PROGRAMACION WEB

Nombre de la practica: El Modelo de Objetos de Documento (DOM)

I. OBJETIVOS

Que el estudiante:

- Adquiera un dominio amplio del Modelo de Objetos de Documento (DOM) definido por la W3C.
- Tenga la capacidad de acceder, eliminar y modificar el contenido de una página web a través del acceso a los nodos que forman el documento.
- Empleen los métodos y propiedades del estándar DOM para tener un dominio completo sobre el documento HTML.
- Haga uso de algunos de los elementos del Modelo de Objetos Tradicional para comparar su eficacia en comparación con el DOM.
- Realice scripts en los que acceda a los elementos del documento web utilizando el Modelo de Objetos Tradicional.

II. INTRODUCCION TEORICA

Los modelos de objetos definen la interfaz para los diversos aspectos del navegador y del documento web que podrán controlarse haciendo uso de JavaScript. En JavaScript se emplean dos modelos de objetos fundamentales, que son:

1. El Modelo de Objetos de Navegador (BOM), y
2. El Modelo de Objetos de Documento (DOM)

El **BOM (Browser Object Document)** proporciona acceso a las diversas características del navegador. Por ejemplo, la ventana del navegador, las características de la pantalla, el historial del navegador, etc. Entre tanto, el **DOM (Document Object Model)**, proporciona acceso al contenido de la ventana del navegador; es decir, el documento web que se muestra (aunque en algunos casos puede ser el conjunto de marcos que forman la página) incluidos los diversos elementos HTML que estén presentes en él, como enlaces, imágenes, tablas, etc.

Uno de los problemas principales es que la división entre el BOM y el DOM es poco clara. Además, las implementaciones de JavaScript de los diferentes navegadores varían significativamente.

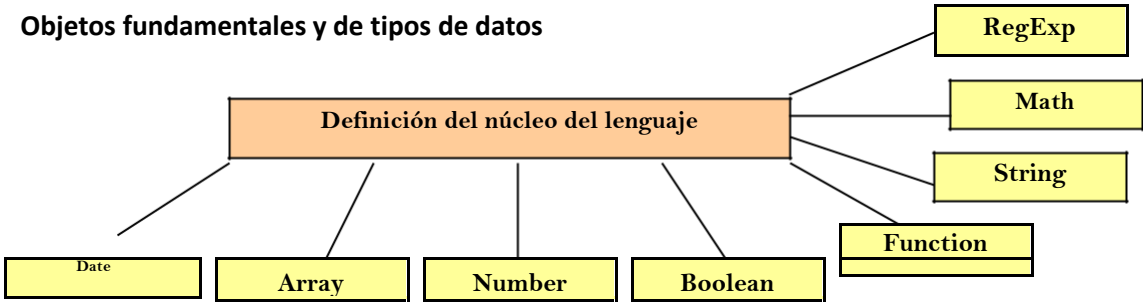
Descripción general del modelo de objetos

El modelo de objetos posee cuatro componentes principales, que son:

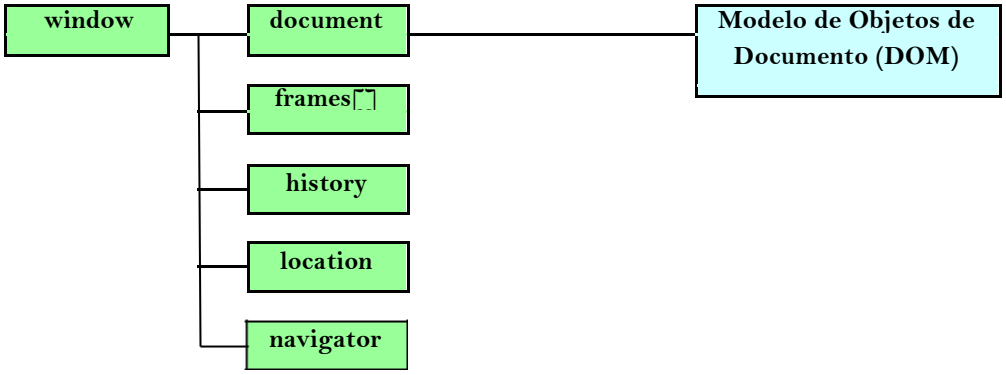
1. Los elementos fundamentales del lenguaje JavaScript (tipos de datos, operadores, instrucciones, etc)
2. Los objetos fundamentales relacionados esencialmente con los tipos de datos (Date, String, Math, etc)
3. Los objetos de navegador (Window, Navigator, Location, etc)
4. Los objetos de documento (Document, Form, Image, etc)

En la siguiente figura se muestran estos componentes:

Objetos fundamentales y de tipos de datos

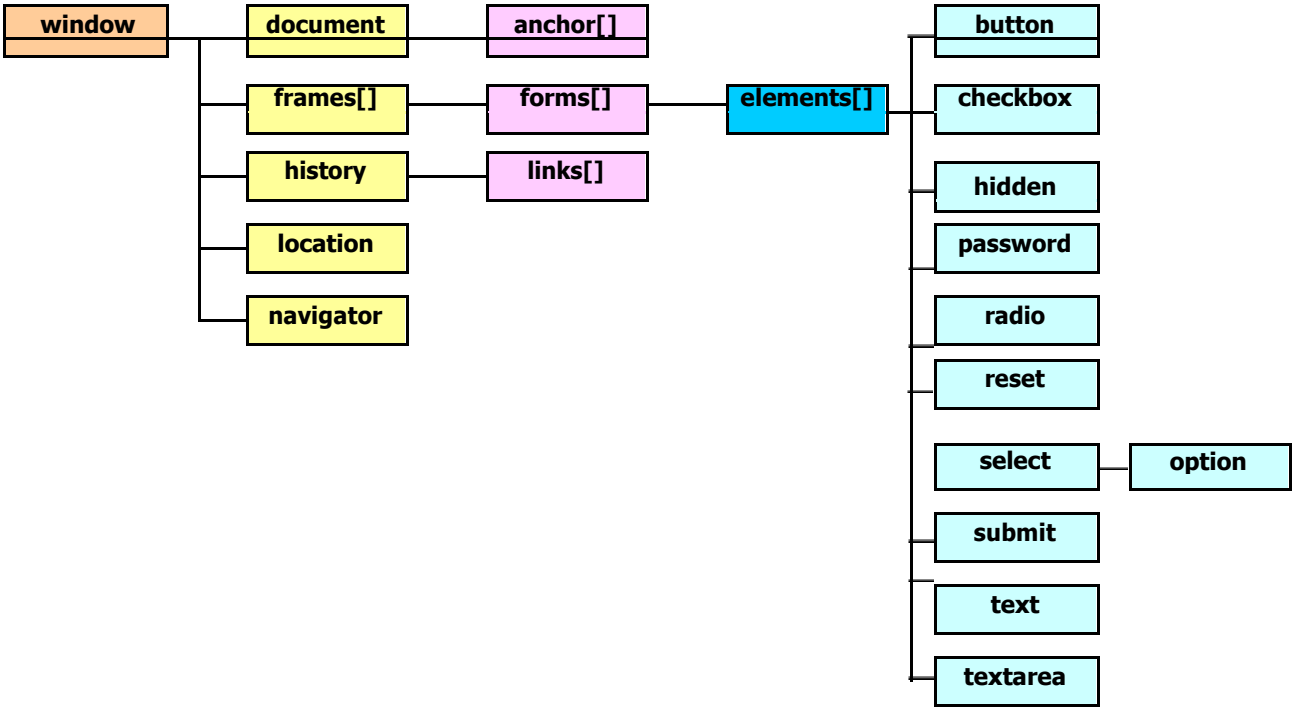


Modelos de objetos del navegador y del documento



Modelo de objetos tradicional

En un principio el modelo de objetos de JavaScript era bastante limitado y se concentraba en dar acceso a las características básicas del navegador y del documento. El modelo de objetos tradicional de JavaScript se muestra a continuación:



Todos estos objetos mostrados se relacionan con JavaScript mediante una sintaxis muy particular, en la que una propiedad de un objeto puede ser también objeto para otro conjunto de propiedades. Así por ejemplo, la propiedad document del objeto Window es objeto para el arreglo (o colección) forms[]. Del mismo modo la colección forms [] tiene los distintos elementos de formulario como propiedades.

Veamos algunos ejemplos que ya conocemos:
Para acceder al método alert () del objeto Window, escribimos la siguiente instrucción:
window.alert (mensaje);

Del mismo modo para un diálogo de confirmación o de ingreso de datos usamos los métodos confirm() y prompt() respectivamente.

Para mandar a desplegar un mensaje directamente en el documento web, escribimos una instrucción como la siguiente:
window.document.write ("Vamos a aprender a usar el DOM");

La descripción general de cada uno de los objetos mostrados en el modelo de objetos tradicional de JavaScript se muestra en la siguiente tabla:

Objeto	Descripción
window	Este objeto se refiere a la ventana del navegador actual
document	Este objeto contiene todos los elementos HTML y todos los fragmentos de texto que conforman el documento web cargado en la ventana del navegador. En el modelo tradicional de JavaScript hace referencia a lo que se encierra entre las etiquetas <body>… </body> del documento web actual.
frames[]	Es una colección con los marcos del documento web actual, si acaso el documento web está formado por marcos. Sólo bajo esta circunstancia esta propiedad contendrá datos. Cada marco de esta colección referencia a otro objeto window, que a su vez, puede contener otros marcos.
history	Es un objeto que tiene el historial de la ventana actual. Específicamente el conjunto de las diversas direcciones URL visitadas por el usuario recientemente.
position	Este objeto contiene la posición actual del documento que se ha visto en forma de dirección URL y las piezas que lo constituyen, como protocolos, nombre de host y rutas.
navigator	Este objeto muestra características básicas del navegador, como su tipo y versión.

Cuando se están haciendo scripts para controlar dinámicamente lo que se debe mostrar en la página web tenemos varias opciones: por posición, por nombre o por ID. Este último método se explicará en la sección del Modelo de Objetos de Documento (DOM).

Acceso a los elementos del documento por posición

El modelo de objetos de JavaScript tradicional facilita el acceso a un conjunto limitado de elementos HTML, tales como: anclas, enlaces, formularios y controles (elementos) de formulario.

Cuando se carga una página web dentro de la ventana de un navegador se instancian objetos de JavaScript para todos los elementos HTML que son codificables en secuencias de comandos scripts. Ya se dijo que este conjunto de elementos codificables es limitados para el modelo de objetos tradicional.

Al observar el siguiente ejemplo de documentos web, notaremos la forma en que podemos acceder a los elementos del documento web a través de su posición dentro de la página web generada:

```
<html>
<head>
   <title>Formulario sencillo</title>
</head>
<body>
<form>
   <input type="text">
</form>
<br><br>
<form>
   <input type="text"><br>
   <input type="text">
</form>
</body>
</html>
```

El ejemplo anterior contiene dos formularios y tres cuadros de texto. Para poder acceder a la primera etiqueta <form> usando acceso por posición de acuerdo al modelo tradicional, debería escribirse una sentencia como la siguiente:

```
window.document.forms[0];
```

Para acceder al segundo, habría que digitar:

```
window.document.forms[1];
```

Del mismo modo si queremos tener acceso al segundo cuadro de texto del segundo formulario. Debe digitar algo como esto:

```
window.document.forms[1].elements[1].text;
```

El método anterior es funcional en tanto no se modifique la posición de los elementos dentro del documento web. Esto es el principal inconveniente, porque en muchas ocasiones se requerirá cambiar de posición los elementos de formulario, o incluso, eliminarlos o agregar nuevos. Esto generaría un caos en el código JavaScript que se tuviese digitado en ese momento, porque sería necesario cambiarlo todo. Por ello se ideó una mejor forma de tener acceso a los elementos del documento, que es la que se explica a continuación.

Acceso a los elementos por nombre

Es muy conveniente la colocación de nombres para todos los elementos HTML que se utilizan en una página. Esto facilita que los lenguajes de script tengan acceso a ellos más adelante. Se puede utilizar el atributo ***id***, que es permitido casi en todos los elementos HTML conocidos. El objetivo de este atributo es vincular un identificador único al elemento.

Para acceder al formulario definido por:

```
<form name="myform" id="myform">
   <input type="text" name="username" id="username">
<form>
```

Mediante su nombre, debería digitar lo siguiente:

```
window.document.myform
```

Recuerde que como el objeto window se puede asumir, también sería válido digitar:

```
document.myform
```

Acceso a los elementos usando arreglos asociativos.

Es una estructura que le permite asociar datos con nombres. JavaScript proporciona *arrays* asociativos como consecuencia del hecho de que las siguientes dos instrucciones son equivalentes object.property, object["property"].

En la mayoría de los casos los arreglos (llamados también colecciones) presentes en el objeto **Document** son asociativos. Esto significa que podemos referirnos a ellos haciendo uso de cadenas de texto como índice del arreglo. Esta cadena tiene que ser el nombre del elemento definido al momento de crear el documento web. Se suele utilizar los atributos ***id*** y ***name*** para establecer estos nombres. Se utilizan dos por razones de compatibilidad. Si bien es cierto, en los navegadores más actuales se usa el atributo id, para nombrar al control, navegadores más antiguos no reconocen ese atributo y por eso es que se suele usar el atributo ***name*** junto con el atributo ***id***.

Si se tiene el siguiente ejemplo:

```
<form name="myform">
   <input type="text" name="mytext"
value=""> </form>
```

Para tener acceso al valor introducido en el campo de texto debemos escribir una sentencia como la siguiente:

```
document.forms["myform"].elements["mytext"].value
```

En Internet Explorer se definen colecciones de elementos a las que se puede acceder mediante un método especial denominado ***item()***. Este método acepta la utilización de cadenas de texto para hacer referencia al elemento a recuperar. Para el ejemplo anterior, tendría que escribirse una sentencia como la siguiente:

```
document.forms.item("myform")
```

Modelo de Objetos de Documento

El **DOM** (**Document Object Model**: **Modelo de Objetos de Documento**) es una interfaz de programación de aplicaciones (API) propuesta por la W3C que crea una correspondencia entre un documento HTML o XML y la jerarquía de objetos del documento presentada al programador. Es mediante el uso de esta interfaz que la totalidad de los elementos que componen una página web (etiquetas, atributos, estilos y contenido) están a disposición de un lenguaje de programación o de scripts, como el caso de JavaScript.

De acuerdo con este estándar los documentos HTML son concebidos como una estructura en la que se realiza una correspondencia entre las etiquetas dispuestas en el documento web con un árbol de nodos. La manipulación de estos nodos es el método recomendado por la W3C para que los navegadores compatibles con los estándares soporten las páginas web que funcionan más como aplicaciones que como típicas páginas estáticas.

Entre los usos más comunes del DOM están la exploración de la estructura del documento, el acceso a las propiedades y los métodos comunes, la modificación del contenido de una página web, la eliminación de elementos, la manipulación de reglas de estilo y la creación dinámica de elementos HTML.

Niveles del DOM

El consorcio W3C propone tres niveles del DOM:

DOM Nivel 0: Es el llamado Modelo de Objetos de JavaScript clásico o tradicional. Este nivel del DOM es equivalente a lo que han soportado el Netscape 3.0 e Internet Explorer 3.0.

DOM Nivel 1: Define las interfaces principales del DOM, como Node, Element, Attr y Document, mediante las cuales es posible manipular todos los elementos de un documento, permitiendo leer y escribir partes de una

página web en todo moemento. Este nivel del DOM proporciona capacidades similares a las de la colección document.all[].

DOM Nivel 2: Incorpora elementos de página relacionados con XML y soporte para Hojas de Estilo en Cascada (CSS). Se centra además en la combinación del DOM Nivel 0 y el DOM Nivel 1. Incorpora, además, soporte para el modelo de eventos que fusiona los conceptos de Netscape 4 e Internet Explorer 4.

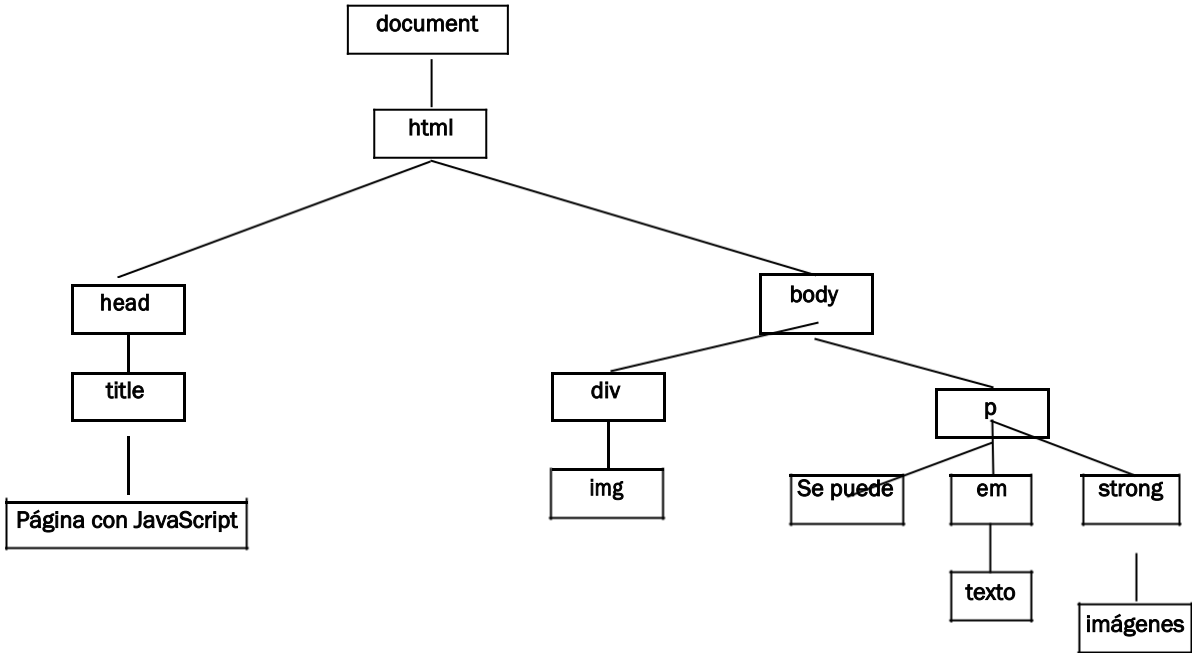
Estructura jerárquica de árbol

Lo que el estándar DOM de la W3C establece, es que todos los elementos presentes en una página web siguen una estructura jerárquica por debajo del objeto document, que se corresponde con el elemento HTML, que se convierte en el elemento padre de todos los elementos presentes en un documento web.

El documento siguiente:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
   <title>P&aacute;gina con JavaScript</title>
</head>
<body>
    <!-- Comentario -->
   <div>
     <img src="imagen.jpg">
   </div>
   <p>
      Se puede incluir <em>texto</em> e
      <strong>im&aacute;genes</strong>.
   </p>
</body>
</html>
```

Se puede representar mediante una estructura jerárquica como la siguiente:



Puede observarse que todos los nodos del árbol de documento son nodos de document. Del mismo modo, puede notarse que head y body son hijos del nodo html, que también es hijo de document. La sección title es hijo de head, y así, sucesivamente.

Tipos de nodos

Los diversos tipos de nodos pueden ser, de acuerdo a su naturaleza, de hasta 12 tipos diferentes. Sin embargo, los que están estrictamente relacionados con HTML, son 6, que se describen a continuación:

No. Tipo de nodo	Tipo	Descripción	Ejemplo
1	Elemento	Un elemento HTML o XML	Texto
2	Atributo	Un atributo para un elemento HTML o XML	cols="12"
3	Texto	Un fragmento de texto incluido en un elemento HTML o XML.	Esto es un fragmento de texto
8	Comentario	Comentario	<!-- Este es un comentario HTML -->
9	Documento	Documento	<html>
10	Tipo de documento	Tipo de documento	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

Los **nodos de elemento** son aquellos que están formados por una etiqueta HTML. Por ejemplo: <a> … , … , … , <img>, etc. Independientemente de si es un elemento abierto o cerrado (con etiqueta de cierre, además de apertura).

Los **nodos de texto** son los fragmentos de texto que se incluyen en una página web. Para que sean correctamente tratados deben de estar incluidos dentro de un nodo de elemento. Por ejemplo: Esto es un nodo de texto . El fragmento “Esto es un nodo de texto” constituye un nodo de texto.

Jerarquías entre nodos

Nodo padre: es aquel nodo que tiene otros nodos bajo de él. Un nodo de texto no puede ser nodo padre, puesto que jerárquicamente, de un fragmento de texto no puede depender ningún elemento de la página. Los nodos de elemento sólo pueden ser nodos padre si se tratan de elementos con etiquetas de cierre. Un elemento vacío, por tanto, no puede ser nodo padre.

Nodo hijo: son los nodos que están por debajo de otros nodos en un diagrama de árbol de documento. Un nodo determinado puede poseer varios nodos hijo.

Nodo hermano: Son los nodos que están al mismo nivel de jerarquía que otro nodo determinado. Es decir, ambos nodos poseen el mismo nodo padre.

Dentro de un documento los nodos forman matrices, denominadas genéricamente **childNodes**. Así, el objeto document tiene una matriz childNodes con un único elemento (**html**). Éste, a su vez, posee una matriz childNodes con dos elementos (**head** y **body**). Sucesivamente, el nodo head, posee normalmente su propia matriz childNodes, donde generalmente el elemento **title** es uno de sus hijos.

Algo muy importante a tener en cuenta es que para hacer referencia a cualquier nodo, es imprescindible que éste haya sido cargado en la ventana del navegador. Por lo tanto, será importante que el código que hace referencia a nodos esté presente al final del elemento body. De lo contrario, tendrá que colocar el código dentro de funciones que deberán ser invocadas mediante el manejador de evento onload asociado al elemento body del documento.

Propiedades y métodos de los nodos

La siguiente tabla resume las principales propiedades y métodos de los nodos definidos por el estándar W3C:

Propiedad o método	Utilidad
className	Indica o establece el origen de clase CSS que afecta al nodo referido.
firstChild	Se refiere al primer nodo hijo de aquél con el que estemos trabajando.
lastChild	Se refiere al último nodo hijo de aquél con el que estemos trabajando.
nextSibling	Se refiere al nodo hermano siguiente de aquél con el que estemos trabajando.
nodeName	Se refiere al nombre que identifica al nodo actual.
nodeType	Se refiere al tipo de nodo (elemento, atributo, comentario, texto, etc.)
nodeValue	El texto que constituye un nodo. Aplicable a nodos de texto. Esta propiedad es null para un nodo de tipo elemento.
ownerDocument	Se refiere al documento propietario de aquél con el que estamos trabajando.
parentNode	Se refiere al nodo padre de aquél con el que estamos trabajando.
previousSibling	Se refiere al nodo hermano anterior de aquél con el que se está trabajando.
tagName	El nombre de la etiqueta de un nodo.
appendChild()	Añade un nodo hijo a aquél nodo con el que se está trabajando.
cloneNode()	Copia un nodo.
createElement()	Crea un nodo de tipo elemento para añadirlo como hijo, al nodo con el que se está trabajando.
createTextNode()	Crea un nodo de texto para añadirlo como hijo, al nodo con el que se está trabajando.
getAttribute()	Obtiene el valor de un atributo.
getElementById()	Se usa para referirse a un nodo mediante su valor de id.
getElementsByTagName()	Permite referirse a un nodo (o conjunto de nodos) por el nombre de la etiqueta.
hasChildNodes()	Determina si un nodo tiene hijos.
insertBefore()	Inserta un nodo hijo en la matriz childNodes de aquél con el que estamos trabajando.
removeAttribute()	Elimina un atributo de un nodo de un elemento.
removeChild()	Elimina el hijo indicado en la matriz childNodes del nodo con el que se está trabajando.
replaceChild()	Sustituye el hijo indicado de la matriz childNodes del nodo con el que se está trabajando por otro.
setAttribute()	Establece un atributo del nodo elemento con el que se está trabajando y su valor.

El objeto Document

El objeto Document posee varios (sub)objetos, a veces en forma de colecciones, además de métodos que tienen por propósito posibilitar la manipulación de los elementos presentes en la página web.

Entre las colecciones del objeto Document se pueden mencionar: forms[], links[], images[], anchors[]. Casi todos los objetos de JavaScript del DOM poseen además del objeto una colección correspondiente. Los métodos más importantes del objeto Document son:

1. getElementById(id).

Devuelve el elemento con el identificador id. Si no existe, devuelve NULL.

2. getElementsByTagName(tag).
   Devuelve un objeto NodeList de todos los elementos en el documento con esta etiqueta tag.

Estos métodos permiten el acceso directo a un elemento mediante su id o a una lista o colección de nodos con el mismo nombre.

Otros métodos del objeto Document son:

1. createElement(tag), que crea un elemento dentro de la estructura del documento. El argumento tag, debe ser el nombre de un elemento HTML válido.
2. createTextNode(text), que crea un nodo de texto con el texto text que luego debería ser agregado a un nodo de tipo elemento.

El subobjeto Element

El tipo de nodo Element amplía el objeto Node con varios métodos y propiedades. Puede poseer diversos tipos de nodos hijos, como: Element, Text, Comment, etc.
Además posee una serie de métodos, como los que se describen a continuación:

1. getAttribute(att) que devuelve el valor del atributo att aplicado sobre el elemento o una cadena vacía si este atributo no tiene ningún valor.
2. setAttribute(att, val) que asigna el valor val al atributo att. Si el atributo no existiese, se crea uno nuevo con el nombre indicado.
3. removeAttribute(att) que elimina el atributo att sobre el elemento aplicado.

III. MATERIALES Y EQUIPO

Para la realización de la guía de práctica se requerirá lo siguiente:

No.	Requerimiento	Cantidad
1	Guía de práctica: El Modelo de Objetos de Documento	1
2	Computadora con editor de HTML y navegadores instalados	1

IV. PROCEDIMIENTO

Ejercicio #1: El siguiente ejemplo, muestra cómo se puede acceder a varias propiedades y funciones del DOM que nos permiten resalta, modificar y eliminar elementos del documento de forma dinámica dentro de la página web.

Archivo 1: dom.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Funcionalidad básica del DOM</title>
    <meta charset="utf-8" /> <link
    rel="stylesheet"
href="http://fonts.googleapis.com/css?family=Oswald" />
    <link rel="stylesheet" href="css/fonts.css" />
    <link rel="stylesheet" href="css/dom.css" />
```

```
    <script src="js/dom.js"></script>
</head>
<body>
<header id="encabezado">
    <h1 id="encabezadogrande" class="highlighted">
        [Encabezadogrande]<br /> Modelo de objetos DHTML
    </h1>
    <h3 id="encabezadochico">
        [Encabezadochico] Funcionalidad de elementos
    </h3>
</header>
<section>
<article id="parrafos">
    <p id="para1">
        [Para1] El Modelo de Objetos de Documento (DOM) permite
        el acceso rápido y dinámico a todos los elementos de HTML5
        para manipularlos con JavaScript.
    </p>
    <p id="para2">
        [Para2] Para mayor información, revise la sección "JavaScript
        y el DOM" del <a href="http://www.deitel.com/javascript"
id="link">
        [link] Centro de recursos de JavaScript</a> de Deitel.
    </p>
    <p id="para3">
        [Para3] Los siguientes botones demuestran: (lista)
    </p>
    <ul id="lista">
        <li id="elemento1">[Elemento1] getElementById y parentNode.</li>
        <li id="elemento2">[Elemento2] insertBefore y appendChild.</li>
        <li id="elemento3">[Elemento3] replaceChilde y removeChild.</li>
    </ul>
</article>
<article>
    <div id="nav" class="nav">
        <form onsubmit="Regresa falso" action="#">
            <p>
                <input type="text" id="gbi" value="encabezadogrande"
                /> <input type="reset" value="Obtener por id"
id="botonPorId" />
            </p>
            <p>
                <input type="text" id="ins" />
                <input type="reset" id="botonInsertar" value="Insertar
antes" />
            </p>
            <p>
                <input type="text" id="adjuntar" />
                <input type="reset" value="Adjuntar hijo"
id="botonAdjuntar" />
            </p>
            <p>
```

```
                <input type="text" id="reemplazar" />
                <input type="reset" value="Reemplazar actual"
id="botonReemplazar" />
            </p><br />
            <p>
                <input type="button" value="Eliminar actual"
id="botonEliminar" />
            </p>
            <p>
                <input type="button" value="Obtener padre"
id="botonPadre" />
            </p>
        </form>
    </div>
</article>
</section>
</body>
</html>
```

Archivo 2: dom.js

```
//Secuencia de comandos para demostrar la funcionalidad
//básica del DOM (Document Object Model)
//Variable que almacena el nodo actual resaltado
var nodoActual;
//Se usa para asignar un id único a los nuevos
elementos var cuentaid = 0;

//Registar manejadores de eventos e inicializar nodoActual
function iniciar(){
    document.getElementById("botonPorId").addEventListener("click",
porId, false);
    document.getElementById("botonInsertar").addEventListener("click",
insertar, false);
    document.getElementById("botonAdjuntar").addEventListener("click",
adjuntarNodo, false);
    document.getElementById("botonReemplazar").addEventListener("click",
reemplazarActual, false);
    document.getElementById("botonEliminar").addEventListener("click",
eliminar, false);
    document.getElementById("botonPadre").addEventListener("click",
padre, false);
    //Inicializar nodo actual
    nodoActual = document.getElementById("encabezadogrande");
}

if(window.addEventListener){
    window.addEventListener("load", iniciar, false);
}
else if(window.attachEvent){
    window.attachEvent("onload", iniciar);
}
```

```
//Obtener y resaltar un elemento en base a su atributo id
function porId(){
    var id = document.getElementById("gbi").value;
    var destino = document.getElementById(id);
    if(destino){
        cambiarA(destino);
    }
}

//Insertar un elemento párrafo antes del elemento actual
//usando el método insertBefore
function insertar(){
    var nuevoNodo = crearNuevoNodo(
      document.getElementById("ins").value
    );
    nodoActual.parentNode.insertBefore(nuevoNodo, nodoActual);
    cambiarA(nuevoNodo);
}

//Adjuntar un nuevo párrafo como hijo del nodo
actual function adjuntarNodo() {
    var nuevoNodo = crearNuevoNodo(
        document.getElementById("adjuntar").value
    );
    nodoActual.appendChild(nuevoNodo);
    cambiarA(nuevoNodo);
}

//Reemplazar el nodo actual selecconado por un nodo
párrafo function reemplazarActual(){
    var nuevoNodo = crearNuevoNodo(
        document.getElementById("reemplazar").value
    );
    nodoActual.parentNode.replaceChild(nuevoNodo, nodoActual);
    cambiarA(nuevoNodo);
}

//Eliminar el nodo actual
function eliminar(){
    if(nodoActual.parentNode == document.body){
        alert("No se puede eliminar un elemento de nivel superior.");
    }
    else{
        var nodoAnterior = nodoActual;
        cambiarA(nodoAnterior.parentNode);
        nodoActual.removeChild(nodoAnterior);
    }
}

//Obtener y resaltar el padre del nodo
actual function padre(){
```

```
    var destino = nodoActual.parentNode;
    if(destino != document.body){
        cambiarA(destino);
    }
    else{
        alert("No hay padre.");
    }
}

//Función ayudante que devuelve un nuevo nodo párrafo
//que contiene un id único y texto dado
function crearNuevoNodo(texto){
    var nuevoNodo = document.createElement("p");
    idNodo = "nuevo" + cuentaid;
    ++cuentaid;
    //Establecer id del nuevoNodo
    nuevoNodo.setAttribute("id", idNodo);
    texto = "[" + idNodo + "]" + texto;
    nuevoNodo.appendChild(document.createTextNode(texto));
    return nuevoNodo;
}

//Función ayudante que cambia a un nuevo nodoActual
function cambiarA(nuevoNodo){
    //Elimina el resaltado anterior
    nodoActual.setAttribute("class", "");
    nodoActual = nuevoNodo;
    //Resaltar al nuevo nodo actual
    nodoActual.setAttribute("class", "highlighted");
    document.getElementById("gbi").value = nodoActual.getAttribute("id");
}
```

Archivo 3: dom.css

```
/* CSS para dom.html */
* {
    margin: 0;
    padding: 0;
}

html {
    height: 100%;
}

body {
    background: rgba(226,226,226,1);
    background: -moz-linear-gradient(top, rgba(226,226,226,1) 0%,
rgba(209,209,209,1) 51%, rgba(219,219,219,1) 100%);
    background: -webkit-gradient(left top, left bottom, color-stop(0%,
rgba(226,226,226,1)), color-stop(51%, rgba(209,209,209,1)),
color-stop(100%, rgba(219,219,219,1)));
```

```
    background: -webkit-linear-gradient(top, rgba(226,226,226,1) 0%,
rgba(209,209,209,1) 51%, rgba(219,219,219,1) 100%);
    background: -o-linear-gradient(top, rgba(226,226,226,1) 0%,
rgba(209,209,209,1) 51%, rgba(219,219,219,1) 100%);
    background: -ms-linear-gradient(top, rgba(226,226,226,1) 0%,
rgba(209,209,209,1) 51%, rgba(219,219,219,1) 100%);
    background: linear-gradient(to bottom, rgba(226,226,226,1) 0%,
rgba(209,209,209,1) 51%, rgba(219,219,219,1) 100%);
    filter: progid:DXImageTransform.Microsoft.gradient(
startColorstr='#e2e2e2', endColorstr='#dbdbdb', GradientType=0 );
    font-size: 16px;
    height: 100%;
}

header h1 {
    text-align: center;
    margin: 20px auto;
    font-family: "Museo300-Regular", Courier;
    font-size: 4em;
    text-transform: uppercase;
    color: #707070;
    text-shadow: 5px 5px 0px #eee,
                 7px 7px 0px #707070;
}

header h3 {
    font-family: "Museo300-Regular", Courier;
    margin: 12px 5%;
    text-align: left;
    font-size: 2.4em;
    text-transform: uppercase;
    color: #222;
    text-shadow: 0px 2px 3px #666;
}

header h5 {
    font-family: Tahoma,Helvetica,Arial,"Liberation Sans";
    font-size: 1.8em;
    text-align: center;
    color: #7C93BA;
    text-shadow: 0 2px 3px #87CEFA;
}

p {
    margin-left: 5%;
    margin-right: 5%;
    font-family: 'Museo300-Regular';
    font-size: 0.9em;
}

ul {
    margin-left: 10%;
```

```
}

a {
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}

.highlighted {
    background-color: Gold;
}

.nav {
    border-top: 3px dashed blue;
    padding-top: 10px;
    width: 100%;
}

input[type="text"] {
    background: #ffd;
    color: #35b128;
    font-size: 1.1em;
    height: 1.8em;
    padding: 2.5px 4px;
    width: 200px;
}

input[type="reset"] {
    border: 2px groove #7c93ba;
    /* Forzar al cambio de puntero del ratón cuando se posicione encima
*/
    cursor:pointer;
    padding: 5px 25px;
    /* Crear efecto de gradiente para el fondo */
    /* Color de fondo sólido para navegadores que no soporten gradientes
*/
    background-color: #6b6dbb;
    background: -webkit-gradient(linear, left top, left bottom,
from(#88add7), to(#6b6dbb));
    background: -webkit-linear-gradient(top, #88add7, #6b6dbb);
    background: -moz-linear-gradient(top, #88add7, #6b6dbb);
    background: -o-linear-gradient(top, #88add7, #6b6dbb);
    background: linear-gradient(top, #88add7, #6b6dbb);
    /* Estilo para el texto dentro del botón */
    font-family: Andika, Arial, sans-serif;
    color:#fff;
    font-size:1.1em;
    letter-spacing:.1em;
    font-variant:small-caps;
    /* Efecto de esquinas redondeadas */
```

```
    -webkit-border-radius: 0 15px 15px 0;
    -moz-border-radius: 0 15px 15px 0;
    -o-border-radius: 0 15px 15px 0;
    -ms-border-radius: 0 15px 15px 0;
    border-radius: 0 15px 15px 0;
    /* Agregar sombra al botón */
    -webkit-box-shadow: rgba(0,0,0,0.75) 0 2px 6px;
    -moz-box-shadow: rgba(0,0,0,0.75) 0 2px 6px;
    -o-box-shadow: rgba(0,0,0,0.75) 0 2px 6px;
    -ms-box-shadow: rgba(0,0,0,0.75) 0 2px 6px;
    box-shadow: rgba(0,0,0,0.75) 0 2px 6px;
}

input[type="reset"]:hover,
input[type="reset"]:focus {
    color:#edebda;
    /* Reduce the spread of the shadow to give a pushed effect
    */ -webkit-box-shadow: rgba(0,0,0,0.25) 0 1px 0px; -moz-box-
    shadow: rgba(0,0,0,0.25) 0 1px 0px; -o-box-shadow:
    rgba(0,0,0,0.25) 0 1px 0px; -ms-box-shadow: rgba(0,0,0,0.25)
    0 1px 0px; box-shadow: rgba(0,0,0,0.25) 0 1px 0px;

}

input[type="button"] {
    /* Forzar al cambio de puntero del ratón cuando se posicione encima
*/
    cursor:pointer;
    /* Agregar relleno dentro del botón */
    padding:5px 25px;
    /* Color del botón */
    background:#35b128;
    /* Color para el borde */
    border:1px solid #33842a;
    /* Crear efecto de esquinas redondeadas para el botón */
    -moz-border-radius: 10px;
    -webkit-border-radius: 10px;
    -o-border-radius: 10px;
    -ms-border-radius: 10px;
    border-radius: 10px;
    /* Crear efecto de sombre en el botón */
    -webkit-box-shadow: 0 0 4px rgba(0,0,0, .75);
    -moz-box-shadow: 0 0 4px rgba(0,0,0, .75);
    box-shadow: 0 0 4px rgba(0,0,0, .75);
    /* Estilo del texto */
    color:#f3f3f3;
}   font-size:1.1em;


input[type="button"]:hover,
input[type="button"]:focus {
    /* Hacer el fondo más oscuro */
```

```
    background-color :#399630;
    /* Reducir el tamaño de la sombra para darle el efecto de botón
presionado */
    -webkit-box-shadow: 0 0 1px rgba(0,0,0, .75);
    -moz-box-shadow: 0 0 1px rgba(0,0,0, .75);
    -o-box-shadow: 0 0 1px rgba(0,0,0, .75);
    -ms-box-shadow: 0 0 1px rgba(0,0,0, .75);
    box-shadow: 0 0 1px rgba(0,0,0, .75);
}
```

Archivo 4: fonts.css

```
@font-face {
    font-family: 'Museo300-Regular';
    src: url('../fonts/Museo300-Regular.eot');
    src: url('../fonts/Museo300-Regular.eot?#iefix') format('embedded-
opentype'),
         url('../fonts/Museo300-Regular.woff') format('woff'),
         url('../fonts/Museo300-Regular.ttf') format('truetype'),
         url('../fonts/Museo300-Regular.svg#MyWebfont') format('svg');
    font-weight: normal;
    font-style: normal;
}
```

Resultado:

Ejercicio #2: El siguiente ejemplo, muestra cómo manipular mediante el DOM campos de un formulario para ordenar una pizza y obtener el total a pagar luego que el usuario selecciona el tamaño e ingredientes que desea para la pizza.

Archivo 1: orderpizza.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Ordenando una pizza</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="css/pizza.css" />
</head>
<body>
<header id="box">
```

```
    <h1 id="flashlight">
        <span id="flash">ORDEN DE</span>
        <span id="light">PIZZA</span>
    </h1>
</header>
<form action="javascript:void(0);" class="demoForm" id="pizzaForm">
    <fieldset>
    <legend>Tamaño e ingredientes</legend>
    <p>Size:
        <label><input type="radio" name="size" value="7.50"
checked="checked" />Pequeña</label>
        <label><input type="radio" name="size" value="12"
/>Mediana</label>
        <label><input type="radio" name="size" value="16"
/>Gigante</label>
        <input type="hidden" name="sz_tot" value="0.00" />
    </p>
    <p id="pizza_toppings"><label>Ingredientes:</label>
        <label><input type="checkbox" name="mushrooms" value="1.25"
/>Hongos</label>
        <label><input type="checkbox" name="onions" value="0.75"
/>Cebolla</label>
        <label><input type="checkbox" name="black olives" value="0.50"
/>Aceitunas</label>
        <label><input type="checkbox" name="ham" value="1.50"
/>Jamón</label>
        <label><input type="checkbox" name="pepperoni" value="1.50"
/>Pepperoni</label>
        <input type="hidden" name="tops_tot" value="0.00" />
    </p>
    <p>
        <label>
            Total: $<input type="text" name="total" class="num"
value="0.00" readonly="readonly" />
        </label>
    </p>
    <p>
        <input type="submit" name="submit" value="Submit" />
    </p>
    </fieldset>
</form>
<script src="js/pizza.js"></script>
</body>
</html>
```

Archivo 2: pizza.js

```
//Redondeando el precio a mostrar a dos cifras decimales
function formatDecimal(val, n) {
    n = n || 2;
    var str = "" + Math.round (parseFloat(val) * Math.pow(10,
    n)); while (str.length <= n) {
```

```
        str = "0" + str;
    }
    var pt = str.length - n;
    return str.slice(0,pt) + "." + str.slice(pt);
}

function getRadioVal(form, name) {
var radios = form.elements[name];
var val;
    for (var i=0, len=radios.length; i<len; i++)
        { if (radios[i].checked == true) {
            val = radios[i].value;
            break;
        }
    }
    return val;
}

//Calcula el subtotal de ingredientes seleccionados
function getToppingsTotal(e) {
    var form = this.form;
    var val = parseFloat(form.elements['tops_tot'].value);
    if ( this.checked == true ) {
        val += parseFloat(this.value);
    } else {
        val -= parseFloat(this.value);
    }

    form.elements['tops_tot'].value = formatDecimal(val);
    updatePizzaTotal(form);
}

//Obtiene el subtotal del valor de la pizza de acuerdo al tamaño
function getSizePrice(e) {
    this.form.elements['sz_tot'].value = parseFloat(this.value);
    updatePizzaTotal(this.form);
}

//Calcula el precio total a cancelar por la pizza tomando en cuenta
//los subtotales de acuerdo al tamaño y a los ingredientes seleccionados
function updatePizzaTotal(form) {
    var sz_tot = parseFloat(form.elements['sz_tot'].value); var
    tops_tot = parseFloat(form.elements['tops_tot'].value);
    form.elements['total'].value = formatDecimal(sz_tot + tops_tot);
}


(function() {
    var form = document.getElementById('pizzaForm'); var
    el = document.getElementById('pizza_toppings');
```

```
    // Determinar los ingredientes seleccionados en las casillas de
verificación
    var tops = el.getElementsByTagName('input');
    for (var i=0, len=tops.length; i<len; i++) {
        if (tops[i].type === 'checkbox') {
            tops[i].onclick = getToppingsTotal;
        }
    }

    var sz = form.elements['size'];
    for (var i=0, len=sz.length; i<len; i++) {
        sz[i].onclick = getSizePrice;
    }

    // set sz_tot to value of selected
    form.elements['sz_tot'].value =
formatDecimal(parseFloat(getRadioVal(form, 'size')));
    updatePizzaTotal(form);
})();
```

Archivo 3: pizza.css

```
/* Estilos para el formulario de orden de pizza */
body {
    font: 16px/1.4 Tahoma,Helvetica,Kalimati,Sans-serif;
}

/* box ---------------------------------------------------- */
#box {
  background: hsl(210, 30%, 0%) radial-gradient( hsl(210, 30%, 20%),
hsl(210, 30%, 0%));
  font-weight: bold;
  min-width: 500px;
  padding: 20px auto;
  text-align: center;
  -webkit-backface-visibility: hidden; /* fixes flashing */
}


/* flashlight --------------------------------------------------- */
#flashlight {
    color: hsla(0,0%,0%,0);
    font-size: 3em;
    perspective: 80px;
    outline: none;
}


/* flash -------------------------------------------------- */
#flash {
  display: inline-block;
  text-shadow: #bbb 0 0 1px,
```

```
                #fff 0 -1px 2px,
                #fff 0 -3px 2px,
                rgba(0,0,0,0.8) 0 30px 25px;
  transition: margin-left 0.3s cubic-bezier(0, 1, 0, 1);
}

#box:hover #flash {
  margin-left: 20px;
  transition: margin-left 1s cubic-bezier(0, 0.75, 0, 1);
}


/* light ------------------------------------------------------ */
#light {
  display: inline-block;
  text-shadow: #111 0 0 1px, rgba(255,255,255,0.1) 0 1px 3px;
}

#box:hover #light {
  text-shadow: #fff 0 0 4px, #fcffbb 0 0 20px;
  transform: rotateY(-60deg);
  transition: transform 0.3s cubic-bezier(0, 0.75, 0, 1), text-shadow
0.1s ease-out;
}

ul {
    margin:1.4em 0 2em;
}

ul li {
    margin-bottom:.8em;
}

form.demoForm fieldset {
    border: 1px solid #ddd;
    margin: 0 auto;
    width: 60%;
}

form.demoForm p {
    line-height: 1.1;
    padding:0 6px;
}
form.demoForm legend {
    margin-left:6px;
}

form.demoForm label {
    margin-right: .8em;
}
form.demoForm input {
    vertical-align: bottom;
```

```
    padding:0;
    margin:0; /* needed for checkboxes */
}
form.demoForm input.num {
    text-align: right;
    width: 50px;
}

/* for ie8+ (wraps tightly) */
form.demoForm input[type=submit] {
    padding:1px 4px;
}
```

Resultado:

Ejercicio #3: El siguiente ejemplo le permite ingresar contenido en una sección de la página, inicialmente invisible, llenando un área de texto que al presionar el botón Enviar del formulario lo va anexando al área de contenido, que primero debe hacerse visible. Se pueden crear nuevos párrafos, eliminarlos o indicar si se insertan antes de párrafos ya existentes.

Archivo 1: nodos.html

```
<!DOCTYPE html>
<html lang="es">
<head>
   <title>Creación y eliminación de nodos</title>
   <meta charset="utf-8" />
   <link rel="stylesheet" href="css/fonts.css" />
   <link rel="stylesheet" href="css/estilosnode.css" />
   <script src="js/nodos.js"></script>
</head>
<body>
<header>
    <h1>Creación y eliminación de nodos</h1>
</header>
<div>
<form action="javascript:void(0);">
<fieldset id="nodeform">
    <legend><span>Manejo de nodos con el DOM</span></legend>
    <label for="textarea" class="txa">Introduzca texto:</label>
```

```
    <textarea name="textArea" id="textArea" class="textareabox" rows="8"
cols="45"></textarea><br />
    <input type="radio" name="nodeAction" id="nodeAction" />
    <label for="nodeAction">Añadir nodo</label>
    <input type="radio" name="nodeAction" id="nodeAction" />
    <label for="nodeAction">Eliminar nodo</label>
    <input type="radio" name="nodeAction" id="nodeAction" />
    <label for="nodeAction">Insertar antes de un nodo</label><br />
    <label for="grafCount">Párrafo número:</label>
    <select name="grafCount" id="grafCount"><br />
    </select><br />
    <input type="submit" value="Enviar" class="boton" />
</fieldset>
</form>
</div>
<div id="modificable">
</div>
</body>
</html>
```

Archivo 2: nodos.js

```
window.onload = initAll;
var nodeChangingArea;

function initAll(){
     document.getElementsByTagName("form")[0].onsubmit = function(){
          return nodeChanger();
     }
     nodeChangingArea = document.getElementById("modificable");
}

function addNode(){
     var inText = document.getElementById("textArea").value;
     var newText = document.createTextNode(inText); var
     newGraf = document.createElement("p");
     newGraf.appendChild(newText);
     nodeChangingArea.appendChild(newGraf);
}

function delNode(){
     var delChoice =
     document.getElementById("grafCount").selectedIndex; var allGrafs =
     nodeChangingArea.getElementsByTagName("p"); var killGraf =
     allGrafs.item(delChoice); nodeChangingArea.removeChild(killGraf);
}

function insertNode(){
     var inChoice = document.getElementById("grafCount").selectedIndex;
     var inText = document.getElementById("textArea").value;
     var newText = document.createTextNode(inText);
     var newGraf = document.createElement("p");
```

```
     newGraf.appendChild(newText);
     var allGrafs = nodeChangingArea.getElementsByTagName("p");
     var oldGraf = allGrafs.item(inChoice);
     nodeChangingArea.insertBefore(newGraf, oldGraf);
}

function replaceNode(){
     var inChoice = document.getElementById("grafCount").selectedIndex;
     var inText = document.getElementById("textArea").value;
     var newText = document.createTextNode(inText);
     var newGraf = document.createElement("p");
     newGraf.appendChild(newText);
     var allGrafs = nodeChangingArea.getElementsByTagName("p");
     var oldGrafs = allGrafs.item(inChoice);
     nodeChangingArea.replaceChild(newGraf, oldGraf);
}

function nodeChanger(){
     var actionType = -1;
     var currentPgraphCount =
nodeChangingArea.getElementsByTagName("p").length;
     var radioButtonSet =
 document.getElementsByTagName("form")[0].nodeAction;
        nodeChangingArea.style.visibility = "visible";
     for(var i=0; i<radioButtonSet.length; i++){
          if(radioButtonSet[i].checked){
             actionType = i;
          }
     }
     switch(actionType){
          case 0:
               addNode();
               break;
          case 1:
               if(currentPgraphCount > 0){
                  delNode();
                  break;
               }
          case 2:
               if(currentPgraphCount > 0){
                  insertNode();
                  break;
               }
          case 3:
               if(currentPgraphCount > 0){
                  replaceNode();
                  break;
               }
          default:
               alert("No ha elegido una opción válida");
     }
     document.getElementById("grafCount").options.length = 0;
```

```
      for(i=0; i<nodeChangingArea.getElementsByTagName("p").length; i++){
           document.getElementById("grafCount").options[i] = new
Option(i+1);
      }
      return false;
}
```

Archivo 3: estilosnode.css

```
*               {
    margin: 0;
     padding: 0;
}

*:focus         {
      outline:none;
}

html, body      {
    height: 100%;
}

body            {
      background-color:rgb(255,215,235);
      background-
      image:url(../img/literatura.jpg);
      background-position:right 68px; background-
      repeat:no-repeat; font-size: 16px;
}

header h1             {
    color:Purple;
    font-family:'GentiumRegular',Geneva,sans-serif;
    font-weight:bold;
    font-size:3.5em;
    padding: 12px;
    text-align:center;
    text-shadow:0 1px  0 #ccc,
                02px  0 #c9c9c9,
                03px  0 #bbb,
                04px  0 #b9b9b9,
                05px  0 #aaa,
                06px 1px rgba(0,0,0,.1),
                00 5px rgba(0,0,0,.1),
                01px 3px rgba(0,0,0,.3),
                03px 5px rgba(0,0,0,.2),
                05px 10px rgba(0,0,0,.25),
                010px 10px rgba(0,0,0,.2),
                020px 20px rgba(0,0,0,.15);
}

fieldset#nodeform {
```

```
    background:rgb(244,244,244);
    background:rgba(244,244,244,0.7);
    -webkit-border-radius: 25px;
    -moz-border-radius: 25px;
    -o-border-radius: 25px;
    -ms-border-radius: 25px;
    border-radius: 25px;
    border:3px double #999;
    margin:16px auto;
    padding:0 50px 50px;
    width:450px;
}

legend       {
     background-color: #d9dad9;
     -webkit-border-radius: 15px;
    -moz-border-radius: 15px;
    -o-border-radius: 15px;
    -ms-border-radius: 15px;
    border-radius: 15px;
     border:1px solid #333;
     font-size:1em;
     font-weight:bold;
     color:Purple;
    margin-left:-20px;
     text-align:left;
     padding:0 10px;
     text-shadow:1px 1px 0 #ccc;
}

label        {
     font-size:1em;
     font-weight:bold;
     margin:17px 0 7px;
     text-align:left;
     text-shadow:1px 1px 0 #fff;
}

textarea     {
     resize:both;
     max-width:450px;
}

input.textbox,
 #nodeform .textareabox {
     /* Saf4+, Chrome */
     background:-webkit-gradient(linear, left top, left bottom,
from(#E3E3E3), to(#FFFFFF));
     /* Chrome 10+, Saf5.1+ */
     background:-webkit-linear-gradient(top, #E3E3E3, #FFFFFF);
     /* FF3.6+ */
     background:-moz-linear-gradient(top, #E3E3E3, #FFFFFF);
```

```
     /* IE10 */
     background:-ms-linear-gradient(top, #E3E3E3, #FFFFFF);
     /* Opera 11.10+ */
     background:-o-linear-gradient(top, #E3E3E3, #FFFFFF);
     background:url(../img/required.png) no-repeat 98% 4% #F0F0EF;
     -webkit-border-radius: 15px;
    -moz-border-radius: 15px;
    -o-border-radius: 15px;
    -ms-border-radius: 15px;
    border-radius: 15px;
     border:1px solid #fff;
     -moz-box-shadow: 0px 2px 0px #999;
     -webkit-box-shadow: 0px 2px 0px #999;
     -o-box-shadow: 0px 2px 0px #999;
     -ms-box-shadow: 0px 2px 0px #999;
    box-shadow: 0px 2px 0px #999;
     padding:5px 10px;
     text-shadow:1px 1px 1px #777;
     -webkit-transition: all 0.5s ease-in-out;
     -moz-transition: all 0.5s ease-in-out;
     -o-transition: all 0.5s ease-in-out;
     -ms-transition: all 0.5s ease-in-out;
     transition: all 0.5s ease-in-out;
     width:400px;
}

input.textbox:focus,
#nodeform .textareabox:focus {
     -moz-box-shadow: 5px 3px 1px #ccc;
     -webkit-box-shadow: 5px 3px 1px #ccc;
     -o-box-shadow: 7px 7px 2px #ccc;
     -ms-box-shadow: 7px 7px 2px #ccc;
    box-shadow: 7px 7px 2px #ccc;
     -webkit-transform: scale(1.05);
     -moz-transform: scale(1.05);
     -o-transform: scale(1.05);
     -ms-transform: scale(1.05);
     transform: scale(1.05);
     text-shadow:1px 1px 3px #777;
}

input.textbox:required,
#nodeform .textareabox:required {
     background:url(../img/required.png) no-repeat 200px 5px #F0F0EF;
     background:url(../img/required.png) no-repeat 200px 5px, -webkit-
gradient(linear, left top, left bottom, from(#E3E3E3), to(#FFFFFF)); /*
Saf4+, Chrome */
     background:url(../img/required.png) no-repeat 200px 5px, -webkit-
linear-gradient(top, #E3E3E3, #FFFFFF); /* Chrome 10+, Saf5.1+ */
background:url(../img/required.png) no-repeat 200px 5px, -moz-
linear-gradient(top, #E3E3E3, #FFFFFF); /* FF3.6+ */
```

```
     background:url(../img/required.png) no-repeat 200px 5px, -ms-linear-
gradient(top, #E3E3E3, #FFFFFF); /* IE10 */
     background:url(../img/required.png) no-repeat 200px 5px, -o-linear-
gradient(top, #E3E3E3, #FFFFFF); /* Opera 11.10+ */
}

input.textbox:required:valid,
#nodeform .textareabox:required:valid {
     background:url(../img/valid.png) no-repeat 200px 5px #F0F0EF;
     background:url(../img/valid.png) no-repeat 200px 5px, -webkit-
gradient(linear, left top, left bottom, from(#E3E3E3), to(#FFFFFF)); /*
Saf4+, Chrome */
     background:url(../img/valid.png) no-repeat 200px 5px, -webkit-
linear-gradient(top, #E3E3E3, #FFFFFF); /* Chrome 10+, Saf5.1+ */
     background:url(../img/valid.png) no-repeat 200px 5px, -moz-linear-
gradient(top, #E3E3E3, #FFFFFF); /* FF3.6+ */
     background:url(../img/valid.png) no-repeat 200px 5px, -ms-linear-
gradient(top, #E3E3E3, #FFFFFF); /* IE10 */
     background:url(../img/valid.png) no-repeat 200px 5px, -o-linear-
gradient(top, #E3E3E3, #FFFFFF); /* Opera 11.10+
*/ }

input.textbox:focus:invalid,
#nodeform .textareabox:focus:invalid,
input.textbox:not(:required):invalid,
#nodeform .textareabox:not(:required):invalid {
     background:url(../img/invalid.png) no-repeat 200px 5px #F0F0EF;
     background:url(../img/invalid.png) no-repeat 200px 5px, -webkit-
gradient(linear, left top, left bottom, from(#E3E3E3), to(#FFFFFF)); /*
Saf4+, Chrome */
     background:url(../img/invalid.png) no-repeat 200px 5px, -webkit-
linear-gradient(top, #E3E3E3, #FFFFFF); /* Chrome 10+, Saf5.1+ */
     background:url(../img/invalid.png) no-repeat 200px 5px, -moz-linear-
gradient(top, #E3E3E3, #FFFFFF); /* FF3.6+ */
     background:url(../img/invalid.png) no-repeat 200px 5px, -ms-linear-
gradient(top, #E3E3E3, #FFFFFF); /* IE10 */
     background:url(../img/invalid.png) no-repeat 200px 5px, -o-linear-
gradient(top, #E3E3E3, #FFFFFF); /* Opera 11.10+ */
}

input[type="submit"] {
     padding:10px;
     margin:0 10px !important;
     width:300px;
}

#modificable {
    background: rgba(254,254,254,1);
    background: -moz-linear-gradient(-45deg, rgba(254,254,254,1) 0%,
rgba(226,226,226,1) 0%, rgba(209,209,209,1) 51%,
rgba(219,219,219,1) 100%);
```

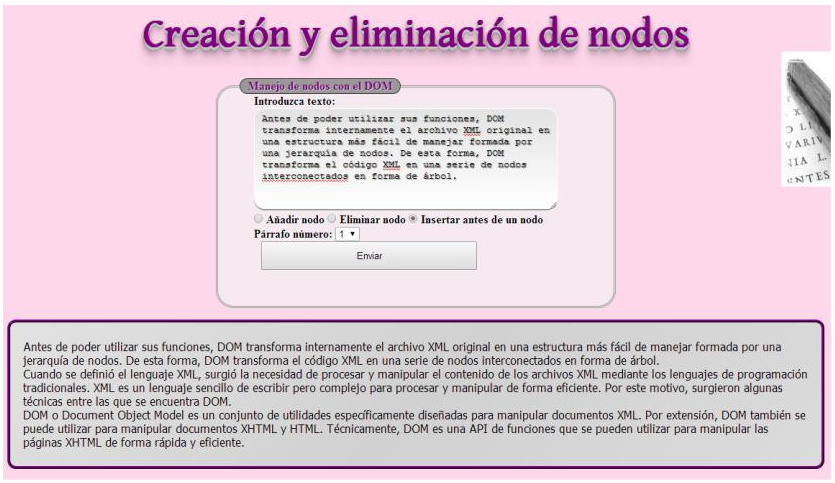
```
    background: -webkit-gradient(left top, right bottom, color-stop(0%,
rgba(254,254,254,1)), color-stop(0%, rgba(226,226,226,1)), color-
  stop(51%, rgba(209,209,209,1)), color-stop(100%, rgba(219,219,219,1)));
       background: -webkit-linear-gradient(-45deg, rgba(254,254,254,1) 0%,
rgba(226,226,226,1) 0%, rgba(209,209,209,1) 51%, rgba(219,219,219,1)
100%);
    background: -o-linear-gradient(-45deg, rgba(254,254,254,1) 0%,
rgba(226,226,226,1) 0%, rgba(209,209,209,1) 51%, rgba(219,219,219,1)
100%);
    background: -ms-linear-gradient(-45deg, rgba(254,254,254,1) 0%,
rgba(226,226,226,1) 0%, rgba(209,209,209,1) 51%, rgba(219,219,219,1)
100%);
    background: linear-gradient(135deg, rgba(254,254,254,1) 0%,
rgba(226,226,226,1) 0%, rgba(209,209,209,1) 51%, rgba(219,219,219,1)
100%);
    filter: progid:DXImageTransform.Microsoft.gradient(
startColorstr='#fefefe', endColorstr='#dbdbdb', GradientType=1 );
    border: 4px ridge Purple;
    -moz-border-radius: 12px;
    -webkit-border-radius: 12px;
    -o-border-radius: 12px;
    -ms-border-radius: 12px;
    border-radius: 12px;
    visibility: hidden;
    font: normal 1em Tahoma,Helvetica,"Liberation Sans";
    height: auto;
    margin: 0 auto;
    padding: 24px 18px;
    width: 80%;
}

#modificable p {
    font: 'GentiumRegular',Tahoma,Helvetica,"Liberation Sans";
    margin-bottom: 12px;
    text-align: justify;
}
```

Archivo 4: fonts.css

```
@font-face {
    font-family: 'GentiumRegular';
    src: url('../fonts/GentiumRegular.eot');
    src:    url('../fonts/GentiumRegular.eot?#iefix')    format('embedded-
opentype'),
         url('../fonts/GentiumRegular.woff') format('woff'),
         url('../fonts/GentiumRegular.ttf') format('truetype'),
         url('../fonts/GentiumRegular.svg#MyWebfont') format('svg');
    font-weight: normal;
    font-style: normal;
}
```

Resultado:



Ejercicio #4: El siguiente ejemplo, muestra la creación de una tabla a partir de un formulario donde se indica el número de filas y columnas de la tabla, así como si se incluye una fila de encabezado o no. Adicionalmente, para cada celda se crea un contenido por defecto.

Archivo 1: tabladom.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Creación de una tabla</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="css/modernform.css" />
    <link rel="stylesheet" href="css/table.css" />
    <script src="js/tabladom.js"></script>
</head>
<body>
<header>
    <h1>Creación de tablas dinámicamente</h1>
</header>
<section>
<article>
<div id="formholder">
    <ul class="tabs blue">
        <li>
            <input name="tabs blue" id="tab1" checked="" type="radio">
            <label for="tab1">Datos de la tabla</label>
            <div id="tab-content1" class="tab-content">
                <p>
                    Ingresa los datos para crear la tabla.
                </p>
                <form name="frmtable" id="frmtable">
                <!-- <form name="register" action=""> -->
<span class="tabaddon"><i class="fa fa-user fa- 2x"></i></span>
```

```
                    <input type="number" name="rows_text" id="rows_text"
required="required" pattern="\d{1,3}" placeholder="(Filas)" class="field"
/>
                    <span class="tabaddon"><i class="fa fa-envelope fa-
2x"></i></span>
                    <input type="number" name="columns_text"
id="columns_text" required="required" pattern="\d{1,3}"
placeholder="(Columnas)" class="field" />
                    <div class="accounttype">
                        <input type="checkbox" value="si" id="radioOne"
name="chkhead" checked="checked" />
                        <label for="radioOne" class="radio"
checked="">Incluir celda de encabezados</label>
                    </div>
                    <div class="btn">
                        <input type="button" name="btnSend" id="btnSend"
value="Crear tabla" />
                    </div>
          </form>
        </div>
        </li>
    </ul>
</div>
</article>
<article>
<div id="tableholder">
</div>
</article>
</section>
</body>
</html>
```

Archivo 2: tabladom.js

```
function init(){
    var boton = document.getElementById("btnSend");
    if(boton.addEventListener){
        boton.addEventListener("click", function(){
           create_table(document.frmtable);
        }, false);
    }
    else if(boton.attachEvent){
      boton.attachEvent("onclick", function(){
           create_table(this.form);
        });
    }
}

function create_table(current_form){
    //Si el navegador no soporta DOM retornar
    if(!document.getElementById){
        return;
```

```
    }
    //Obtener datos de filas y columnas
    var table_rows = current_form.rows_text.value;
    var table_columns = current_form.columns_text.value;
    //Obtener el elemento div donde se mostrará la tabla var
    table_div = document.getElementById("tableholder");
    //Crear y añadir el elemento table
    var table_node = document.createElement("table");
    table_div.appendChild(table_node); //Crear cabecera
    de la tabla con caption
    var table_caption = document.createElement("caption");
    table_node.appendChild(table_caption);
    //Crear y añadir el elemento tbody a la tabla var
    tbody_node = document.createElement("tbody");
    table_node.appendChild(tbody_node);
    //Ciclo o lazo para añadir todos los elementos tr para las filas
    for(var row_counter=1; row_counter<=table_rows; row_counter++){
        var tr_node, td_node;
        tr_node = document.createElement("tr");
        tbody_node.appendChild(tr_node);
        //Ciclo o lazo para añadir todos los elementos td
        //para las columnas de la fila actual
for(var col_counter=1; col_counter<=table_columns;
col_counter++){
            if(row_counter == 1 && document.frmtable.chkhead.checked){
                td_node = document.createElement("th");
            }
            else {
                td_node = document.createElement("td");
            }
            tr_node.appendChild(td_node);
            //Añadir el nodo de texto con el
            contenido //a la celda actual
var text_node = document.createTextNode("Celda(" +
row_counter + ", " + col_counter + ")");
            td_node.appendChild(text_node);
        }
    }
    //Añadir borde a la tabla
    table_node.border = 1;
    //Agregar nodo de texto a la cabecera de la tabla (caption) var
    text_node_caption = document.createTextNode("Nueva tabla");
    table_caption.appendChild(text_node_caption); //Asignar un id
    al cuerpo de la tabla
    tbody_node.id = "tablebody";
}

if(window.addEventListener){
    window.addEventListener("load", init, false);
}
else if(window.attachEvent){
    window.attachEvent("onload",
    init);}
```

Archivo 3: table.css

```
#tableholder {
    height: auto;
    margin: 24px auto;
    width: 100%;
}

table {
    border-collapse: separate;
    border-spacing: 0;
    position: absolute;
    table-layout: auto;
    top: 280px;
    width: 99%;
}

table caption {
    font: bold 2em/2 Tahoma,Helvetica,"Liberation Sans";
    color: MidnightBlue;
    background-color: #333;
    -webkit-background-clip: text;
    -moz-background-clip: text;
    -o-background-clip: text;
    -ms-background-clip: text;
    background-clip: text;
    text-shadow: 2px 3px 6px rgba(60,60,90,0.75);
    padding: 12px auto;
}

thead {
    background: #395870;
    color: #fff;
}

th {
    background: MidnightBlue;
    color: #fcfcf0;
    padding: 4px 3px;
    min-width: 19%;}

td {
    color: MidnightBlue;
    padding: 4px 3px;
    min-width: 19%;
}

tbody tr:nth-child(even) {
    background: #f6f6df;
}
```

```
tbody tr:nth-child(odd) {
    background: #ececcf;
}

td {
    border-bottom: 1px solid #cecfd5;
    border-right: 1px solid #cecfd5;
}

td:first-child {
    border-left: 1px solid #cecfd5;
}
```

Archivo 4: modernform.css

```
@import url(http://fonts.googleapis.com/css?family=Lato:300,400,700);
@import url(http://fonts.googleapis.com/css?family=Roboto:100);
@import url(http://fonts.googleapis.com/css?family=Raleway:100,200,300);
@import url(http://fonts.googleapis.com/css?family=Montserrat:400,700);

* {
    margin: 0;
    padding: 0;
    -moz-box-sizing: border-box;
    -webkit-box-sizing: border-box;
    box-sizing: border-box;
}

html, body {
    height: 100%;
}

body {
    background: #FFFFFF;
    color: #fff;
    font-family: 'Lato',Arial,sans-serif;
    text-align: left;
}

header {
    border-bottom: 1px solid #EDEFF0;
    color: #009EE0;
    font-family: 'Lato',Arial,sans-serif;
    height: 100px;
    line-height: 100px;
    margin: 0 auto;
    max-width: 100%;
    position: relative;
    text-align: center;
}

header h1 {
```

```
    color:#60cffc;
    font-size: 3em;
    font-weight: bold;
    line-height: 1.3;
    line-height:100px;
    text-align: justify;
    text-shadow:
      1px -1px 0 #767676,
      -1px 2px 1px #737272,
      -2px 4px 1px #767474,
      -3px 6px 1px #787777,
      -4px 8px 1px #7b7a7a,
      -5px 10px 1px #7f7d7d,
      -6px 12px 1px #828181,
      -7px 14px 1px #868585,
      -8px 16px 1px #8b8a89,
      -9px 18px 1px #8f8e8d,
      -10px 20px 1px #949392,
      -11px 22px 1px #999897,
      -12px 24px 1px #9e9c9c,
      -13px 26px 1px #a3a1a1,
      -14px 28px 1px #a8a6a6,
      -15px 30px 1px #adabab,
      -16px 32px 1px #b2b1b0,
      -17px 34px 1px #b7b6b5,
      -18px 36px 1px #bcbbba,
      -19px 38px 1px #c1bfbf,
      -20px 40px 1px #c6c4c4;
}

small {
    bottom: 25px;
    color: #EF0C72;
    font-size: 18px;
    font-weight: bold;
    position: relative;
}

.container {
    float: left;
    height: 481px;
    padding: 20px 0;
    width: 100%;
}

section {
    position: relative;
    height: 442px;
    margin: 12px auto;
    width: 96%;
}
```

```
article    {
    width: 100%;
}

@font-face {
  font-family: 'FontAwesome';
  src: url('../fonts/fontawesome-webfont.eot?v=4.0.3');
  src: url('../fonts/fontawesome-webfont.eot?#iefix&v=4.0.3')
format('embedded-opentype'),
       url('../fonts/fontawesome-webfont.woff?v=4.0.3') format('woff'),
       url('../fonts/fontawesome-webfont.ttf?v=4.0.3')
format('truetype'),
       url('../fonts/fontawesome-webfont.svg?v=4.0.3#fontawesomeregular')
format('svg');
  font-weight: normal;
  font-style: normal;
}

/* Anulando estilos de campos de formulario cuando se obtenga el foco
*/ :focus {outline:none}

.fa {
  display: inline-block;
  font-family: FontAwesome;
  font-style: normal;
  font-weight: normal;
  line-height: 1;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  cursor:pointer;
}

.fa:hover{
    -moz-transform:scale(1.2); /* Firefox */
    -ms-transform:scale(1.2); /* IE 9 */
    -o-transform:scale(1.2); /* Opera */
    -webkit-transform:scale(1.2); /* Safari and Chrome */
    transform:scale(1.2);
    -ms-transition: 0.5s;
    -moz-transition: 0.5s;
    -o-transition: 0.5s;
    -webkit-transition: 0.5s;
    transition: 0.5s;

}

/* makes the font 33% larger relative to the icon container */
.fa-2x {
    font-size: 20px;
    line-height:40px;
}
```

```
/* Font Awesome uses the Unicode Private Use Area (PUA) to ensure
screen readers do not read off random characters that represent icons
*/ .fa-user:before {
  content: "\f0ce";
}
.fa-envelope:before {
  content: "\f0db";
}
.fa-lock:before {
  content: "\f113";
}
.fa-facebook:before {
  content: "\f02e";
}
.fa-twitter:before {
  content: "\f099";
}
.fa-degree:before {
  content: "\f02e";
}

.tabs h1 {
    font-size: 32px;
  font-family: 'Roboto';
  margin-bottom:15px;

}
.tabs p {
    font-size: 14px;
    line-height: 20px;
    margin-bottom: 8px;
}

/* Each Tab CSS to make Radio button Work*/
/* Blue Color Tab - If you want to used blue tab,
*/ /* just used following css*/

/******************** START ********************/
.tabs.blue {
    float: left;
    list-style: none;
    position: relative;
    text-align: left;
    width: 394px;
}

.tabs.blue li {
    float: right;
    display: block;
}

.tabs.blue input[type="radio"] {
```

```
    left: -9999px;
    position: absolute;
    top: -9999px;
}

.tabs.blue label {
    display: block;
    padding: 14px 31px;
    font-size: 16px;
    font-weight: normal;
    cursor: pointer;
    position: relative;
    top: 5px;
    -moz-transition: all 0.2s ease-in-out;
    -o-transition: all 0.2s ease-in-out;
    -webkit-transition: all 0.2s ease-in-out;
    transition: all 0.2s ease-in-out;
}

.tabs.blue .tab-content {
  z-index: 2;
  display: none;
  overflow: hidden;
  width: 100%;
  font-size: 17px;
  line-height: 25px;
  padding: 15px;
  position: absolute;
  top: 53px;
  left: 0;
}

.tabs.blue [id^="tab"]:checked + label {
    padding-top: 17px;
    top: 5px;

}

.tabs.blue [id^="tab"]:checked ~ [id^="tab-content"]
    { display: block;
}

/* TAB COLOR SCHEME FOR BLUE */
#formholder .tabs label {
    background: #16b3f5;
}

#formholder .fa:hover{
    color:#009EE0;
}

#formholder .tabs label:hover {
```

```
    background: #009EE0;
}

#formholder .tabs .tab-content {
    background: #009EE0;
}

#formholder .tabs [id^="tab"]:checked + label
    { background: #009EE0;
}

#formholder .tabs input[type="button"] {
    background:#3a57af;
    color: #f0f3f6;
}

#formholder .tabs input[type="button"]:hover {
    background:#c4f1fe;
    color: #060930;
}

#formholder .tabaddon {
    color: #16B3F5;
}

/******************** END ********************/

/* TAB Form */
.tabs input[type="text"],
.tabs input[type="number"],
.tabs input[type="date"],
.tabs input[type="password"],
.tabs input[type="email"],
.tabs select {
    border: medium none;
    color: #757575;
    height: 40px;
    line-height: 40px;
    margin: 0 auto;
    margin-bottom: 10px;
    padding: 0 5px;
    width: 89%;
}

/* Estilos para los input type=radio del género */
.tabs .accounttype input[type="checkbox"] {
    /* visibility: hidden; */
    display: none;
}

.tabs .accounttype label.radio {
    cursor: pointer;
```

```
    display: inline-block;
    margin-bottom: 15px;
    overflow: visible;
    position: relative;
    text-indent: 20px;
}

.tabs .accounttype label.radio:before {
    background: #778899;
    -ms-border-radius: 100%;
    -o-border-radius: 100%;
    -moz-border-radius: 100%;
    -webkit-border-radius: 100%;
    border-radius: 10%;
    content:'';
    position: absolute;
    top: 16px;
    left: 6px;
    height: 20px;
    width: 20px;
}

.tabs .accounttype label.radio:after {
    background: transparent;
    border: 3px solid #ffffff;
    border-top: none;
    border-right: none;
    content: '';
    height: 0.25em;
    left: 10px;
    opacity: 0;
    position: absolute;
    -webkit-transform: rotate(-45deg);
    -moz-transform: rotate(-45deg);
    -o-transform: rotate(-45deg);
    -ms-transform: rotate(-45deg);
    transform: rotate(-45deg);
    top: 22px;
    width: 0.5em;
}

.tabs .accounttype input[type="checkbox"]:checked + label:after
    { opacity: 1;
}

/* Estilos del elemento select de carrera */
.tabs select {
    float:left;
    width: 89%;
}

.tabs select.last {
```

```
    margin-right:0px;
}

.tabs .btn {
    float: left;
    margin-top: 15px;
    width: 100%;
}

.tabs .btn em{
    font-size:12px;
}

.tabs input[type="button"]{
    display:block;
    float:right;
    cursor:pointer;
    border-top: 2px solid #a3ceda;
    border-left: 2px solid #a3ceda;
    border-right: 2px solid #4f6267;
    border-bottom: 2px solid #4f6267;
    padding: 10px 20px !important;
    font-size: 14px !important;
    background-color: #c4f1fe;
    font-weight: bold;
    color: #2d525d;
    text-decoration:none;
}

.tabs input[type="button"]:hover{
    -webkit-transition: 0.5s;
    -moz-transition: 0.5s;
    -o-transition: 0.5s;
    -ms-transition: 0.5s;
    transition: 0.5s;
}

.tabs h4{
    font-style:italic;
    text-align:center;
    margin-bottom:10px;
    font-weight:normal;
}

.tabaddon {
    background: none repeat scroll 0 0 #FFFFFF;
    border-right: 1px solid #ebebeb;
    float: left;
    height: 40px;
    text-align: center;
    width: 40px;
}
```

```
.tabs .social {
    float:left;
    margin-bottom:10px;
    width:100%;
}
.tabs .social a {
    color:#FFF;
    float:left;
    font-size:14px;
    padding: 10px 65px 10px 30px;
    text-decoration:none;
}

.tabs .social .fa{
    margin-right:25px;
}

.tabs .social .facebook{
    background: none repeat scroll 0 0 #3B5998;
}

.tabs .social .facebook:hover{
    background:#3e5fa4;
    -moz-transition: 0.5s;
    -ms-transition: 0.5s;
    -o-transition: 0.5s;
    -webkit-transition: 0.5s;
    transition: 0.5s;
}

.tabs .social .twitter{
    background: none repeat scroll 0 0 #16B3F5;
}

.tabs .social .twitter:hover{
    background:#34bbf4;
    -moz-transition: 0.5s;
    -ms-transition: 0.5s;
    -o-transition: 0.5s;
    -webkit-transition: 0.5s;
    transition: 0.5s;
}
```

Resultado:

Nueva tabla

Celda(1, 1)	Celda(1, 2)	Celda(1, 3)	Celda(1, 4)
Celda(2, 1)	Celda(2, 2)	Celda(2, 3)	Celda(2, 4)
Celda(3, 1)	Celda(3, 2)	Celda(3, 3)	Celda(3, 4)
Celda(4, 1)	Celda(4, 2)	Celda(4, 3)	Celda(4, 4)
Celda(5, 1)	Celda(5, 2)	Celda(5, 3)	Celda(5, 4)
Celda(6, 1)	Celda(6, 2)	Celda(6, 3)	Celda(6, 4)
Celda(7, 1)	Celda(7, 2)	Celda(7, 3)	Celda(7, 4)
Celda(8, 1)	Celda(8, 2)	Celda(8, 3)	Celda(8, 4)
Celda(9, 1)	Celda(9, 2)	Celda(9, 3)	Celda(9, 4)

Ejemplo #5: El siguiente ejemplo muestra cómo acceder al contenido de una tabla mediante colecciones de tablas como tBodies y métodos como insertRow(). Al final se muestran los datos de la tabla en un elemento div.

Archivo 1: tablaposiciones.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Liga de Campeones de Europa - Grupo
    A</title> <meta charset="utf-8" />
    <link rel="stylesheet" href="css/basic.css" />
    <link rel="stylesheet" href="css/table.css" />
    <script src="js/tablapos.js"></script>
</head>
<body>
<header>
    <h1>Tabla de posiciones</h1>
</header>
<section>
<article>
<table>
<caption>Grupo A</caption>
<thead>
```

```
<tr>
    <th scope="col">
        Posición
    </th>
    <th scope="col">
        Equipo
    </th>
    <th scope="col">
        PJ
    </th>
    <th scope="col">
        PG
    </th>
    <th scope="col">
        PE
    </th>
    <th scope="col">
        PP
    </th>
    <th scope="col">
        GF
    </th>
    <th scope="col">
        GC
    </th>
    <th scope="col">
        PTS
    </th>
</tr>
</thead>
<tbody>
<tr class="even">
    <th scope="row" class="position">
        1
    </th>
    <td>
        Manchester United
    </td>
    <td>
        20
    </td>
    <td>
        16
    </td>
    <td>
        2
    </td>
    <td>
        2
    </td>
    <td>
        28
```

```
    </td>
    <td>
        13
    </td>
    <td>
        50
    </td>
</tr>
<tr>
    <th scope="row" class="position">
        2
    </th>
    <td>
        Juventus
    </td>
    <td>
        20
    </td>
    <td>
        12
    </td>
    <td>
        8
    </td>
    <td>
        0
    </td>
    <td>
        24
    </td>
    <td>
        18
    </td>
    <td>
        44
    </td>
</tr>
<tr class="even">
    <th scope="row" class="position">
        3
    </th>
    <td>
        AJAX
    </td>
    <td>
        20
    </td>
    <td>
        10
    </td>
    <td>
        6
```

```
    </td>
    <td>
        4
    </td>
    <td>
        21
    </td>
    <td>
        19
    </td>
    <td>
        36
    </td>
</tr>
<tr>
    <th scope="row" class="position">
        4
    </th>
    <td>
        Roma
    </td>
    <td>
        20
    </td>
    <td>
        8
    </td>
    <td>
        7
    </td>
    <td>
        5
    </td>
    <td>
        17
    </td>
    <td>
        22
    </td>
    <td>
        31
    </td>
</tr>
<tr class="even">
    <th scope="row" class="position">
        5
    </th>
    <td>
        Porto
    </td>
    <td>
        20
```

```
    </td>
    <td>
        6
    </td>
    <td>
        4
    </td>
    <td>
        10
    </td>
    <td>
        14
    </td>
    <td>
        30
    </td>
    <td>
        22
    </td>
</tr>
<tr>
    <th scope="row" class="position">
        6
    </th>
    <td>
        Galatasaray
    </td>
    <td>
        20
    </td>
    <td>
        4
    </td>
    <td>
        3
    </td>
    <td>
        13
    </td>
    <td>
        9
    </td>
    <td>
        38
    </td>
    <td>
        15
    </td>
</tr>
</tbody>
</table>
<div id="results">
```

```
</div>
</article>
</section>
</body>
</html>
```

Archivo 2: tablapos.js

```
function init(){
    //Variable donde iremos acumulando los resultados que queremos
presentar
    var results = "";
    //Obtener la cantidad de filas de la tabla
    var tablas = document.getElementsByTagName("table");
    var numRows = tablas[0].rows.length;
    //Creando el nodo dentro del elemento div donde mostraremos los
resultados
    var div = document.getElementById("results");
    //Creando un nodo para titular de los resultados
    var highlighted = document.createElement("p");
    div.appendChild(highlighted);
    highlighted.setAttribute("class", "highlighted");
    //Creando el nodo de texto para el titular de los resultados
    highlighted.appendChild(document.createTextNode("Recorrido por los
nodos de la tabla"));
    var paragraph1 = document.createElement("p");
    div.appendChild(paragraph1);
    results += "Cantidad de filas: " + numRows + "\n";
    //Agregando los resultados que queremos mostrar en el div
    paragraph1.appendChild(document.createTextNode(results));
    results = "";
    //Cantidad de elementos thead

    //Cantidad de elementos tbody
    var numBodies = tablas[0].tBodies.length;
    //Creando otro párrafo dentro del div para continuar mostrando
resultados
    var paragraph2 = document.createElement("p");
    div.appendChild(paragraph2);
    results += "Cantidad de tbody en la tabla: " + numBodies +
    "\n"; //Agregando los resultados que queremos mostrar en el div
    paragraph2.appendChild(document.createTextNode(results));
    results = "";
    //Cantidad de filas del primer elemento tbody
    var numRowsBody = tablas[0].tBodies[0].rows.length;
    //Creando otro párrafo dentro del div para continuar mostrando
resultados
    var paragraph3 = document.createElement("p");
    div.appendChild(paragraph3);
    results += "Cantidad del filas del elemento tbody: " + numRowsBody +
"\n";
    //Agregando los resultados que queremos mostrar en el div
```

```
    paragraph3.appendChild(document.createTextNode(results));
    results = "";
    //Cantidad de columnas del primer elemento tbody
    var numCols = tablas[0].rows[0].cells.length;
    //Creando otro párrafo dentro del div para continuar mostrando
resultados
    var paragraph4 = document.createElement("p");
    div.appendChild(paragraph4);
    results += "Cantidad de columnas del elemento tbody: " + numCols
+ "\n";
    //Agregando los resultados que queremos mostrar en el div
    paragraph4.appendChild(document.createTextNode(results));
    results = new Array();
    //Navegando por toda la tabla de filas a
    columnas var rows = tablas[0].rows; var
    rowsTotal = rows.length;
    var fila = "";
    //Recorrido por todas los elementos de la
    tabla for(var i=0; i<rowsTotal; i++){
        var fila = rows[i];
        var columnas = fila.cells;
        var colsTotal = columnas.length;
        for(var j=0; j<colsTotal; j++){
            var columna = columnas[j];
            results += "(fila: " + i + ", col: " + j + ") = " +
columna.innerHTML + "\n";
        }
        //Creando otro párrafo dentro del div para continuar mostrando
resultados
        fila[i] = document.createElement("p");
        div.appendChild(fila[i]);
        //Agregando los resultados que queremos mostrar en el div
        fila[i].appendChild(document.createTextNode(results));
        results = "";
    }
}

if(window.addEventListener){
    window.addEventListener("load", init, false);
}
else if(window.attachEvent){
     window.attachEvent("onload", init);
}
```

Archivo 3: basic.css

```
* {
    margin: 0;
    padding: 0;
}

html, body {
```

```
    height: 100%;
}

body {
    font-size: 16px;
    background: #d5d5d5;;
}

header h1 {
    color: #6d0019;
    font-size: 3.6em;
    text-align: center;
    text-shadow: 1px 1px  0 #CCC,
                 2px 2px  0 #CCC, /* end of 2 level deep grey shadow */
                 3px 3px  0 #444,
                 4px 4px  0 #444,
                 5px 5px  0 #444,
                 6px 6px  0 #444; /* end of 4 level deep dark shadow */
    /* CSS3 Transition Effect */
     -webkit-transition: all 0.5s ease-out;     /* Safari & Chrome */
     -moz-transition: all 0.5s ease-out;   /* Firefox */
     -o-transition: all 0.5s ease-out;      /* Opera */
     transition: all 0.5s ease-out;         /* Estándar */
}

header h1:hover {
    /* CSS3 Transform Effect */
    -webkit-transform: scale(1.2);  /* Safari & Chrome */
     -moz-transform: scale(1.2);     /* Firefox */
     -o-transform: scale(1.2);       /* Opera */
     transform: scale(1.2);          /* Estándar */
}

div#results {
    background: rgba(237,237,237,1);
    background:    -moz-linear-gradient(left,    rgba(237,237,237,1)    0%,
rgba(245,245,245,1) 47%, rgba(235,232,235,1) 100%);
    background:  -webkit-gradient(left  top,  right  top,  color-stop(0%,
rgba(237,237,237,1)),    color-stop(47%,    rgba(245,245,245,1)),    color-
stop(100%, rgba(235,232,235,1)));
    background:   -webkit-linear-gradient(left,   rgba(237,237,237,1)   0%,
rgba(245,245,245,1) 47%, rgba(235,232,235,1) 100%);
    background:    -o-linear-gradient(left,    rgba(237,237,237,1)    0%,
rgba(245,245,245,1) 47%, rgba(235,232,235,1) 100%);
    background:    -ms-linear-gradient(left,    rgba(237,237,237,1)    0%,
rgba(245,245,245,1) 47%, rgba(235,232,235,1) 100%);
    background:   linear-gradient(to   right,   rgba(237,237,237,1)   0%,
rgba(245,245,245,1) 47%, rgba(235,232,235,1) 100%);
    filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#ededed',
endColorstr='#ebe8eb', GradientType=1 );
    -moz-border-radius: 20px;
    -webkit-border-radius: 20px;
```

```
    -o-border-radius: 20px;
    -ms-border-radius: 20px;
    border-radius: 20px;
    -webkit-box-shadow: 10px 10px 5px 0px rgba(56,54,56,1);
    -moz-box-shadow: 10px 10px 5px 0px rgba(56,54,56,1);
    -o-box-shadow: 10px 10px 5px 0px rgba(56,54,56,1);
    -ms-box-shadow: 10px 10px 5px 0px rgba(56,54,56,1);
    box-shadow: 10px 10px 5px 0px rgba(56,54,56,1);
    font-family:Arial, Helvetica, sans-serif;
    font-size:0.85em;
    height: auto;
    margin: 0 auto;
    width: 75%;
}

div#results p {
    padding-left: 20px;
    padding-right: 20px;
    padding-bottom: 16px;
    color: #666;
}

p.highlighted {
    background: #6d0019;
    -moz-border-radius-topleft:20px;
    -webkit-border-top-left-radius:20px;
    border-top-left-radius:20px;
    -moz-border-radius-topright:20px;
    -webkit-border-top-right-radius:20px;
    border-top-right-radius:20px;
    color: #fff !important;
    font-size: 1em;
    font-weight: bold;
    margin-bottom: 16px;
    padding: 16px 20px;
}
```

Archivo 4: table.css

```
table {
    background:#eaebec;
    border:#ccc 1px solid;
    -moz-box-shadow: 0 1px 2px #d1d1d1;
    -webkit-box-shadow: 0 1px 2px #d1d1d1;
    -moz-border-radius:3px;
    -webkit-border-radius:3px;
    border-radius:3px;
    box-shadow: 0 1px 2px #d1d1d1;
    color:#666;
    font-family:Arial, Helvetica, sans-serif;
    font-size:0.85em;
    margin:20px auto;
```

```
    text-shadow: 1px 1px 0px #fff;
}

table caption {
    color:#fff;
    font:normal 2em Arial;
    text-shadow: 0 1px 0 #ccc,
                 0 2px 0 #c9c9c9,
                 0 3px 0 #bbb,
                 0 4px 0 #b9b9b9,
                 0 5px 0 #aaa,
                 0 6px 1px rgba(0,0,0,.1),
                 0 0 5px rgba(0,0,0,.1),
                 0 1px 3px rgba(0,0,0,.3),
                 0 3px 5px rgba(0,0,0,.2),
                 0 5px 10px rgba(0,0,0,.25),
                 0 10px 10px rgba(0,0,0,.2),
                 0 20px 20px rgba(0,0,0,.15);
}

table th {
    border-bottom:1px solid #e0e0e0;
    background: #a90329; /* Old browsers */
    background: -moz-linear-gradient(top, #a90329 0%, #8f0222 44%,
#6d0019 100%); /* FF3.6+ */
    background: -webkit-gradient(linear, left top, left bottom, color-
stop(0%,#a90329), color-stop(44%,#8f0222), color-stop(100%,#6d0019)); /*
Chrome,Safari4+ */
    background: -webkit-linear-gradient(top, #a90329 0%,#8f0222
44%,#6d0019 100%); /* Chrome10+,Safari5.1+ */
    background: -o-linear-gradient(top, #a90329 0%,#8f0222
44%,#6d0019 100%); /* Opera 11.10+ */
    background: -ms-linear-gradient(top, #a90329 0%,#8f0222 44%,#6d0019
100%); /* IE10+ */
    background: linear-gradient(to bottom, #a90329 0%,#8f0222 44%,#6d0019
100%); /* W3C */
    filter: progid:DXImageTransform.Microsoft.gradient(
startColorstr='#a90329', endColorstr='#6d0019',GradientType=0 ); /* IE6-9
*/
    border-top:1px solid #fafafa;
    color: #feb645;
    padding:21px 25px 22px 25px;
}

table th:first-child {
    padding-left:20px;
    text-align: left;
}

table tr:first-child th:first-child {
    -moz-border-radius-topleft:3px;
    -webkit-border-top-left-radius:3px;
```

```
    border-top-left-radius:3px;
}

table tr:first-child th:last-child {
    -moz-border-radius-topright:3px;
    -webkit-border-top-right-radius:3px;
    border-top-right-radius:3px;
}

table tr {
    padding-left:20px;
    text-align: center;
}

table td:first-child {
    border-left: 0;
    padding-left:20px;
    text-align: left;
}

table td {
    background: #fafafa;
    background: -webkit-gradient(linear, left top, left bottom,
from(#fbfbfb), to(#fafafa));
    background: -moz-linear-gradient(top,  #fbfbfb,  #fafafa);
    border-top: 1px solid #ffffff;
    border-bottom:1px solid #e0e0e0;
    border-left: 1px solid #e0e0e0;
    padding:18px;
}

table tr th.position {
    text-align: center;
}

table tr.even td {
    background: rgba(237,237,237,1);
    background: -moz-linear-gradient(left, rgba(237,237,237,1) 0%,
rgba(245,245,245,1) 47%, rgba(235,232,235,1) 100%);
    background: -webkit-gradient(left top, right top, color-stop(0%,
rgba(237,237,237,1)), color-stop(47%, rgba(245,245,245,1)), color-
stop(100%, rgba(235,232,235,1)));
    background: -webkit-linear-gradient(left, rgba(237,237,237,1) 0%,
rgba(245,245,245,1) 47%, rgba(235,232,235,1) 100%);
    background: -o-linear-gradient(left, rgba(237,237,237,1) 0%,
rgba(245,245,245,1) 47%, rgba(235,232,235,1) 100%);
    background: -ms-linear-gradient(left, rgba(237,237,237,1)
0%, rgba(245,245,245,1) 47%, rgba(235,232,235,1) 100%);
    background: linear-gradient(to right, rgba(237,237,237,1)
0%, rgba(245,245,245,1) 47%, rgba(235,232,235,1) 100%);
    filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#ededed',
endColorstr='#ebe8eb', GradientType=1 );
```

```
}

table tr:last-child td {
    border-bottom:0;
}

table tr:last-child td:first-child {
    -moz-border-radius-bottomleft:3px;
    -webkit-border-bottom-left-radius:3px;
    border-bottom-left-radius:3px;
}

table tr td:last-child {
    font-weight: bold;
    color: #6d0019;
}

table tr:last-child td:last-child {
    -moz-border-radius-bottomright:3px;
    -webkit-border-bottom-right-radius:3px;
    border-bottom-right-radius:3px;
}

table tr:hover td {
    background: rgba(219,219,219,1);
    background: -moz-linear-gradient(left, rgba(219,219,219,1) 0%,
rgba(209,209,209,1) 51%, rgba(224,224,224,1) 100%);
    background: -webkit-gradient(left top, right top, color-stop(0%,
rgba(219,219,219,1)), color-stop(51%, rgba(209,209,209,1)), color-
stop(100%, rgba(224,224,224,1)));
    background: -webkit-linear-gradient(left, rgba(219,219,219,1) 0%,
rgba(209,209,209,1) 51%, rgba(224,224,224,1) 100%);
    background: -o-linear-gradient(left, rgba(219,219,219,1) 0%,
rgba(209,209,209,1) 51%, rgba(224,224,224,1) 100%);
    background: -ms-linear-gradient(left, rgba(219,219,219,1)
0%, rgba(209,209,209,1) 51%, rgba(224,224,224,1) 100%);
    background: linear-gradient(to right, rgba(219,219,219,1)
0%, rgba(209,209,209,1) 51%, rgba(224,224,224,1) 100%);
    filter: progid:DXImageTransform.Microsoft.gradient(
startColorstr='#dbdbdb', endColorstr='#e0e0e0', GradientType=1 ); }
```

Resultado:

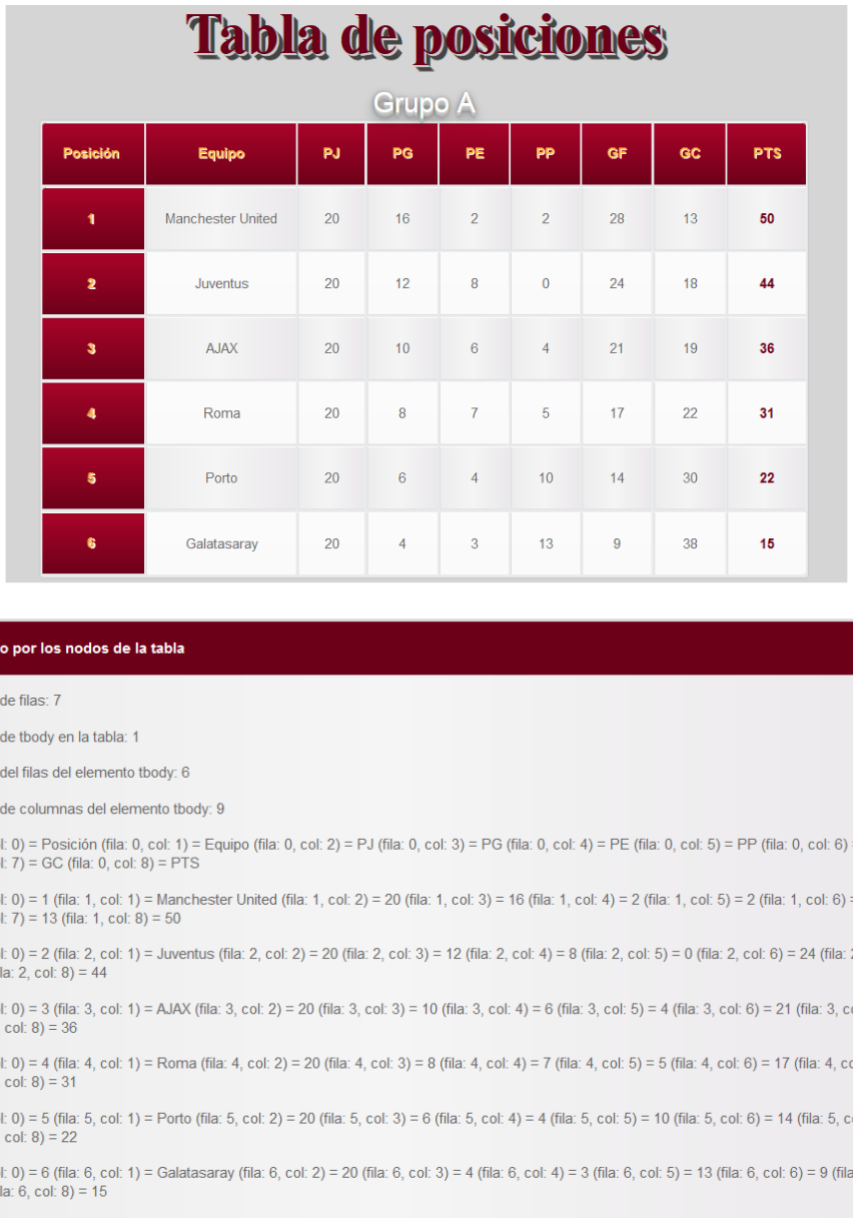


Tabla de posiciones

Grupo A

Posición	Equipo	PJ	PG	PE	PP	GF	GC	PTS
1	Manchester United	20	16	2	2	28	13	50
2	Juventus	20	12	8	0	24	18	44
3	AJAX	20	10	6	4	21	19	36
4	Roma	20	8	7	5	17	22	31
5	Porto	20	6	4	10	14	30	22
6	Galatasaray	20	4	3	13	9	38	15

Recorrido por los nodos de la tabla

Cantidad de filas: 7

Cantidad de tbody en la tabla: 1

Cantidad del filas del elemento tbody: 6

Cantidad de columnas del elemento tbody: 9

(fila: 0, col: 0) = Posición (fila: 0, col: 1) = Equipo (fila: 0, col: 2) = PJ (fila: 0, col: 3) = PG (fila: 0, col: 4) = PE (fila: 0, col: 5) = PP (fila: 0, col: 6) = GF (fila: 0, col: 7) = GC (fila: 0, col: 8) = PTS

(fila: 1, col: 0) = 1 (fila: 1, col: 1) = Manchester United (fila: 1, col: 2) = 20 (fila: 1, col: 3) = 16 (fila: 1, col: 4) = 2 (fila: 1, col: 5) = 2 (fila: 1, col: 6) = 28 (fila: 1, col: 7) = 13 (fila: 1, col: 8) = 50

(fila: 2, col: 0) = 2 (fila: 2, col: 1) = Juventus (fila: 2, col: 2) = 20 (fila: 2, col: 3) = 12 (fila: 2, col: 4) = 8 (fila: 2, col: 5) = 0 (fila: 2, col: 6) = 24 (fila: 2, col: 7) = 18 (fila: 2, col: 8) = 44

(fila: 3, col: 0) = 3 (fila: 3, col: 1) = AJAX (fila: 3, col: 2) = 20 (fila: 3, col: 3) = 10 (fila: 3, col: 4) = 6 (fila: 3, col: 5) = 4 (fila: 3, col: 6) = 21 (fila: 3, col: 7) = 19 (fila: 3, col: 8) = 36

(fila: 4, col: 0) = 4 (fila: 4, col: 1) = Roma (fila: 4, col: 2) = 20 (fila: 4, col: 3) = 8 (fila: 4, col: 4) = 7 (fila: 4, col: 5) = 5 (fila: 4, col: 6) = 17 (fila: 4, col: 7) = 22 (fila: 4, col: 8) = 31

(fila: 5, col: 0) = 5 (fila: 5, col: 1) = Porto (fila: 5, col: 2) = 20 (fila: 5, col: 3) = 6 (fila: 5, col: 4) = 4 (fila: 5, col: 5) = 10 (fila: 5, col: 6) = 14 (fila: 5, col: 7) = 30 (fila: 5, col: 8) = 22

(fila: 6, col: 0) = 6 (fila: 6, col: 1) = Galatasaray (fila: 6, col: 2) = 20 (fila: 6, col: 3) = 4 (fila: 6, col: 4) = 3 (fila: 6, col: 5) = 13 (fila: 6, col: 6) = 9 (fila: 6, col: 7) = 38 (fila: 6, col: 8) = 15

Ejemplo #6: El siguiente ejemplo muestra la creación de un formulario mediante un panel de botones de formulario, donde cada botón representa un tipo diferente de campo. Se consideran únicamente tipos de campos input y textarea.

Archivo 1: formbuilder.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <title>
        Formulario creado dinámicamente con
    JavaScript </title>
    <link rel="stylesheet" href="css/form.css" />
```

```
    <script src="js/form.js"></script>
</head>
<body>
<div class="main_content">
<!--
================================================================================
===============
Header Div.
================================================================================
===============
-->
<div class="first">
    <p>
        <img id="logo" src="img/form.png" />
        <h1>Constructor de formularios</h1>
    </p>
</div>
<!--
================================================================================
=============
This Div is for the Buttons. When user click on buttons, respective field
will appear.
================================================================================
==============
-->
<div class="two">
    <h4>Campos de formulario usados con frecuencia</h4>
    <button onclick="nameFunction()">Nombre</button> <button
    onclick="emailFunction()">Correo</button> <button
    onclick="contactFunction()">Contacto</button> <button
    onclick="textareaFunction()">Comentario</button> <button
    onclick="fileFunction()">Foto</button> <button
    onclick="resetElements()">Restablecer</button>
</div>
<!--
================================================================================
===============
This Div is meant to display final form.
================================================================================
===============
-->
<div class="three">
    <h2>Formulario creado dinámicamente</h2>
    <form action="#" id="mainform" method="get" name="mainform">
        <span id="myForm"></span>
        <p></p>
        <input type="submit"
    value="Enviar"> </form>
</div>
</div> <!-- Fin de div id="main_content" -->
</body>
</html>
```

Archivo 2: form.js

```
/* Inicializando variable global i */
var i = 0;

/*********************************************************************
* Función para incremento automático de contador que se concatenará *
* con el valor del atributo name en los campos de formulario.       *
*********************************************************************/
function increment(){
    i += 1;
}

/*********************************************************************
* Función que permite eliminar dinámicamente campos de formulario *
*********************************************************************/
function removeElement(parentDiv,
    childDiv){ if (childDiv == parentDiv){
        alert("Contenedor padre no puede ser removido.");
    }
    else if (document.getElementById(childDiv)){
        var child = document.getElementById(childDiv);
        var parent = document.getElementById(parentDiv);
        parent.removeChild(child);
    }
    else{
        alert("Contenedor hijo ha sido removido o no
        existe."); return false;
    }
}

/*********************************************************************
* Función que será llamada cuando el usuario haga clic en el campo *
* de texto Nombre.                                                  *
*********************************************************************/
function nameFunction(){
    var r = document.createElement("span");
    var y = document.createElement("input");
    y.setAttribute("type", "text");
    y.setAttribute("placeholder", "Nombre");
    var g = document.createElement("img");
    g.setAttribute("src", "img/delete.png");
    increment();
    y.setAttribute("name", "textelement_" + i);
    r.appendChild(y);
    g.setAttribute("onclick", "removeElement('myForm','id_" + i + "')");
    r.appendChild(g);
    r.setAttribute("id", "id_" + i);
    document.getElementById("myForm").appendChild(r);
}
```

```
/*****************************************************************
* Función que será llamada cuando el usuario haga clic en el campo *
* de texto email                                                  *
*****************************************************************/
function emailFunction(){
    var r = document.createElement("span"); var y =
    document.createElement("input");
    y.setAttribute("type", "text");
    y.setAttribute("placeholder", "Correo electrónico");
    var g = document.createElement("img");
    g.setAttribute("src", "img/delete.png"); increment();

    y.setAttribute("name", "textelement_" + i);
    r.appendChild(y);
    g.setAttribute("onclick", "removeElement('myForm','id_" + i + "')");
    r.appendChild(g);
    r.setAttribute("id", "id_" + i);
    document.getElementById("myForm").appendChild(r);
}

/*****************************************************************
* Función que será llamada cuando el usuario haga clic en el campo *
* de texto Contacto.                                              *
*****************************************************************/
function contactFunction(){
    var r = document.createElement("span");
    var y = document.createElement("input");
    y.setAttribute("type", "text");
    y.setAttribute("placeholder", "Contact");
    var g = document.createElement("img");
    g.setAttribute("src", "img/delete.png");
    increment();
    y.setAttribute("Name", "textelement_" + i);
    r.appendChild(y);
    g.setAttribute("onclick", "removeElement('myForm','id_" + i + "')");
    r.appendChild(g);
    r.setAttribute("id", "id_" + i);
    document.getElementById("myForm").appendChild(r);
}

/*****************************************************************
* Función que será llamada cuando el usuario haga clic en el botón *
* Comentario.                                                     *
*****************************************************************/
function textareaFunction(){
    var r = document.createElement("span"); var
    y = document.createElement("textarea"); var
    g = document.createElement("img");
    y.setAttribute("cols", "18");
    y.setAttribute("placeholder",
    "Comentario"); g.setAttribute("src",
    "img/delete.png"); increment();
```

```
    y.setAttribute("name", "textelement_" + i);
    r.appendChild(y);
    g.setAttribute("onclick", "removeElement('myForm','id_" + i + "')");
    r.appendChild(g);
    r.setAttribute("id", "id_" + i);
    document.getElementById("myForm").appendChild(r);
}

/*****************************************************************
* Función que será llamada cuando el usuario haga clic en el botón *
* Adjunto.                                                        *
*****************************************************************/
function fileFunction(){
    var r = document.createElement("span");
    var y = document.createElement("input");
    y.setAttribute("type", "file");
    y.setAttribute("accept", "image/*");
    y.setAttribute("name", "fileelement_");
    increment();
    r.appendChild(y);
    var g = document.createElement("img");
    g.setAttribute("src", "img/delete.png");
    g.setAttribute("onclick", "removeElement('myForm','id_" + i + "')");
    r.appendChild(g);
    r.setAttribute("id", "id_" + i);
    document.getElementById("myForm").appendChild(r);
}

/*****************************************************************
* Función que será llamada cuando el usuario haga clic en el botón *
* reset del formulario.                                           *
*****************************************************************/
function resetElements(){
    document.getElementById('myForm').innerHTML = '';
}
```

Archivo 3: form.css

```
* {
    margin: 0;
    padding: 0;
}

*:focus {
     outline: none;
}

html, body {
    height: 100%;
}

body{
```

```
    background-color:#f9ebe8;
    font-size: 16px;
    margin:45px auto;
    width:980px;
}

form{
    border-top:1px dotted #D9D9D9;
    margin:10px 180px;
    width:330px;
}

button{
    color:#4C4C4C;
    height:40px;
    margin-bottom:20px;
    margin-left:20px;
    width:246px;
}

input{
    -moz-border-radius:5px;
    -webkit-border-radius:5px;
    -o-border-radius:5px;
    -ms-border-radius:5px;
    border-radius:5px;
    border:4px solid #acbfa5;
    height:28px;
    margin:20px 0 10px;
    padding:5px;
    width:280px;
}

input[type="submit"] {
    background-color:#35c8ef;
    -moz-border-radius:5px;
    -webkit-border-radius:5px;
    -o-border-radius:5px;
    -ms-border-radius:5px;
    border-radius:5px;
    border:3px outset #a8d0e3;
    color:#fff;
    font-weight:bold;
    height:38px;
    width:100px;
}

input[type="submit"]:hover { background-
    color:rgba(60,210,250,0.8); border:3px
    inset #a8d0e3; color:#fff;

}
```

```
textarea{
    border-radius:5px;
    border:4px solid #acbfa5;
    height:70px;
    margin:20px 0 10px;
    padding:5px;
    width:280px;
}

.four p{
    color:#fff;
    padding:15px 0;
    text-align:center;
}

.first {
    background-color:#41A2CD;
    padding: 6px 8px;
    height:60px;
    width:960px;
}

.first img {
    float: left;
    top: 4px;
}

.first h1 {
    color: #7d3520;
    float: left;
    font-size: 3em;
    text-shadow: 1px 1px 0 #f1c5b6,
                 4px 4px 0 rgba(0, 0, 0, 0.25);
    top: 0;
}

.first p{
    color:#fff;
}

.two{
    background-color:#fff;
    clear: both;
    float:left;
    height:600px;
    width:290px;
}

.main_content{
    background-color:#fff;
    height:auto;
```

```
    width:960px;
}

.two h4{
    color:#4C4C4C;
    text-align:center;
}

.three{
    background-color:#fff;
    border-left:1px solid #D0D0D0;
    float:left;
    padding: 6px 4px;
    text-align:center;
    width:660px;
}

.three h2 {
    color: #41A2CD;
}

.four{
    background-color:#41A2CD;
    clear:both;
    height:55px;
    width:960px;
}
```

Resultado:

V. BIBLIOGRAFIA

- Flanagan, David. JavaScript La Guía Definitiva. 1ra Edición. Editorial ANAYA Multimedia. 2007. Madrid, España.
- Deitel, Paul / Deitel, Harvey / Deitel, Abbey. Internet & World Wide Web. Cómo programar. 5a. Edición. Editorial Pearson. 2014. México D.F..
- Tom Negrito / Dori Smith. JavaScrip y AJAX para diseño web. Editorial Pearson Prentice Hall. Madrid, España. 2007.
- John Resig / Bear Bibeault. JavaScript Ninja.1a. Edición. Editorial Anaya Multimedia / Manning. Septiembre 2013. Madrid, España.
- Powell, Thomas / Schneider, Fritz. JavaScript Manual de Referencia. 1ra Edición. Editorial McGraw-Hill. 2002. Madrid, España.
- McFedries, Paul. JavaScript Edición Especial. 1ra Edición. Editorial Prentice Hall. 2002. Madrid, España.