores programadores?

ACTIVIDAD 1

Realiza un programa que cuando se pulse un botón con el texto "Nuevo número", añada un elemento con un número aleatorio a una lista desordenada (elemento UL).

ACTIVIDAD 2

Realiza un programa que cree dinámicamente una tabla de 100x100. Cada elemento de la tabla
tendrá un número único, que empezará en 1 y se irá incrementando de 1 en 1.
Esta página además tendrá un botón que será "Calcular numero casi primos". Este botón hará que
todas las celdas de la tabla que tengan números "Casi primos" se pongan con un fondo amarillo.
Definimos aquí que es un "Número casi primo":

- Número casi primo : es un número que solo es divisible por sí mismo, la unidad y por un solo
número que no sea ni la unidad ni si mismo.

Ejemplo:
2 no es un número casi primo, porque es divisible por 1 y por 2, pero no por otro número. 4 es un número casi primo, porque es divisible por 1, por 4 y por 2. 8 no es un número casi primo, porque es divisible por 1, por 8 y por 2, pero además también es divisible por 4.

ACTIVIDAD 3

Realiza un programa que cree 100 elementos "checkbox" con números aleatorios. Además la página tendrá un botón "Marcar todos" y un botón "Desmarcar todos", con su correspondiente funcionalidad.

ACTIVIDAD 4

Realiza un programa que tenga 3 elementos <p> y al hacer clic sobre ellos desaparezcan (se oculten) y al hacer doble clic (los elimine del DOM). También deberá tener un botón "Reaparecer" que hará que aparezcan todos los elementos desaparecidos (pero no los eliminados).