	UNIVERSIDAD DON BOSCO **FACULTAD DE ESTUDIOS TECNOLÓGICOS** **COMPUTACIÓN**
CICLO 2 **2020**	*GUIA DE LABORATORIO #9* **Nombre de la Practica:** **Manejo de Hojas de Estilo con el DOM** **Lugar de Ejecución:** **Centro de cómputo** **Tiempo Estimado:** **2 horas con 30 minutos** **MATERIA:** **Lenguajes Interpretado en el Cliente**

I. OBJETIVOS

Que el estudiante:

- Recuerde los aspectos principales de la utilización de Hojas de Estilo en Cascada.
- Identifique las propiedades y métodos principales para el manejo de Hojas de Estilo desde un script.
- Haga uso de las propiedades y métodos de estilo del DOM para modificar dinámicamente un documento web.
- Realice aplicaciones con HTML Dinámico (DHTML) modificando propiedades de estilo de forma dinámica utilizando JavaScript.

II. INTRODUCCION TEORICA

Hojas de Estilo en Cascada

Las Hojas de Estilo en Cascada son un estándar para especificar la presentación de documentos HTML o XML. De acuerdo a este estándar, los elementos HTML deberían utilizarse solamente para definir la estructura del documento, dejando la tarea de la presentación y formato del mismo a las Hojas de Estilo. Esto significa que deberían evitarse a toda costa, la utilización de elementos HTML desaprobados como FONT, CENTER, B, I, U y los atributos HTML relacionados con la presentación, como bgcolor, align, bordercolor, etc.

CSS (*Cascading Style Sheets*) es una tecnología diseñada para ser utilizada por los diseñadores web o cualquiera preocupado por la precisión visual de los documentos HTML. Es vital su utilización desde JavaScript debido a que el Modelo de Objetos de Documento (DOM) permite gestionar, dinámicamente, a través de secuencias de comando, los estilos aplicados a los elementos individuales de un documento. La utilización conjunta de CSS y JavaScript permite crear una variedad de efectos visuales conocidos como HTML Dinámico (DHTML).

Formas de aplicación de estilos en un documento

Antes que nada, hay que recordar cómo se pueden aplicar estilos en un documento. Básicamente son tres formas:

1. Estilos en línea

Se aplica una o varias reglas de estilo de forma directa en la etiqueta, haciendo uso del atributo style, válido para diversos elementos HTML.

Ejemplo:

```
<p style="color:green;border:solid red 2px;">Este párrafo tiene el texto verde y un borde rojo.</p>
```

2. Estilos internos

Es una hoja de estilo que se incrusta dentro del documento HTML dentro del elemento STYLE que debe estar en la cabecera del documento (HEAD). Debe proporcionar atributos adicionales al elemento, entre los que se destacan: rel, type, href y media.
Ejemplo:

```
<style type="text/css">
   h1 {
      border-width:1;
      border:solid;
      text-align:center
   }
</style>
```

3. Estilos externos
Es una hoja de estilo que está almacenada en un archivo .css independiente y que se vincula al documento donde se quiere aplicar, haciendo uso del elemento LINK, especificando en su atributo href el documento .css que contiene la definición de estilos para el documento.
Ejemplo:

```
<link rel="stylesheet" type="text/css" href="misestilos.css" />
```

Donde se ha utilizado sintaxis XHTML.

El archivo misestilos.css podría contener un código CSS como el siguiente:

```
h1    {
   font-family:"Bookman Old Style";
   font-weight:bold;
   font-size:20pt;
   font-style:Italia;
   color:Green;
   text-align:center;
}
p  {
   font:italic ligher 1em Garamond;
   width:80%;
   margin:auto;
   text-align:justify;
}
.frase  {
   font:normal bold 0.8em Georgia;
   text-align:center;
}
```

CSS y DHTML
Los desarrolladores web de contenido DHTML, deben poner principal atención en la posibilidad de utilizar atributos CSS normales para especificar la visibilidad, el tamaño y la posición exacta de elementos individuales dentro de un documento. Otras propiedades CSS permiten definir el orden de apilamiento, la transparencia, el área de recorte, los márgenes, el relleno, los bordes y los colores. Para programar con DHTML, es importante saber cómo funcionan estas propiedades. La siguiente tabla resume la mayor parte de estas:

Atributo	Descripción
position	Especifica el tipo de posicionamiento aplicado a un elemento.
top	Indica la posición del borde superior de un elemento. El valor de la medida puede ser indicado en forma relativa o absoluta.
left	Indica la posición del borde izquierdo de un elemento. El valor de la medida puede ser indicado en forma relativa o absoluta.

bottom	Indica la posición del borde inferior de un elemento. El valor de la medida puede ser indicado en forma relativa o absoluta.
right	Indica la posición del borde derecho de un elemento. El valor de la medida puede ser indicado en forma relativa o absoluta.
width, height	Especifican el tamaño de un elemento. El ancho y el alto respectivamente.
z-index	Indica el orden de apilamiento de un elemento con relación a cualquier otro elemento superpuesto, permitiendo establecer una tercera dimensión para indicar la posición del elemento.
display	Especifica cómo se va a presentar un elemento en el documento y, más aún, si se presentará o no.
visibility	Indica si un elemento será visible o no en el documento. El elemento seguirá ocupando un espacio físico en el documento, independientemente de si es visible o no.
clip	Define un área de recorte para un elemento; sólo se muestran dentro de esta área partes del elemento. Especifica lo que se debe hacer si un elemento es mayor que el espacio asignado para éste.
margin	Especifica el margen que el área de contenido del elemento tendrá con respecto a los border del mismo.
border	Indica el ancho del borde de un elemento y la apariencia de este.
background	Especifica el color o la imagen de fondo a utilizar en un elemento, permitiendo definir otras características adicionales de este fondo, como si se repetirá la imagen hasta llenar el fondo, si se mantendrá o no fija cuando se desplace el contenido de la ventana, etc.
opacity	Especifica la opacidad (o transparencia) de un elemento. Este es un atributo de CSS3 admitido en varios navegadores actuales.

La clave del DHTML: el posicionamiento absoluto

El atributo *position* del CSS, indica el tipo de posicionamiento aplicado a un elemento. Los valores posibles para este atributo son cuatro:

- **static**: es el valor predeterminado y se utiliza para indicar que el elemento debe colocarse de acuerdo al flujo normal del contenido del documento (en casi todas las aplicaciones esto significa que debe colocarse de izquierda a derecha o de arriba hacia abajo, mientras exista espacio para el mismo). Los elementos ubicados de forma estática no pueden ser elementos DHTML, lo que significa que no se pueden definir atributos como top, left, right o bottom para posicionarlos de forma dinámica.
- **absolute**: permite establecer la posición del elemento con respecto a su elemento contenedor. Los elementos ubicados de forma absoluta se colocan independientemente del espacio utilizado por el resto de elementos presentes en el documento y no formará parte del flujo normal. Un elemento posicionado de forma absoluta se ubica en el documento con relación a su elemento contenedor. Con frecuencia este elemento contenedor es el elemento BODY, pero si el elemento está anidado dentro de otro elemento, entonces, este elemento será con respecto al que se ubicará. Este es el tipo de posicionamiento utilizado con mayor frecuencia para aplicaciones DHTML, de hecho es el más recomendado por razones de compatibilidad entre navegadores. También, es preciso asegurarse de colocar los elementos posicionados de forma absoluta dentro de elementos DIV o SPAN para asegurarse de su funcionamiento apropiado en navegadores Internet Explorer.
- **fixed**: permite definir la posición de un elemento con respecto a la ventana del explorador web. En teoría, los elementos con posición fija siempre serán visibles y no se desplazarán con el resto de elementos presentes en el documento cuando se utilicen barras de desplazamiento o teclado para acceder a otras partes de la página web. Al igual que los elementos posicionados de forma absoluta, los elementos con posición fija se ubican de forma independiente del resto de elementos y no forman parte del flujo normal del documento. El Internet Explorer 6.0 e inferiores no soportan el posicionamiento fijo.
- **relative**: este tipo de posicionamiento hace que un elemento se coloque, primero, según el flujo normal del documento. Luego, se ajusta su ubicación con relación a su posición en el flujo normal. El espacio asignado para este elemento en el flujo normal, sigue siendo ocupado, aunque el elemento ya no esté en esa posición. Estos significa que el resto de elementos en el flujo normal no pueden ocupar esta posición.

Una vez que se posiciona el elemento con uno de los tres valores que hagan posible realizar aplicaciones de HTML dinámico (*absolute*, *fixed* o *relative*), puede establecerse su posición con una combinación de los atributos ***left***, ***top***, ***right*** y ***bottom***. La técnica de posicionamiento más común es utilizar los atributos ***left*** y ***top***.
Por ejemplo, para posicionar un elemento a 60 píxeles de la parte izquierda y a 75 píxeles de la parte superior del documento, debe especificarse una regla de estilo como la siguiente:

```
selector   {
    position:absolute;
    left:60px;
    top:75px;
}
```

El DOM y las CSS
Las Hojas de Estilo es otro de los aspectos soportados por la mayor parte de los navegadores actuales. Es el DOM Nivel 2 el que permite manipular los valores de las propiedades CSS de los elementos presentes en un documento.

Manipulación de estilos insertados en línea
La forma más sencilla de manipular valores de CSS con JavaScript es a través de la propiedad style que corresponde a la especificación de hojas de estilo insertado en línea para un elemento HTML específico.
Por ejemplo:

```
<p id="myParagraph">Esto es una prueba</p>
```

Insertando el estilo en línea:

```
<p id="myParagraph" style="color:brown">Esto es una prueba</p>
```

Para manipular las interfaces del DOM con JavaScript puede utilizar un script como este:

```
theElement = document.getElementById("myParagraph");
theElement.style.color = "green";
```

En el ejemplo anterior puede verse que se ha utilizado la propiedad style del elemento cuyo id fue definido como "myParagraph". Luego, se estableció la propiedad color del objeto style como "green". Esta propiedad color se corresponde exactamente con la propiedad color de las CSS. El punto clave es, por tanto, saber a qué propiedad del DOM hacer referencia para modificar una propiedad de estilo en particular. La tabla siguiente muestra esta correspondencia:

Propiedad CSS	Propiedad DOM Nivel 2
background	background
background-attachment	backgroundAttachment
background-color	backgroundColor
background-image	backgroundImage
background-position	backgroundPosition
background-repeat	backgroundRepeat
border	border
border-color	borderColor
border-style	borderStyle
border-top	borderTop
border-right	borderRight
border-left	borderLeft
border-bottom	borderBottom
border-top-color	borderTopColor
border-right-color	borderRightColor
border-bottom-color	borderBottomColor
border-left-color	borderLeftColor
border-top-style	borderTopStyle

border-right-style	borderRightStyle
border-bottom-style	borderBottomStyle
border-left-style	borderLeftStyle
border-top-width	borderTopWidth
border-right-width	borderRightWidth
border-bottom-width	borderBottomWidth
border-left-width	borderLeftWidth
border-width	borderWidth
clear	clear
clip	clip
color	color
display	display
float	cssFloat
font	font
font-family	fontFamily
font-size	fontSize
font-style	fontStyle
font-variant	fontVariant
font-weight	fontWeight
height	height
left	left
letter-spacing	letterSpacing
line-height	lineHeight
list-style	listStyle
list-style-image	listStyleImage
list-style-position	listStylePosition
list-style-type	listStyleType
margin	margin
margin-top	marginTop
margin-right	marginRight
margin-bottom	marginBottom
margin-left	marginLeft
overflow	overflow
padding	padding
padding-top	paddingTop
padding-right	paddingRight
padding-bottom	paddingBottom
padding-left	paddingLeft
position	position
text-align	textAlign
text-decoration	textDecoration
text-indent	textIndent
text-transform	textTransform
top	top
vertical-align	verticalAlign
visibility	Visibility
white-space	whiteSpace
width	width
word-spacing	WordSpacing
z-index	zIndex

Estilo dinámico utilizando clases y colecciones

Mediante la utilización de selectores de clase es posible aplicar un conjunto de reglas de estilo a elementos iguales o diferentes. En el siguiente ejemplo se muestran dos clases definidas en un elemento STYLE:

```
<style type="text/css">
   .On     {
      background-color:LightSkyBlue;
      color:Crimson;
      border:5px Crimson groove;
   }

   .Off     {
      background-color:LightSteelBlue;
      color:DarkBlue;
      border:5px DarkBlue ridge;
   }
</style>
```

La idea es definir un elemento HTML como perteneciente a una de las clases. Tomemos la clase .Off y hagamos que el elemento h1 sea de esa clase. Haciéndolo desde HTML estático, tendríamos que crear una etiqueta agregando un atributo class, como se muestra a continuación:

```
<h1 id="encendido" class="Off">Poner el t&iacute;tulo</h1>
```

Para manipular el aspecto del elemento de encabezado h1, puede hacer uso de JavaScript, redefiniendo la clase (***class***) del elemento. El valor del atributo ***class*** puede manipularse haciendo uso de la propiedad del DOM denominada ***className***, de la siguiente forma:

```
theElement = document.getElementById("encendido");
theElement.className = "On";
```

En la práctica estos nombres de clase se redefinen haciendo uso del manejo de eventos de manera que al pasar el ratón por encima (**onmouseover**) del elemento se le da un nombre y al retirarlo (**onmouseout**) se le devuelve el nombre original, con el que se nombró al elemento al cargar la página.

```
<h1 id="encendido" class="Off" onmouseover="this.className='On'"
       onmouseout="this.className='Off'">Poner el t&iacute;tulo</h1>
```

Otra forma de manipular estilos es utilizar el método getElementByTagName() y efectuar los cambios de estilo sobre cada uno de los elementos individuales devueltos. En el procedimiento se mostrará un ejemplo de cómo emplear este método.

Acceso a reglas de estilo complejas

El DOM Nivel 2, permite tener acceso a la colección styleSheets[] que forma parte del objeto Document. A través de esta colección se puede tener acceso a las diversas etiquetas <style> y <link> de un documento. Para acceder a los estilos definidos en estas etiquetas se puede hacer uso de una instrucción como cualquiera de las siguientes:

```
var firstStyleSheet = document.styleSheets[0];
```

o también:

```
var firstStyleSheet = document.styleSheets.item(0);
```

La manipulación de las propiedades del objeto correspondiente al elemento STYLE hallado en el documento HTML es directa, ya que sus propiedades se corresponden con la de los atributos HTML que ya conocemos. Como: type, href, media, disabled, entre otros.

En el DOM, el objeto **CSSStyleSheet** hereda las características del objeto **StyleSheet** y luego añade la colección **cssRules[]**, que contiene las diversas reglas del bloque de estilos, así como los métodos **insertRule()** y **deleteRule()**. Cabe destacar que el Internet Explorer sigue su propio modelo para manipular estilos. Para

este navegador habrá que hacer uso de métodos similares, que son: **addRule()** y **removeRule()** para el objeto **styleSheet**.

La sintaxis del método **insertRule()** es `theStyleSheet.insertRule("texto_regla_estilo", indice)`. Donde, `"texto_regla_estilo"` es una cadena que contiene el selector de hojas de estilo y la declaración. En tanto que, `indice` es la posición para insertarlo dentro del conjunto de reglas.

La sintaxis del método **addRule()**, es similar y tiene la siguiente estructura:

```
theStyleSheet.addRule("h3","color:blue;font-size:12pt;",4);
```

El acceso a las reglas individuales es inmediato al hacer uso de la colección **cssRules[]**, o para Internet Explorer, con la colección no estándar **rules[]**. Una vez que se accede a una regla, se puede establecer o modificar su propiedad selectorText para examinar el selector de una regla en particular. También se puede utilizar la propiedad style para acceder el conjunto real de reglas.

El DOM nivel 2 proporciona diversos métodos para modificar reglas de estilo. Por ejemplo: getPropertyValue() y setProperty(). Sin embargo, es aconsejable acceder a estas a través del objeto style y luego, a la propiedad DOM que corresponda a la propiedad CSS en cuestión. Por ejemplo:

```
theStyleSheet.cssRules[0].style.color = 'blue';   //modifica o añade una propiedad color a
                                                    //la primera regla CSS de la hoja de
estilos
theStyleSheet.rules[0].style.color = 'blue';  //Utilice esta en Internet Explorer
```

III. MATERIALES Y EQUIPO

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

No.	Requerimiento	Cantidad
1	Guía de práctica #6: Manejo de Hojas de Estilo con el DOM	1
2	Computadora con editor HTML y navegadores instalados	1
3	Memoria USB o disco flexible	1

IV. PROCEDIMIENTO

Ejercicio #1: Probador de estilos mediante un campo de selección de formulario, controlando con JavaScript no invasivo los estilos que se aplican a un elemento h1. La secuencia de comando es capaz de agregar más de una clase, permitiendo además de agregar clases, eliminarlas para manejar de forma apropiada el hecho de que un elemento puede tener más de una clase. Si la clase ya fue asignada, la clase no se vuelve a agegar.

Guión 1: testclasses.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Cambio de clases CSS dinámico</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="css/fonts.css" />
    <link rel="stylesheet" href="css/headings.css" />
    <link rel="stylesheet" href="css/button.css" />
    <script src="js/CSSClass.js"></script>
    <script src="js/classname.js"></script>
</head>
<body>
<header>
    <h1 id="head">Texto de prueba estilos con className</h1>
</header>
```

```
<section>
<article>
<form name="frmclass">
    <label>Estilo:</label>
    <select name="classes">
        <option value="" selected="selected">(Escoja un estilo)</option>
        <option value="formal">Formal</option>
        <option value="letterpress">Texto incrustado</option>
        <option value="offset">Sombra desplazada</option>
        <option value="tdeffect">Efecto 3D</option>
    </select><br />
    <input type="button" name="agregar" id="add" value="Agregar" />
    <input type="button" name="eliminar" id="remove" value="Eliminar" />
</form>
</article>
</section>
</body>
</html>
```

Guión 2: CSSClass.js

```
/**************************************************************************
* CSSClass.js: utilidades para manipular la clase CSS de un elemento HTML *
* Descripción: el módulo define un sólo objeto de espacios de nombres al  *
*              que se denominará CSSClass. Este objeto contiene funciones *
*              de utilidad para trabajar con el atributo class (propiedad *
*              className) de elementos HTML.                              *
* Argumentos:  Todas las funciones aceptan dos argumentos: el elemento e  *
*              que se va a probar o a manipular y la clase CSS c que se   *
*              va a probar, añadir o eliminar.                            *
**************************************************************************/
//Crear un objeto de espacio de nombres usando notación de literal de objeto
var CSSClass = {}; //Sintaxis declarativa para objetos

//Devolver true si el elemento e es miembro de la clase c
//y devolver false en caso contrario
CSSClass.is = function(e, c){
    if(typeof e == "string") e = document.getElementById(e);
    //Antes de realizar una búsqueda con regexp, optimizar casos comunes
    var classes = e.className;
    if(!classes) return false;    //No es miembro de ninguna clase
    if(classes == c) return true; //Es miembro sólo de una clase
    //En todos los demás casos, utilizar una expresión regular
    //para buscar c como palabra usando \b para coincidir con
    //cualquier límite de palabra. Note que la expresión devolverá
    //true o false dependiendo si el método search() encuentra o no
    //coincidencia con el patrón
    return e.className.search("\\b" + c + "\\b") != -1;
};

//Añadir la clase c al className del elemento e, si no está ya incluido
CSSClass.add = function(e, c){
    if(typeof e == "string") e = document.getElementById(e); //Id del elemento
    if(CSSClass.is(e, c)) return; //Si ya es miembro no hacer nada
    if(e.className) c = " " + c   //Si es necesario utilizar separador de
                                  //espacio en blanco si ya había otra clase
    //Añadir la nueva clase al final
    e.className += c;
};

//Eliminar todas las apariciones de la clase c del className
//del elemento e
```

```
CSSClass.remove = function(e, c){
    if(typeof e == "string") e = document.getElementById(e);
    //Buscar en className todas las apariciones de c y reemplazarlas
    //con "". Se utilizará \s* para coincidir con cualquier número
    //de caracteres de espacio en blanco y el modificador global "g"
    //para hacer que la expresión regular coincida con cualquier
    //número de apariciones.
    e.className = e.className.replace(new RegExp("\\b"+c+"\\b\\s*", "g"), "");
}
```

Guión 3: classname.js

```
function init(){
    var btnagregar = document.getElementById("add");
    var btneliminar = document.getElementById("remove");
    var hElement = document.getElementById("head");
    var clsName;
       //alert(hElement + " | " + clsName);
    //Controlar evento click sobre el botón Agregar
    if(btnagregar.addEventListener){
        btnagregar.addEventListener("click", function(){
             clsName                                                                    =
document.frmclass.classes.options[document.frmclass.classes.selectedIndex].value;
             CSSClass.add(hElement,clsName);
        }, false);
    }
    else if(btnagregar.attachEvent){
        btnagregar.attachEvent("onclick", function(){
                                                                                clsName      =
document.frmclass.classes.options[document.frmclass.classes.selectedIndex].value;
            CSSClass.add(hElement,clsName);
        });
    }
    //Controlar evento click sobre botón Eliminar
    if(btneliminar.addEventListener){
        btneliminar.addEventListener("click", function(){
                                                                                clsName      =
document.frmclass.classes.options[document.frmclass.classes.selectedIndex].value;
            CSSClass.remove(hElement,clsName);
        }, false);
    }
    else if(btneliminar.attachEvent){
        btneliminar.attachEvent("onclick", function(){
                                                                                clsName      =
document.frmclass.classes.options[document.frmclass.classes.selectedIndex].value;
            CSSClass.remove(hElement,clsName);
        });
    }
}

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

Resultado:

Ejercicio #2: El siguiente ejemplo muestra cómo crear un menú oculto que se muestre y se oculte al dar clic sobre un control de flecha al costado de la página.

Guión 1: hidetogglemenu.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Menú oculto</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="css/hidemenu.css" />
    <script src="js/togglemenu.js"></script>
</head>
<body>
<div id="box" class="hide">
    <ul id="tab">
        <li>
            <img id="arrow" src="img/arrowleft.png">
        </li>
    </ul>
    <div id="links">
        <div id="deco">
            <div class="bt">
                <a href="index.php">Inicio</a>
            </div>
            <div class="bt">
                <a href="tut.php">Tutoriales</a>
            </div>
            <div class="bt">
                <a href="about.php">Acerca de</a>
            </div>
            <div class="bt">
                <a href="contact.php">Contacto</a>
            </div>
        </div>
    </div>
</div>
</body>
</html>
```

Archivo 2: togglemenu.js

```
function init(){
    var imgElem = document.getElementById("box");
    var links = document.getElementsByTagName("a");
    if(imgElem.addEventListener){
        imgElem.addEventListener("click", function(){
            toggle("box");
        }, false);
    }
    else if(imgElem.attachEvent){
        imgElem.attachEvent("onclick", function(){
            toggle("box");
        });
```

```
    }
    for(var i=0; i<links.length; i++){
        links[i].setAttribute("href", "javascript:void(0)");
    }
}

function toggle(id) {
    var el = document.getElementById(id);
    var img = document.getElementById("arrow");
    var box = el.getAttribute("class");
    if(box == "hide"){
        el.setAttribute("class", "show");
        delay(img, "img/arrowright.png", 400);
    }
    else{
        el.setAttribute("class", "hide");
        delay(img, "img/arrowleft.png", 400);
    }
}

function delay(elem, src, delayTime){
    window.setTimeout(function() {
        elem.setAttribute("src", src);
    }, delayTime);
}

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

Resultado:

Ejercicio #3: Agregar, modificar, habilitar y deshabilitar reglas de estilo usando el DOM para manejo dinámico de Hojas de Estilo.
Guión 1: estilosDOM.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Manejo de reglas de estilo con el DOM</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="css/estilosdom.css" />
</head>
<body>
<header>
    <h1>Documento de prueba CSS</h1>
    <p>Esto es un párrafo de <strong>prueba</strong>.</p>
    <p>Aquí viene más texto <em>engañoso</em>.</p>
    <p>Hecho. No necesitamos <strong>seguir</strong> con esto</p>
</header>
<section>
<article>
<h3>Fin del documento de prueba</h3>
<form>
    <input type="button" value="Habilitar" id="enabled">
    <input type="button" value="Deshabilitar" id="disabled">
```

```
    <input type="button" value="Modificar regla" id="change">
    <input type="button" value="Borrar regla" id="remove">
    <input type="button" value="Añadir regla" id="add">
</form>
</article>
</section>
<script src="js/estilosdom.js"></script>
</body>
</html>
```

Guión 2: estilosdom.js

```
function init(){
    var btnenabled = document.getElementById("enabled");
    var btndisabled = document.getElementById("disabled");
    var btnchange = document.getElementById("change");
    var btnremove = document.getElementById("remove");
    var btnadd = document.getElementById("add");
    //Implementación del botón Habilitar reglas de estilo
    if(btnenabled.addEventListener){
        btnenabled.addEventListener("click", function(){
            document.styleSheets[0].disabled = false;
        }, false);
    }
    else if(btnenabled.attachEvent){
        btnenabled.attachEvent("onclick", function(){
            document.styleSheets[0].disabled = false;
        });
    }
    //Implementación del botón deshabilitar reglas de estilo
    if(btndisabled.addEventListener){
        btndisabled.addEventListener("click", function(){
            document.styleSheets[0].disabled = true;
        }, false);
    }
    else if(btndisabled.attachEvent){
        btndisabled.attachEvent("onclick", function(){
            document.styleSheets[0].disabled = true;
        });
    }
    //Implementación del botón modificar regla
    if(btnchange.addEventListener){
        btnchange.addEventListener("click", modifyRule, false);
    }
    else if(btnchange.attachEvent){
        btnchange.attachEvent("onclick", modifyRule);
    }
    //Implementación del botón borrar o quitar regla
    if(btnremove.addEventListener){
        btnremove.addEventListener("click", deleteRule, false);
    }
    else if(btnremove.attachEvent){
        btnremove.attachEvent("onclick", deleteRule);
    }
    //Implementación del botón añadir regla
    if(btnadd.addEventListener){
        btnadd.addEventListener("click", addRule, false);
    }
    else if(btnadd.attachEvent){
        btnadd.attachEvent("onclick", addRule);
    }
}
```

```
function modifyRule(){
    var styleSheet = document.styleSheets[0];
    if(styleSheet.rules){
        styleSheet.cssRules = styleSheet.rules;
    }
    if(styleSheet.cssRules[0]){
        styleSheet.cssRules[0].style.color = 'purple';
        styleSheet.cssRules[0].style.fontSize = '30pt';
        styleSheet.cssRules[0].style.backgroundColor = 'gold';
    }
}

function deleteRule(){
    var styleSheet = document.styleSheets[0];
    /* if(styleSheet.rules){
        styleSheet.cssRules = styleSheet.rules;
    }
    if(styleSheet.cssRules.length > 0){
        if(styleSheet.removeRule)
            styleSheet.removeRule(0);
        else if(styleSheet.deleteRule)
            styleSheet.deleteRule(0);
    } */
    //Haciendo compatible la propiedad rules
    if(styleSheet.rules){
        styleSheet.cssRules = styleSheet.rules;
    }
    if(styleSheet.cssRules.length > 0){
        if(styleSheet.removeRule)
            styleSheet.removeRule(0);
        else if(styleSheet.deleteRule)
            styleSheet.deletRule(0);
    }
}

function addRule(){
    var styleSheet = document.styleSheets[0];
    /* if(styleSheet.addRule){
              styleSheet.addRule("h3","{text-align:center;  font-family:\"Century  Gothic\";
font-size:18pt; color:Brown;}",4);
    }
    else if(styleSheet.insertRule){
            styleSheet.insertRule("h3   {text-align:center;  font-family:\"Century  Gothic\";
font-size:18pt; color:Brown;}",4);
    } */
    if("insertRule" in styleSheet){
              styleSheet.insertRule('h3  {text-align:center;  font-family:"Century  Gothic";
font-size:18pt; color:Brown;}', 4);
    }
    else if("addRule" in styleSheet){
              styleSheet.addRule('h3',  'text-align:center;  font-family:"Century  Gothic";
font-size:18pt; color:Brown;', 4);
    }
}

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

```
}
```

Ejercicio #4: Gráficos de barras con JavaScript. Con este ejemplo se ilustra cómo crear una aplicación que grafique valores que miden el uso de un determinado navegador o sistema operativo. La aplicación muestra varios controles de tipo botón de opción de formulario para escoger los resultados que se desean graficar, el color de los gráficos y la orientación de las barras.

Guión 1: graph.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Gráficos de barras con JavaScript</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="css/fonts.css" />
    <link rel="stylesheet" href="css/graph.css" />
    <link rel="stylesheet" href="css/formgraph.css" />
    <script src="js/graph.js"></script>
</head>
<body>
<header>
    <h1>Gráfico de barras</h1>
</header>
<section>
<article>
<form action="javascript:void(0)">
    <fieldset class="group">
    <legend><span class="labelgroup">Seleccione una gráfica:</span></legend>
    <input type="radio" name="type" id="browser" value="browser" checked="checked" />
    <label for="browser">Uso de navegadores</label><br />
    <input type="radio" name="type" id="platform" value="platform" />
    <label for="platform">Uso de Sistemas Operativos</label>
    </fieldset>
    <fieldset class="group">
    <legend>Elija un color:</legend>
    <input type="radio" name="color" id="colorred" value="lilRed.gif" checked="checked" />
    <label for="colorred">Red</label>
    <input type="radio" name="color" id="colorgreen" value="lilGreen.gif" />
    <label for="colorgreen">Verde</label>
    <input type="radio" name="color" id="colorblue" value="lilBlue.gif" />
    <label for="colorblue">Azul</label>
    </fieldset>
    <fieldset class="group">
    <legend>Elija una dirección:</legend>
    <input type="radio" name="direction" id="horizontal" value="horizontal"
checked="checked" />
    <label for="horizontal">Horizontal</label>
    <input type="radio" name="direction" id="vertical" value="vertical" />
    <label for="vertical">Vertical</label>
    </fieldset>
</form>
</article>
<article id="chartArea"></article>
</section>
</body>
</html>
```

Guión 2: graph.js

```
function initAll(){
    var radioButtons = document.getElementsByTagName("input");
    for(var i=0; i<radioButtons.length; i++){
```

```
        if(radioButtons[i].type == "radio"){
            if(radioButtons[i].addEventListener){
                radioButtons[i].addEventListener("click", chgChart, false);
            }
            else if(radioButtons[i].attachEvent){
                radioButtons[i].attachEvent("onclick", chgChart);
            }
        }
    }
    chgChart();
}

function chgChart(){
    //Definiendo el objeto y propiedades para el gráfico de navegadores (browsers)
    var bChart = new Object();
    bChart.name = "Uso anual de navegadores";
                                                    bChart.years            =            new
Array("2006","2007","2008","2009","2010","2011","2012","2013","2014","2015");
                    bChart.fieldnames     =     new     Array("Firefox","Chrome","Internet
Explorer","Safari","Opera");
    bChart.field1 = new Array(38.9,31.9,21.2,12.4,6.6,5.1,3,2.8,2.5,1.8);
    bChart.field2 = new Array(3.4,6.5,8.9,13.3,17.5,22.1,26.4,35.8,48.7,24.5);
    bChart.field3 = new Array(51.4,50.5,48.9,38.3,31.2,22.1,16.4,11.8,10.7,4.1);
    bChart.field4 = new Array(2.6,3.1,2.9,3.3,3.2,2.4,2.1,1.8,1.3,0.75);
    bChart.field5 = new Array(0.6,1.1,1.9,2.3,3.1,2.7,1.6,1.8,1.5,0.3);
     bChart.fields = new Array(bChart.field1, bChart.field2, bChart.field3, bChart.field4,
bChart.field5);
    //Definiendo el objeto y propiedades para el gráfico de Sistemas Operativos
    var jsChart = new Object();
    jsChart.name = "Uso anual de Sistemas Operativos";
                                                    jsChart.years            =            new
Array("2006","2007","2008","2009","2010","2011","2012","2013","2014","2015");
    jsChart.fieldnames = new Array("Windows","MAC-OS","Linux","Otros");
    jsChart.field1 = new Array(63.4,66.5,78.4,80.2,88.1,89.1,94,89.6,95.8,48.25);
    jsChart.field2 = new Array(10.2,8.5,7.4,8.2,6.1,7.3,5.6,7.6,8.8,9.5);
    jsChart.field3 = new Array(5.8,6.9,8.1,8.2,7.6,10.4,9.1,10.5,12.4,6.2);
    jsChart.field4 = new Array(3.2,4.1,3.5,3.8,4.6,5.2,3.8,3.2,4.8,1.75);
                                                    jsChart.fields           =            new
Array(jsChart.field1,jsChart.field2,jsChart.field3,jsChart.field4);

    //Dibujo del gráfico de barras
    var radioButtons = document.getElementsByTagName("input");
    var currDirection = getButton("direction");
    var imgSrc = "img/" + getButton("color");
    var thisChart = (getButton("type") == "browser") ? bChart : jsChart;
    var chartBody = "<h2>" + thisChart.name + "</h2>\n<table>\n";

    for(var i=0; i<thisChart.years.length; i++){
        chartBody += "<thead>\n";
        if(currDirection == "horizontal"){
                  chartBody  +=  "<tr>\n<th  rowspan='"  +  (thisChart.fields.length+1)  +  "'
class='horiz'>";
            chartBody += thisChart.years[i] + "\n";
            chartBody += "</th>\n";
            chartBody += "<td colspan='2'></td></tr>";
            for(var j=0; j<thisChart.fieldnames.length; j++){
              chartBody += "<tr>\n<td class='horiz'>";
              chartBody += thisChart.fieldnames[j] + "\n";
              chartBody += "</td><td>";
              chartBody += "<img src='" + imgSrc + "' height='15' ";
              chartBody += "width='" + thisChart.fields[j][i]*6 + "' ";
```

```
            chartBody += "alt='Barras horizontales' />";
            chartBody += "  " + thisChart.fields[j][i] + "</td>\n</tr>\n";
          }
        }
        else{
            chartBody += "<tr>\n<th rowspan='2' class='vert'>";
            chartBody += thisChart.years[i];
            chartBody += "</th>";
            for(var j=0; j<thisChart.fieldnames.length; j++){
                chartBody += "<td class='vert'>\n";
                chartBody += "<img src='" + imgSrc + "' alt='Barras verticales' ";
                chartBody += "hspace='10' width='15' ";
                chartBody += "height='" + thisChart.fields[j][i]*3 + "' />\n</td>\n";
            }
            chartBody += "</tr>\n<tr>\n";
            for(j=0; j<thisChart.fieldnames.length; j++){
                chartBody += "<td class='vert'>\n";
                chartBody += thisChart.fields[j][i] + "<br />\n";
                chartBody += thisChart.fieldnames[j] + "<br /><br /></td>\n";
            }
            chartBody += "</tr>\n";
        }
        chartBody += "</thead>\n";
    }
    chartBody += "</table>";
    document.getElementById("chartArea").innerHTML = chartBody;

    function getButton(buttonSet){
        for(var i=0; i<radioButtons.length; i++){
            if(radioButtons[i].name == buttonSet && radioButtons[i].checked){
                return radioButtons[i].value;
            }
        }
        return -1;
    }
}

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

Resultado:

Ejercicio #5: Ventanas movibles. Con este ejemplo se ilustra una aplicación que permite mover sobre la pantalla dos pequeñas ventanas, validando límites de ventana dentro del navegador.
Archivo 1: ventanascss.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Ventanas creadas con CSS</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="css/windows.css">
    <script src="js/loadevents.js"></script>
    <script src="js/drag.js"></script>
</head>
<body>
<!-- Crea una ventana de prueba con las CSS definida -->
<div class="window" style="left:10px;top:10px;z-index:10;">
<div class="titlebar" id="titlew1">
   Ventana de prueba
</div>
<div class="content">
    <p>
        La paleta de colores RGB (RVA en español) consta,
        básicamente, de tres colores primarios aditivos:
        Rojo-Verde-Azul. Estos colores primarios aditivos, en HTML,
        están representados por tres pares hexadecimales del tipo
        0xHH-HH-HH según el siguiente formato:
    </p>
    <p class="centrado">
        #RRGGBB (= #RRVVAA)
    </p>
    <p>
        Los valores que puede adoptar cada uno de los tres pares hexadecimales
        van del 0x00 (0 decimal) al 0xFF (255 decimal). Cuanto mayor sea el
        valor del par, tanto mayor será también la intensidad
        (matiz, brillo o claridad) del color correspondiente a ese par
        (y viceversa). Esto implica que el extremo inferior de la escala
        cromática parte de una intensidad (grado) de color mínima
        (nulo = par 0x00), pasa por una intensidad de color media (mediano =
        par 0x80 [128 decimal]) hasta llegar a una intensidad de color
        máxima (saturado = par 0xFF). El grado de más alta pureza
        (absoluto) de un color primario aditivo estará determinado por la
        presencia total del mismo (saturación = 0xFF) junto con la
        ausencia total (nulidad = 0x00) de los otros dos colores primarios
        aditivos.
    </p>
</div>
</div>
<!-- Crea otra ventana en otra posición superpuesta -->
<div class="window" style="left:75px;top:110px;z-index:20;">
<div class="titlebaractive" id="titlew2">
   Otra ventana
</div>
<div class="content translucent" style="background-color:#d0d0d0;font-weight:bold;">
    <p>
        Esta es otra ventana. Su valor <tt>z-index</tt> la pone encima
        de la otra ventana. Los estilos CSS hacen que su &aacute;rea
        de contenido se vea transparente en los navegadores que soportan
        esta caracter&iacute;stica.
```

```
    </p>
    <img src="img/3dboxmodel.jpg">
</div>
</div>
</body>
</html>
```

Guión 2: drag.js

```
/************************************************************
*  drag.js:  este script sirve para arrastrar elementos HTML  *
*            posicionados de forma absoluta.                  *
*                                                             *
*  El script define una sola función drag() diseñada para     *
*  llamarse desde un controlador de evento onmousedown. Los   *
*  eventos mousemove siguientes se moverán al elemento        *
*  especificado. Un evento mouseup terminará el arrastre.     *
************************************************************/

// elementToDrag es el elemento que recibe al evento mousedown
// o algún elemento contenedor. Tiene que colocarse de forma
// absoluta. Sus valores style.left y style.top se cambiarán
// basándose en el arrastre del usuario.
// event es el objeto event para el evento mousedown.
function drag(elementToDrag, event){
   //posición del ratón (en coordenadas de ventana) en la que
   //se inicia el arrastre.
   var startX = event.clientX, startY = event.clientY;
   //posición original ()en coordenadas del documento) del
   //elemento que se va a arrastrar. Como elementToDrag se ha
   //posicionado de forma absoluta, suponemos que su offsetParent
   //es el cuerpo del documento.
   var origX = elementToDrag.offsetLeft, origY = elementToDrag.offsetTop;
   //Aunque las coordenadas se calculen en sistemas de coordenadas
   //diferentes, podemos calcular todavía la diferencia existente
   //entre ellas y utilizarlas en la función moveHandler(). Esta
   //acción funciona porque la posición de la barra de desplazamiento
   //nunca cambia durante el arrastre.
   var deltaX = startX - origX, deltaY = startY - origY;
   //Registrar los controladores de eventos que van a responder a
   //los eventos mousemove y al evento mouseup que sigue a este
   //evento mousedown.
   if(document.addEventListener){  //Modelo de eventos DOM Nivel 2
      document.addEventListener("mousemove", moveHandler, true);
      document.addEventListener("mouseup", upHandler, true);
   }
   else if(document.attachEvent){  //Modelo de eventos de IE 5+
         //En el modelo de eventos de IE, se capturan los eventos
         //llamando al método setCapture() en el elemento.
         elementToDrag.setCapture();
         elementToDrag.attachEvent("onmousemove", moveHandler);
         elementToDrag.attachEvent("onmouseup", upHandler);
         //Tratar la pérdida de captura del ratón como un evento mouseup
         elementToDrag.attachEvent("onlosecapture", upHandler);
   }
   else{
         //En IE 4 no se puede utilizar attachEvent() o setCapture(),
         //por lo que se establecen controladores de eventos directamente
         //en el objeto document y se espera a que los eventos del ratón
         //que se necesitan se propaguen.
         var oldmovehandler = document.onmousemove;
         var olduphandler = document.onmouseup;
```

```
         document.onmousemove = moveHandler;
         document.onmouseup = upHandler;
   }
   if(event.stopPropagation) event.stopPropagation();  //Nivel 2 del DOM
   else event.returnValue = false;                     //IE

   //Controlador que captura los eventos mousemove cuando se está
   //arrastrando un elemento. Es el responsable de mover el elmento
   function moveHandler(e){
         if(!e) e = window.event;   //Modelo de eventos IE
         elementToDrag.style.left = (e.clientX - deltaX) + "px";
      elementToDrag.style.top = (e.clientY - deltaY) + "px";
      if(e.stopPropagation) e.stopPropagation();  //Nivel 2 del DOM
      else e.cancelBubble = true;                 //IE
   }

   //Controlador que captura el evento mouseup final que se produce
   //al final del arrastre
   function upHandler(e){
     var titleBar = document.getElementById('titlew1');
     var titleBarSecond = document.getElementById('titlew2');
     titleBar.className == 'titlebaractive' ? titleBar.className = 'titlebar' :
titleBar.className = 'titlebaractive';
      titleBarSecond.className == 'titlebar' ? titleBarSecond.className = 'titlebaractive'
: titleBarSecond.className = 'titlebar';
     if(!e) e = window.event;  //Modelo de eventos de IE
         //Anular el registro de los controladores de eventos de captura
         if(document.removeEventListener){
               document.removeEventListener("mouseup", upHandler, true);
               document.removeEventListener("mousemove", moveHandler, true);
      }
      else if(document.detachEvent){   //Modelo de eventos IE
         elementToDrag.detachEvent("onlosecapture", upHandler);
         elementToDrag.detachEvent("onmouseup", upHandler);
         elementToDrag.detachEvent("onmousemove", moveHandler);
         elementToDrag.releaseCapture();
      }
      else{   //Modelo de eventos de IE 4
               //Restablecer los controladores originales
               document.onmouseup = olduphandler;
               document.onmousemove = oldmovehandler;
      }
      //No permitir que se propague más el evento
      if(e.stopPropagation) e.stopPropagation();  //Nivel 2 del DOM
      else e.cancelBubble = true;                 //IE
   }
}
```

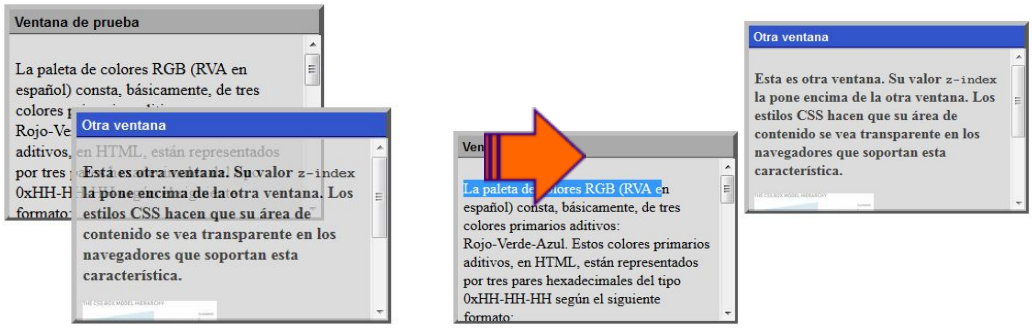
Guión 3: loadevents.js

```
function init(){
    var divw1 = document.getElementById("titlew1");
    var divw2 = document.getElementById("titlew2");
    if(divw1.addEventListener){
        divw1.addEventListener("mousedown", function(event){
             drag(this.parentNode, event);
        }, false);
    }
    else if(divw1.attachEvent){
       divw1.attachEvent("onmousedown", function(event){
             drag(this.parentNode, event);
      });
```

```
    }
    if(divw2.addEventListener){
        divw2.addEventListener("mousedown", function(event){
             drag(this.parentNode, event);
       });
    }
    else if(divw2.attachEvent){
        divw2.attachEvent("onmousedown", function(event){
             drag(this.parentNode, event);
        });
    }
}

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

Resultado: Después de cargar la página y mover las ventanas con el ratón por el área de la ventana del navegador.



V. DISCUSION DE RESULTADOS

1. Realice una aplicación con JavaScript que permita modificar el estilo de un párrafo de texto que inicialmente debe aparecer sin estilos y que al seleccionar con un control de formulario de tipo SELECT entre tres estilos que ud. debe crear y luego presionar un botón aplicar se modifique. Considere estilos para el tipo de fuente, color, alineación (izquierda, justificada y centrada). Además, agregue un botón para eliminar los estilos del párrafo para que el texto que contiene regrese al estado inicial; es decir, sin estilos aplicados. El párrafo debe tener por lo menos tres líneas con texto legible y coherente.
2. Realice una pequeña aplicación con JavaScript que procese los datos ingresados mediante una caja de texto de modo que al presionar un botón se vayan agregando los valores numéricos ingresados en un cuadro de lista. Su aplicación debe validar los datos para que sólo se ingresen valores numéricos en el cuadro de lista. Adicionalmente, debe existir otro botón que al presionarlo genere una gráfica de barras como la del ejemplo 4 del procedimiento donde muestre los valores ingresados como barras.

VII. BIBLIOGRAFIA

- Flanagan, David. JavaScript La Guía Definitiva. 1ra Edición. Editorial ANAYA Multimedia. 2007. Madrid, España.
- Tom Negrito / Dori Smith. JavaScrip y AJAX para diseño web. Editorial Pearson Prentice Hall. Madrid, España. 2007.

- 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.