

CAPÍTULO II

2. JAVASCRIPT (JSCRIPT)

2.1 Introducción

JavaScript es un lenguaje de programación creado para programar páginas Web. Aunque quizás su nombre pudiera sugerir lo contrario, Java y Javascript son lenguajes de programación diferentes y fueron pensados para diferentes propósitos.

Por una parte, Java es un potente aunque más complejo lenguaje de programación orientado a objetos creado por la empresa Sun Microsystems en la década de 1990. Este lenguaje es de propósito general lo cual significa que con éste se pueden crear aplicaciones de distinta naturaleza que no tengan ninguna relación con Internet.

Por otro lado, JavaScript es un modesto aunque sencillo lenguaje de programación destinado únicamente para la manipulación de páginas web. En sus inicios la compañía Netscape desarrolló este lenguaje bajo el nombre de LiveScript, sin embargo, poco después Netscape formó una alianza con Sun Microsystems (creador de Java) para desarrollar JavaScript. De esta alianza surgió un lenguaje más sencillo de utilizar que Java y útil para dotar de dinamismo a las aplicaciones web. De esta manera, el explorador Netscape 2.0 fue el primero capaz de interpretar código JavaScript.

La compañía Microsoft ha creado su propia versión de JavaScript llamada JScript la cual es empleada para programar y ejecutar páginas Web en uno de los exploradores de mayor uso: Internet Explorer.

JavaScript no es el único lenguaje en el que se puede programar una página web. VBScript (Visual Basic Script) es otro programa que puede ser utilizado para este propósito, sin embargo, aquí se tratará acerca de algunas características de JavaScript.

El conocido lenguaje HTML (Hyper Text Markup Language) que se traduce como Lenguaje de marcas hipertextuales es un lenguaje de presentación utilizado para estructurar el contenido de páginas web. Utilizando HTML es posible crear botones, colocar imágenes, etiquetas, cuadros de texto, crear hipervínculos, entre otras cosas. Sin embargo, en ocasiones el sólo conocimiento de HTML no es suficiente para darle a una página web la interactividad y flexibilidad que se requiere. En cambio el conocimiento de HTML unido al de JavaScript es una útil combinación que puede ayudar a los programadores a desarrollar interesantes páginas web.

JavaScript maneja las mismas nociones de otros lenguajes que manejan programación estructurada como Pascal y C++, entre otras cosas maneja estructuras de repetición, estructuras de decisión, funciones, variables, recursividad, etc. Los programadores de C++ encontrarán que existe similitud en la sintaxis de su programa con la

de JavaScript. Sin embargo, a diferencia de Pascal y C++ en JavaScript no es posible definir directamente una matriz de $n \times m$ ni definir el tipo de una variable de forma explícita.

Una de las cosas interesantes de los lenguajes HTML y JavaScript es que sus códigos pueden ser escritos sin ninguna dificultad en un archivo de texto. Es decir, un programador puede escribir su “sofisticado” código en el simple Block de Notas de Microsoft Windows. Lo único que necesita hacer al terminar de escribir su código es cambiar la extensión del archivo de *.txt a *.html y ya tendrá una página Web.

Existen programas especializados para escribir códigos de páginas Web como Front Page de Microsoft, Microsoft Script Editor o DreamWeaver, pero éstos no son imprescindibles para programar aunque sí brindan una serie de facilidades en relación al sencillo Block de Notas.

Por otro lado, una de las desventajas de la programación de código script es que la correcta ejecución de un programa depende del explorador del cliente. Lamentablemente los exploradores no tienen

un estándar por ello es posible que un código funcione perfectamente en Mozilla Firefox pero presente problemas al querer ser ejecutado en Internet Explorer, por ejemplo. Las instrucciones o funciones que un explorador reconoce no siempre serán reconocidas en otro. Es por esto que es conveniente que los programadores prevengan esta situación diseñando sus códigos de tal manera que dependiendo del tipo de navegador del cliente se ejecute una versión de sus páginas.

2.2 Referencia rápida

Para adherir lenguaje ya sea JavaScript o VBScript en un código html se lo realiza utilizando el elemento `<script language="lenguaje">`. Dentro de este elemento puede escribirse todas las instrucciones que se desee como lo muestra el simple código siguiente:

- `<HTML>`
- `<HEAD></HEAD>`
- `<BODY>`
- `<script language="javascript">`
- `<!--`

- `alert("JavaScript");`
- `//-->`
- `</script>`
- `</BODY>`
- `</HTML>`

Al igual que otros elementos de HTML el elemento `<script>` debe cerrarse con la instrucción `</script>`. De esta manera se puede alternar dentro de un mismo programa código HTML y código JavaScript, con la facilidad de poder escribir código JavaScript en cualquier parte del programa utilizando las instrucciones `<script></script>`. En el código anterior se utiliza la instrucción `alert("JavaScript")`, ésta permite mostrar mensajes por pantalla y es bastante útil en la depuración de programas. Si, por ejemplo, un programador no estuviera seguro del valor que toma la variable `p` en una parte del programa, puede incluir la instrucción `alert(p)`.

Las variables de JavaScript son sensibles a las mayúsculas, es decir la variable `p` será diferente de la variable `P`. En la sintaxis, Javascript tiene similitudes con C++. Aquí se verán algunas de las

instrucciones y características que han sido utilizadas para el desarrollo de la página web que se mostrará en el capítulo III.

Estructuras de selección y repetición

La estructura de selección que se utiliza es la *if . . . [else]*, ésta permite ejecutar una instrucción o un grupo de instrucciones si una determinada condición es verdadera y caso contrario ejecutar otra u otras instrucciones. La sintaxis para esta estructura condicional es la siguiente:

- *if* (condicion)
- { instrucciones;}
- [*else*
- { instrucciones;}]

Un programador de C++ reconocerá que esta instrucción es exactamente la misma que en su lenguaje. Los corchetes [] presentes en la sintaxis anterior indican que la instrucción *else* es opcional.

Las estructuras de repetición utilizadas son la condicional *do...while* y la instrucción *for*. La primera estructura permite repetir una instrucción o un bloque de instrucciones mientras una condición sea

verdadera. La segunda permite repetir una instrucción o un bloque de instrucciones mientras la variable relacionada a la estructura cumpla una determinada condición, esta estructura es utilizada para tareas comunes como la lectura de un arreglo. La sintaxis de la estructura *do . . . while* es la siguiente:

- do
- {
- instrucción 1;
- instrucción 2;
-
- instrucción n;
- }
- while (condición);

La sintaxis de la instrucción *for* es la siguiente:

- for (i=Viñicial; condición; incremento/decremento)
- {
- Instrucción 1;
- Instrucción 2;
-
- Instrucción n;
- }

En la sintaxis anterior *i* es la variable relacionada al bucle, ésta es utilizada para evaluar una condición que determina si el ciclo debe detenerse e incremento o decremento hace que la variable *i* aumente o disminuya según sea el caso. Este incremento o decremento no tiene que ser de 1 en 1. El siguiente ejemplo muestra 4 mensajes con los números 10,7,4 y 1 respectivamente:

- `<HTML>`
- `<BODY>`
- `<script language="javascript">`
- `<!--`
- `for (i=10;i>0;i=i-3)`
- `alert(i);`
- `//-->`
- `</script>`
- `</BODY>`
- `</HTML>`

La instrucción *for* del código anterior hace que la variable *i* vaya reduciendo su valor de 3 en 3 empezando en 10 y terminando en 1.

Puesto que JavaScript es sensible a las mayúsculas es necesario ser muy cuidadoso, si por ejemplo se escribe la siguiente instrucción:

- `If (a>5) alert("Es mayor");`

Internet Explorer mostrará error puesto que la instrucción es *if* y no *If*.

Declaración de variables:

En cuanto a la declaración de variables JavaScript es muy cómodo para el programador puesto que una variable no necesita declararse previamente antes de ser utilizada. Así mismo en la declaración de una variable no se especifica el tipo. Una variable puede ser declarada y recibir un valor al mismo tiempo o solamente ser declarada haciendo uso de la palabra reservada *var*, como en los siguientes ejemplos:

- `var x;`
- `var numero=0;`
- `var tasa=0.25, mes="Enero", dia;`

Como se muestra en las líneas anteriores varias variables pueden ser declaradas al mismo tiempo. También se puede escribir una asignación múltiple en una misma línea de código de la siguiente manera: `tipoA_xl=tipoA_lx=tipoB_xl=tipoB_lx=40;`. De esta manera la variable `tipoA_xl` tendrá el valor de 40 al igual que las demás y no será necesario haber declarado previamente estas variables.

Declaración de arreglos:

La declaración de arreglos al igual que la de variables no especifica tipo de dato y es muy cómoda porque el tamaño del arreglo se adapta a la cantidad de valores que se le haya ingresado.

Una forma sencilla de declarar los arreglos es la siguiente:

- `var nombre_de_arreglo = new Array();`

En la línea anterior se puede prescindir de la palabra reservada `var`.

Al asignarle a una variable `New Array()` lo que en realidad se está creando es un objeto de tipo `Array` con sus propios métodos y propiedades. Para referirnos al elemento `i` del arreglo “colores”, por ejemplo, se especifica `colores[i]`. Nuevamente, al igual que en C++ los arreglos en JavaScript empiezan con el subíndice 0, de tal manera que el primer elemento del arreglo `colores` es `colores[0]` y no `colores[1]`.

Una propiedad bastante útil de este objeto es `length` la cual devuelve el tamaño o cuantos elementos tiene un arreglo. Hay que

tomar en cuenta que si el arreglo *colores* tiene 5 elementos su propiedad *length* será 5 pero su último elemento será *colores[4]*. El código que sigue a continuación ilustra lo que hasta aquí se ha mencionado en relación a los arreglos:

- <HTML>
- <BODY>
- <script language="javascript">
- <!--
- colores = new Array("orange","red","cyan","brown","purple");
- cadena=colores[0];
- for (i=1;i<colores.length;i++)
- cadena=cadena+"-"+colores[i];
- alert(cadena);
- //-->
- </script>
- </BODY>
- </HTML>

Las líneas anteriores que es posible declarar un arreglo con el valor de cada uno de sus elementos a la vez:

```
colores = new Array("orange","red","cyan","brown","purple");
```

De esta manera la propiedad *length* del nuevo objeto llamado *colores* será igual a 5. La variable *cadena* es inicializada con el

valor string *“orange”* y la estructura *for* permite que se le vayan añadiendo los demás colores. La figura 2.1 muestra el resultado de este código anterior.

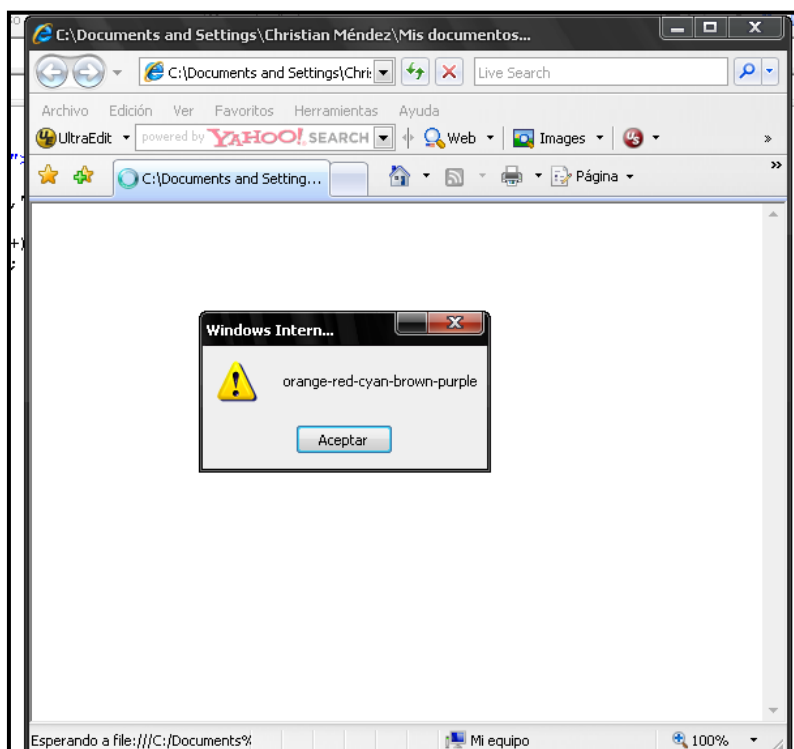


FIGURA 2.1 LECTURA DE UN OBJETO ARRAY()

El método `write()` del objeto `document`:

Una característica de JavaScript es que utilizando el método `write()` del objeto intrínseco `document` es posible escribir código HTML desde el propio código JavaScript. Por ejemplo si se deseara mostrar un botón se lo podría hacer comúnmente como sigue:

- `<input type='button' value='Calcular' name='btnTexto' onclick='Mostrar()' ID=Button1>`

Sin embargo también se podría crear el mismo botón desde el código JavaScript utilizando el método *write* de la siguiente forma:

- `document.write("<input type='button' value='Calcular' name='btnTexto' onclick='Mostrar()' ID=Button1>")`

Como se observa en el ejemplo anterior, utilizar el método *write* es como escribir directamente código HTML. Esto resulta muy útil puesto que se puede flexibilizar el código HTML. Por ejemplo, se puede crear un número variable de botones dependiendo del valor que toma una determinada variable.

La figura 2.2 muestra el código que flexibiliza la creación de botones que se mencionó anteriormente. Se observa que la cantidad de botones que se muestran depende de la variable *numero_de_botones*. La flexibilización va más allá, tanto la propiedad *value* como la propiedad *ID* de cada botón depende de la variable *i*.

```

uso del método write.html*
Eventos y objetos de cliente (No hay eventos)
<HTML><BODY>
<script language="javascript">
<!--
var numero_de_botones=7;

for (i=1;i<=numero_de_botones;i++)
document.write("<input type='button' value='botón "+i+" ID=Button"+i+"><p>");

//-->
</script>
</BODY></HTML>

```

FIGURA 2.2 CÓDIGO PARA LA CREACIÓN FLEXIBLE DE BOTONES A TRAVÉS DEL METODO WRITE().

En el código anterior basta con cambiar el valor de la variable *numero_de_botones* a 12 para que se creen en pantalla esa cantidad de botones. Se nota entonces, que Javascript permite al programador ir un poco más allá en el desarrollo de páginas web. La figura 2.3 muestra la ejecución del código de la figura 2.2. El uso del elemento `<p>` al final de la instrucción `document.write()` permite que la creación de botones sea en columna y saltando un espacio.

Declaración de matrices:

Lamentablemente en JavaScript no es posible declarar directamente arreglos multidimensionales. Para obtener arreglos multidimensionales es necesario declarar un arreglo de arreglos, es

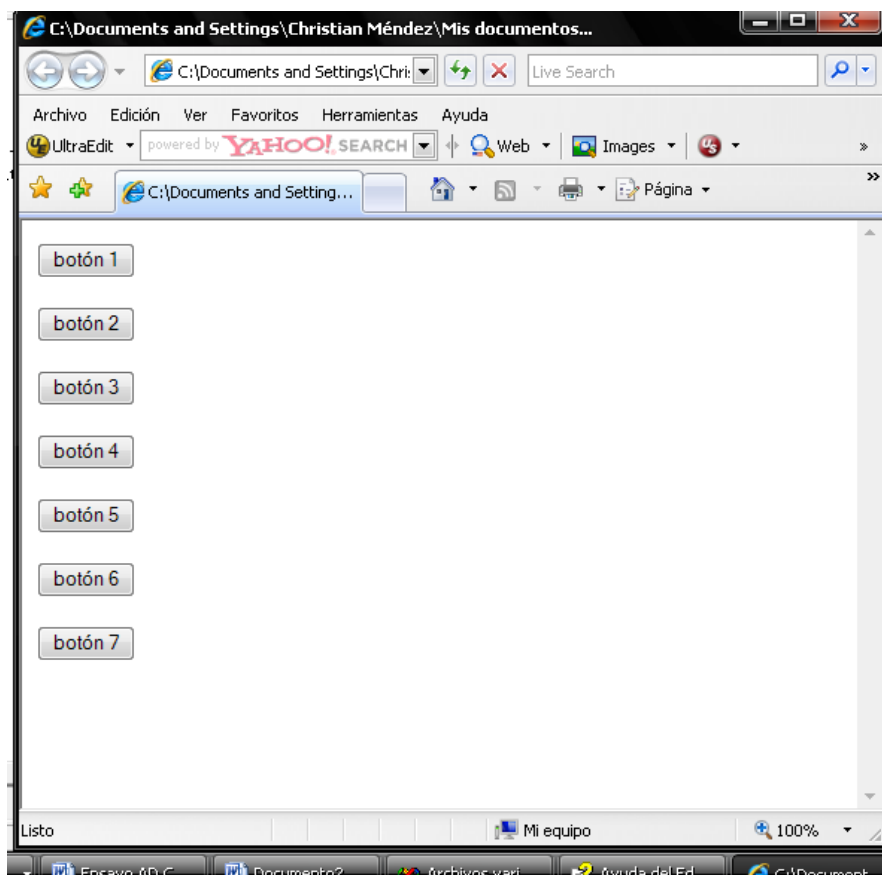


FIGURA 2.3 CREACIÓN FLEXIBLE DE BOTONES CON EL METODO WRITE().

decir un objeto de tipo Array donde cada elemento a su vez es un objeto Array. Esto se lo puede hacer de la siguiente forma:

```
a=new Array(new Array(),new Array(),new Array(),new Array());
```

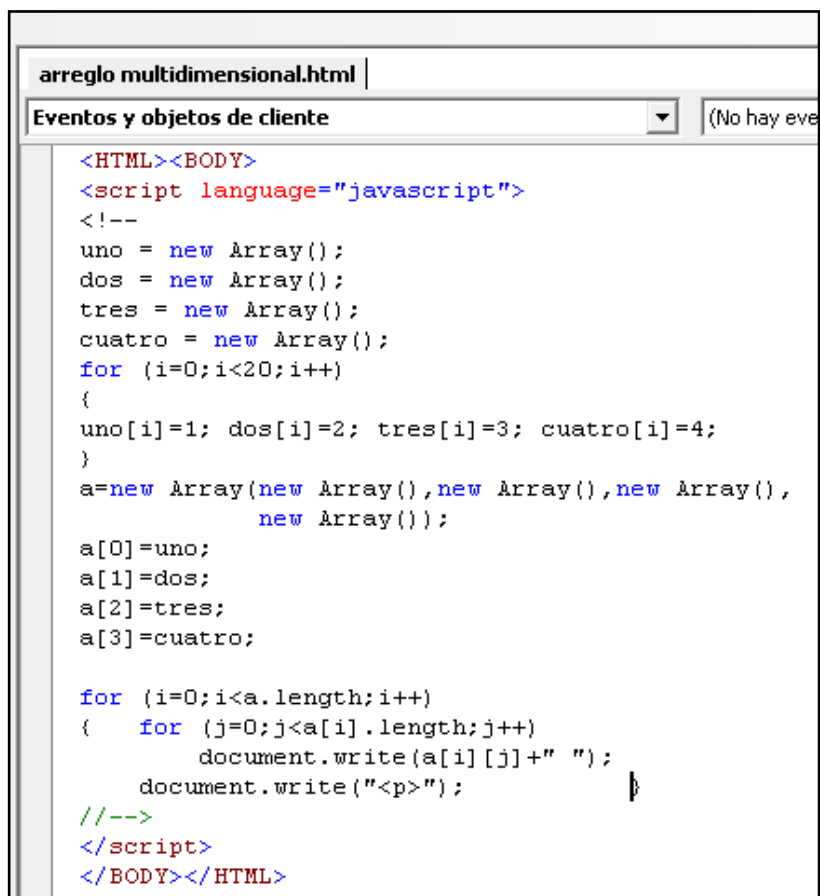
Un ejemplo de la creación de una matriz de 3x2 por ejemplo, sería el siguiente:

```
a=new Array(new Array(2,4),new Array(3,6),new Array(41,10));
```


Para leer cualquier elemento una matriz se podrá utilizar la notación `a[i][j]` así, el elemento `a[0][1]` en el ejemplo anterior será igual a 4.

La figura 2.4 muestra un ejemplo más elaborado que ilustra el empleo de un arreglo de arreglos para simular matrices. En este ejemplo se crea una matriz de 4x20 con filas de 1's, 2's, 3's y 4's respectivamente. En el ciclo *for* anidado que está al final del código se nota que la variable *i* llega hasta 1 menos que `a.length` mientras que *j* llega hasta 1 menos `a[i].length`. La propiedad `length` puede ser utilizada puesto que tanto `a` como `a[i]` son objetos *Array*.

También aparece en el código de la figura 2.4 la facilidad de realizar la asignación de un arreglo completo a una variable tipo *Array*. Por ejemplo, el arreglo *uno* es un conjunto de 20 unos. Este se lo asigna directamente al arreglo `a[0]` de la siguiente manera: `a[0]=uno`.



The image shows a browser window titled "arreglo multidimensional.html". The address bar shows "Eventos y objetos de cliente" and "(No hay eve". The main content area displays the following JavaScript code:

```
<HTML><BODY>
<script language="javascript">
<!--
uno = new Array();
dos = new Array();
tres = new Array();
cuatro = new Array();
for (i=0;i<20;i++)
{
uno[i]=1; dos[i]=2; tres[i]=3; cuatro[i]=4;
}
a=new Array(new Array(),new Array(),new Array(),
            new Array());

a[0]=uno;
a[1]=dos;
a[2]=tres;
a[3]=cuatro;

for (i=0;i<a.length;i++)
{
  for (j=0;j<a[i].length;j++)
    document.write(a[i][j]+" ");
  document.write("<p>");
}
//-->
</script>
</BODY></HTML>
```

FIGURA 2.4 CÓDIGO JSCRIPT PARA LA CREACIÓN DE UN ARREGLO MULTIDIMENSIONAL

El resultado visual del código anterior es un arreglo rectangular de 4 x 20 tal como lo muestra la figura 2.5.

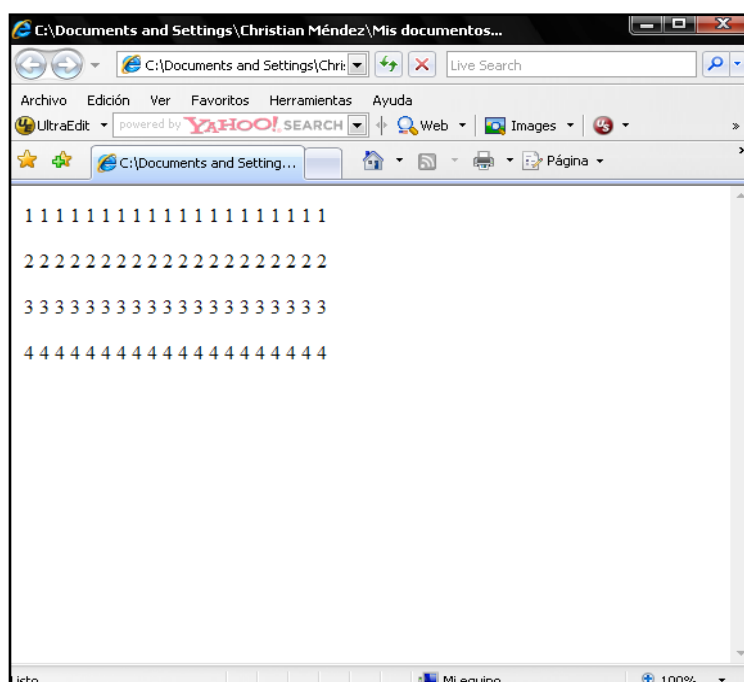


FIGURA 2.5 PRESENTACIÓN DE UNA MATRIZ DE 4 X 20

Uso de funciones:

Las funciones pueden ser declaradas en cualquier parte de un código JavaScript con la condición de que su invocación sea posterior a su declaración. También se puede invocar a estas funciones desde código HTML, al ocurrir un evento provocado por el usuario.

Los elementos de HTML contienen la posibilidad de actuar según un evento ocurra. Por ejemplo, el elemento contiene la

posibilidad de ejecutar una acción al pasar el ratón sobre su imagen. Es común que una página web actúe de alguna manera cuando se da click a uno de sus botones. Un botón con propiedad *name* igual a *btnCalcular* sería el siguiente:

- `<input type='button' value='Calcular' name='btnCalcular' onclick='Metodo()>`

En las líneas anteriores se muestra que el botón “btnCalcular” ejecutará la función “Metodo()” cuando se de click sobre él (evento onclick).

El siguiente código muestra el texto del botón btnTexto (su propiedad *value*) cuando se da click sobre él:

```
<HTML><BODY>
<input type='button' value='Calcular' name='btnTexto'
onclick='Mostrar()' ID=Button1>
<script language="javascript">
<!--
function Mostrar()
{ alert(Button1.value);}
//-->
</script> </BODY>
</HTML>
```

La figura 2.6 muestra el resultado dar click sobre el botón btnTexto.

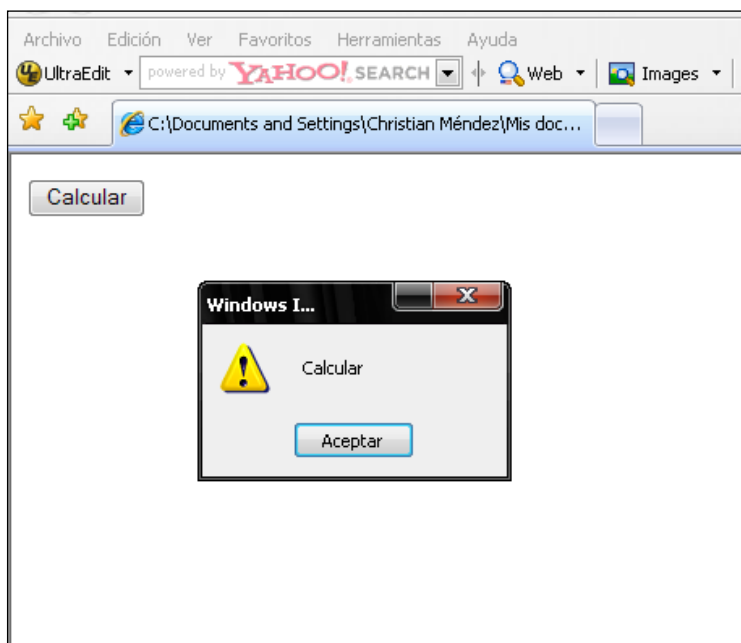


FIGURA 2.6 PRESENTACIÓN DE LA PROPIEDAD VALUE DE UN BOTÓN.

Uso de librerías:

En ocasiones los programadores hacen uso de clases junto con sus métodos y propiedades que ellos u otros han hecho anteriormente. De esta manera ellos pueden simplificar su trabajo. Las librerías son código JavaScript con extensión *.js. Las librerías *.js pueden ser llamadas utilizando la siguiente instrucción:

```
<script type="text/javascript" src="nombre_de_libreria.js"></script>
```

La página que se mostrará en el capítulo III utiliza 3 librerías: una para el manejo de matrices, las otras para la graficación de diferentes figuras y para el movimiento de imágenes respectivamente.

El objeto Math:

Puesto que la página que se ha diseñado en el presente trabajo aplica métodos de discriminación, el código para esta página requiere efectuar cálculos matemáticos, a este respecto resulta útil el objeto Math. Los métodos que son parte del objeto Math de JScript son los siguientes: *abs*, *acos*, *asin*, *atan*, *atan2*, *ceil*, *cos*, *exp*, *floor*, *log*, *max*, *min*, *pow*, *random*, *round*, *sin*, *sqrt*, *tan*. Algunos de estos métodos son utilizados en el desarrollo del programa.

El método `random()` es útil en este trabajo para generar conjuntos de datos parcialmente aleatorios. Este método devuelve un número pseudoaleatorio entre 0 y 1. Este método puede ser utilizado para generar un número aleatorio entre dos números cualesquiera a y b. La siguiente función llamada "aleatorio" genera un número pseudoaleatorio entre dos parámetros:

```
function aleatorio(a,b)
{ return Math.random()*(b-a) + a; }
```

El código que se muestra a continuación utiliza la función aleatorio para presentar 10 números pseudoaleatorios entre 4 y 9.

- <HTML>
- <BODY>
- <script language="javascript">
- <!--
- function aleatorio_entero(a,b)
- { return Math.round(Math.random()*(b-a) + a); }
- function aleatorio(a,b)
- { return Math.random()*(b-a) + a; }
- for (i=1;i<=10;i++)
- document.write("<p>" + aleatorio(4,9))
- //-->
- </script>
- </BODY></HTML>

El resultado del código anterior se muestra en la figura 2.7. Se observa en esta figura que los números que se presentan no son enteros, para que éstos sean enteros se puede utilizar otro método del objeto Math llamado round() que redondea una cifra haciéndola

entera. En el código anterior también se incluye una función llamada `aleatorio_entero(a,b)` que utiliza el método `round()` para producir números enteros y ésta se la puede incluir en la instrucción `document.write()` si se desea números enteros.

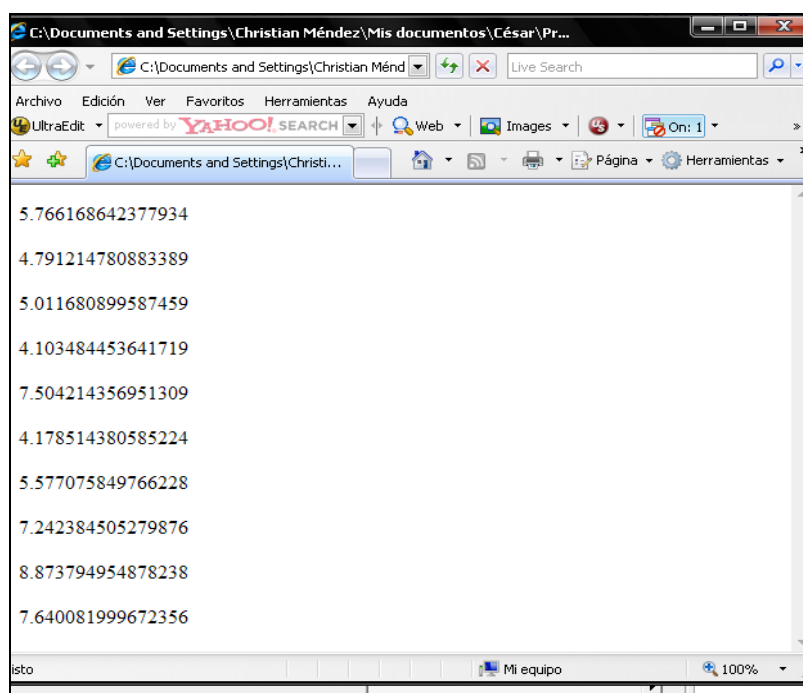


FIGURA 2.7 PRESENTACIÓN DE NÚMEROS UTILIZANDO EL MÉTODO `RANDOM()`

2.3 Librería JavaScript Vector Graphics: `wz_jsgraphics.js`

Esta es una librería desarrollada por Walter Zorn que contiene métodos para graficar una serie de figuras geométricas tales como:

rectángulos, elipses, líneas, polígonos, arcos. Aunque los detalles para el uso de esta librería podrían encontrarse en el sitio web www.walterzorn.com aquí se exponen algunos aspectos acerca de cómo usar esta librería en lo relacionado a la página web para el aprendizaje de análisis discriminante.

El primer paso es añadir la librería para que pueda ser utilizada. Esto se lo hace con la siguiente instrucción la cual ya se mencionó anteriormente:

```
<script type="text/javascript" src="wz_jsgraphics.js"></script>
```

Luego de esto es necesario definir un objeto de la clase `jsGraphics()` como si se definiera cualquier otro objeto:

```
var a = new jsGraphics();
```

Se puede definir un objeto `jsGraphics()` de dos maneras:

- ✓ Si el constructor no incluye argumento, el área de graficación será todo el documento: `var a = new jsGraphics();`
- ✓ Si el constructor incluye argumento, éste debe ser una capa previamente definida la cual será el área de graficación: `var a=new jsGraphics("miCapa");`

Será útil entonces definir una capa para que ésta represente el plano bidimensional donde se graficarán los puntos del conjunto de datos a discriminar.

Uno de los métodos que ofrece esta librería gráfica es `fillEllipse(X,Y,ancho,alto)`. Esta instrucción sirve para graficar una elipse pintada por dentro, la cual se encuentra inscrita en el rectángulo de ancho y alto especificados y cuyo vértice superior izquierdo está en la columna X y la fila Y. La figura 2.8 muestra el esquema de graficación de una elipse. El rectángulo de la figura se muestra en líneas punteadas porque no se grafica sino que sólo sirve de referencia para graficar la elipse.

El código para graficar una elipse en la capa1 sería el siguiente:

```
var plano=new jsGraphics("capa1");  
plano.fillEllipse(0,0,20,20);  
plano.paint();
```

De esta manera pudieran graficarse puntos (elipses muy pequeñas) en el plano bidimensional utilizando las coordenadas x,y de esta manera: `plano.fillEllipse(x,y,3,3)`; Puesto que son elipses

muy pequeñas (más bien círculos), en la instrucción anterior se nota que el ancho y alto de éstas es apenas 3.

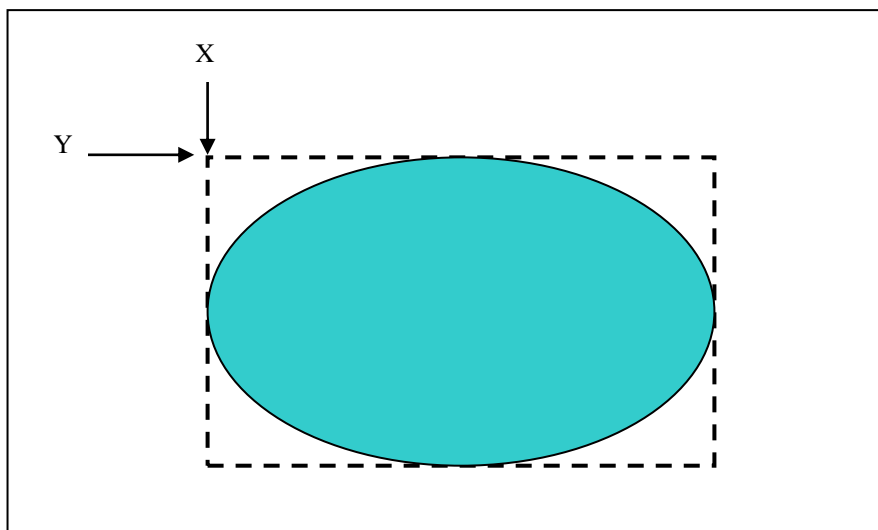



FIGURA 2.8 ESQUEMA DE GRAFICACIÓN DE UNA ELIPSE USANDO LA LIBRERÍA WZ_JSGRAPHICS.JS

La librería gráfica también permite ubicar una imagen cualquiera en la posición (x,y) como si se tratase de una elipse. Como se verá más adelante esto es muy útil para que los puntos graficados puedan ser desplazados por el usuario. La instrucción para realizar esto es la siguiente:

`drawImage("src", X, Y, ancho, alto);` donde `src` es el nombre de la imagen, por ejemplo "balloon.jpg"; y (X,Y) la posición donde se ubica el extremo superior izquierdo de la imagen.

Utilizando el método `drawImage` para una capa determinada y seleccionando las imágenes adecuadas se logran visualizar en el plano imágenes como si éstas fueran puntos, sin necesidad de graficar elipses. En la figura 2.9 se muestra la imagen  repetidas veces como puntos distribuidos en el plano.

El código para mostrar un punto parecido a los de la figura 2.9 es:

- `var plano = new jsGraphics("myCanvas");`
- `plano.drawImage("BD14868_.GIF",30,50,10,10);`
- `plano.paint();`

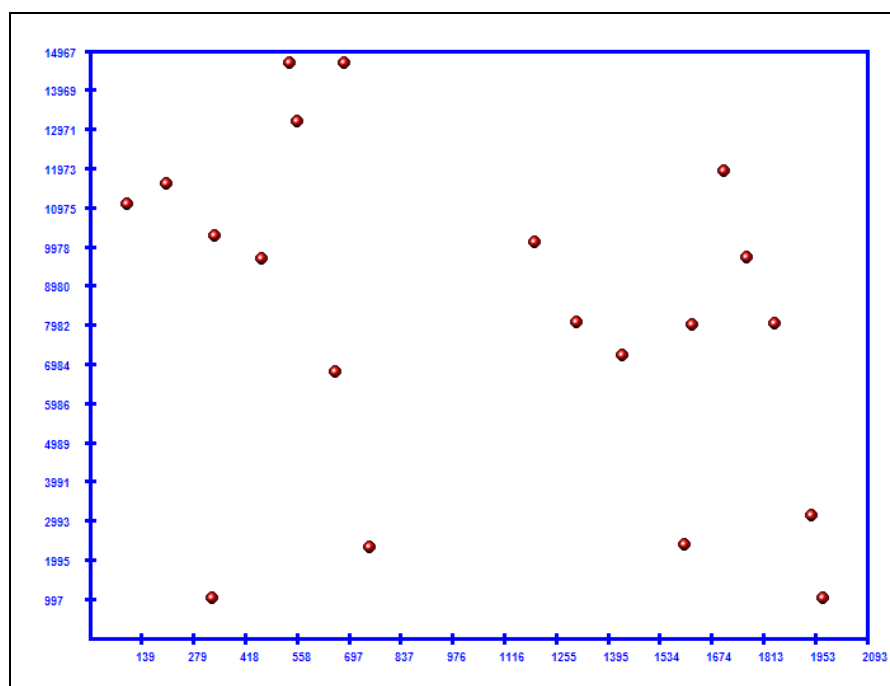


FIGURA 2.9 VISUALIZACIÓN DE IMÁGENES COMO PUNTOS EN EL PLANO UTILIZANDO EL MÉTODO `DRAWIMAGE()` DE LA LIBRERÍA `WZ_JSGRAPHICS.JS`

2.4 Librería JavaScript Drag & Drop: wz_dragdrop.js

Esta librería también ha sido desarrollada por Walter Zorn y se la utiliza para una de las partes esenciales de este proyecto. Aquí sólo se presentará como hacer para que una imagen cualquiera se haga movable. Mayores detalles de cómo utilizar esta librería podrían encontrarse en el sitio www.walterzorn.com.

Con el uso la librería wz_dragdrop.js es posible desarrollar páginas web en donde el usuario puede desplazar tanto imágenes como capas (layers).

El procedimiento para hacer que una imagen se pueda desplazar libremente por el usuario se resume de la siguiente manera:

- 1) Se incluye la librería apropiada
- 2) Se crea las imágenes que se desee utilizando HTML común, por ejemplo:

```
<IMG src="balloon.gif" name="balon">
```

```
< IMG src="face.gif" name="cara">
```

- 3) En las últimas líneas antes de </BODY> se debe incluir la instrucción: SET_DHTML con la propiedad "name" de las imágenes que se desea sean movibles, para hacer movibles las

imágenes creadas anteriormente la instrucción sería:

```
SET_DHTML("balon", "cara");
```

Si solamente se deseara que la imagen “balon” sea movable, se deberá incluir solamente el parámetro “balon” en la instrucción SET_DHTML.

Cuando se explicó la librería wz_graphics.js se mencionó que para simular la graficación de puntos se podría utilizar el método fillEllipse o el método drawImage. La ventaja de utilizar drawImage es que los puntos pueden hacerse movibles, pues éstos en realidad son pequeñas imágenes. De esta forma podríamos tener n puntos en el plano xy que pudieran ser desplazados por el usuario.