Bases de Javascript

Departamento Cencias de Computación
Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán

Ana Nieves Rodríguez

I - Presentación de JAVASCRIPT

Introducción

En clases anteriores hemos aprendido a programar en lenguaje HTML básico, pero ahora queremos dar un poco de dinamismo a nuestras páginas, así que el lenguaje estudiado es insuficiente porque solo nos sirve para presentar el texto, definir su estilo y algo más.
Hurgando un poco en la historia, vemos que el primer ayudante del lenguaje HTML fue Java mediante el uso de los Applets. Los Applets son pequeños programas incrustados en las páginas web, que fueron usadas por Netscape para programar en las páginas web. Netscape construyó sus navegadores compatibles con los Applets y desarrolló un lenguaje de programación llamado LiveScrip, que era más fácil de usar que java.
Luego surgió una alianza entre Netscape y Sun Microsystems (creador de java), para desarrollar en forma conjunta este nuevo lenguaje, de manera que se creó el JavaScript, útil dentro de una página web, fácil de programar aún para los neófitos y que permite crear programas pequeños dentro de una página, por otro lado se buscaba que JavaScript no necesite un kit de desarrollo (caso de java) ni que los scripts requieran de un compilador, ni realizar archivos externos al código HTML, como ocurre con los applets de java.
La sintaxis de JavaScript se parece a la de Java, pero son dos cosas distintas, a JavaScript se lo interpreta, mientras que a los programas de java se lo compila. Microsoft creó el JavaScript que se desarrolló para aprovechar las ventajas de Explorer.

Elementos básicos para programar en JavaScript

JavaScript no es un lenguaje de programación propiamente dicho, es un lenguaje de Script u orientado a documento, es una secuencia de comandos que se evalúa y ejecuta sobre la marcha, es usado para crear pequeños programas dentro de una página web que permiten interactuar con el usuario. Algunas características son:

- Permite interactuar con el usuario (a pesar de haberlo dicho anteriormente, creo que es necesario reiterarlo).
- Crear efectos especiales
- Se pueden validar datos de un formulario.
- Crear rollovers (cambia la imagen cuando se pasa el mouse).
- Crear navegadores desplegables.
- Se puede realizar la apertura de ventanas secundarias.
- Trabajar con la barra de estado, Etc.

Qué se necesita para programar un JavaScript?

1. Cualquier editor de textos como bloc de notas ya que los textos no necesitan ser enriquecidos, si lo hace con word, asegúrese de guardar el archivo como de solo texto.
2. Navegador compatible con javascript.

Las rutinas de JavaScript se escriben en un documento HTML, esto significa que en la página web conviven dos lenguajes de programación: HTML y JavaScript. Para que HTML interprete el lenguaje "invasor" (JavaScript) es necesario que se incorporen determinados tags propios de JavaScript de manera que el HTML reconozca que se trata de un script de JavaScript, así se deben incorporar los siguientes tags de acuerdo al navegador de que disponga, las posibilidades son:

1. <script>.....</script>
2. <script language="JavaScript">....</script>
3. <script type="text/javascript">.....</script>

Nota: Tenga en cuenta que JavaScript es sensible al ingreso, por lo tanto si ingresó <script> con minúscula, deberá realizar el cierre con minúscula, si lo hizo con mayúscula, su cierre debe ser con mayúscula, lo mismo sucede con el uso de variables.

Donde se escribe un JAVASCRIPT?

- La programación de JavaScript se realiza en un documento HTML
    - Se pueden incorporar en la cabecera entre las etiquetas <HEAD> </HEAD>
    - En el cuerpo de la página entre las etiquetas <BODY> </BODY>, a esto se le llama código incrustado.
    - Se pueden introducir varios script en una misma página con etiquetas <script> diferentes
- O se puede realizar un archivo externo.
    - Cuando existen varias funciones probadas y que pueden servir a mas de un programa se puede realizar un archivo externo.
    - El archivo externo normalmente tiene la extensión .JavaScript y se lo incluye de la siguiente manera, suponiendo que el archivo con las funciones se llame externo:

<script src="externo.JS">
</script>

Formas de Ejecutarlo

- **Directa**: se incorpora el script en el cuerpo o en la cabecera de una página html de modo que el navegador interprete cada línea.
- **Como respuesta a un evento**:
    - Un evento es la acción que realiza el usuario mientras visita una página. Por ejemplo el movimiento el mouse, pulsar un botón, la selección de un texto o la carga de un formulario. El programa captura el evento realizado por el usuario, realiza una tarea y emite una respuesta mediante el uso de manejadores de eventos. Estos manejadores son funciones propias de JavaScript.

Forma de ocultar un JAVASCRIPT en navegadores antiguos:

En el caso de navegadores antiguos que no comprenden JAVASCRIPT, se escribe el código como si fuese un

comentario, para ello se incorpora <!-- y -->, como hemos visto en lenguaje HTML.

```
<script type="text/javascript">
<! --Código javascript -- >
</script>
```

- Se puede indicar un texto alternativo si el navegador no permite código javascript como se muestra a continuación:

```
<NOSCRIPT>
El navegador no comprende los scripts
</NOSCRIPT>
```

Tener en cuenta que:

1. En la sintaxis de JAVASCRIPT Se respetan letras mayúsculas y minúsculas
2. Las instrucciones se separan con punto y coma, si se escriben dos instrucciones en la misma línea.
3. La otra forma de separarlas es mediante un salto de línea

Se recomienda usar siempre el punto y coma después de cada instrucción

Quien interpreta las instrucciones?

Incorporamos acá un concepto básico que se desarrollará en otras materias de la carrera: el modelo cliente/servidor. Este modelo es definido por IBM de la siguiente manera: "Es la tecnología que proporciona al usuario final acceso transparente a las aplicaciones, datos, servicios, o cualquier otro recurso, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicios hechos por estaciones de trabajo inteligentes o "clientes", resultan en un trabajo realizado por otros computadores llamados "servidores".

1. Es una relación entre procesos corriendo en máquinas separadas en donde:
   a. El Servidor es el proveedor de servicios.
   b. El Cliente es el consumidor de servicios.
2. El Cliente y el Servidor interactúan por medio de pasajes de mensajes.
   a. El Cliente solicita el servicio
   b. El Servidor proporciona una respuesta.

Por tanto existen dos actores: el servidor y el cliente. En esta arquitectura, la computadora de los usuarios (cliente), produce una demanda de información a la computadora que proporciona información (servidor) quien da respuesta a la demanda del cliente.

Nuestro primer JavaScript

Haremos nuestro primer programa en JavaScript, escriba el siguiente código y guárdelo con el nombre *primer_js.htm* en su disquete

```
<html><body>
<script type="text/javascript">
document.write("Hola programador!!!")
</script>
</body></html>
```

Ejemplo 1

Un programa es una secuencia de instrucciones.

En la página realizada hemos *incrustado* un script en el código HTML, porqué decimos esto? Porque el código del script se encuentra en el cuerpo de la página, o sea dentro de las etiquetas <body>..</body>

Explicación del ejemplo:

1. Se comienza con el tag <script>
2. La orden *document.write*, establece que se escriba en el documento html la frase "Hola programador" que debe ir entre comillas dobles o simples.

Ejemplo mediante un evento:

```
<html><body>
<FORM>
<INPUT TYPE="button" NAME="boton"
VALUE="Presione" onClick="document.write('Hola
Programador!!!')">
</FORM>
</body></html>
```

Ejemplo 2

En este caso incorporamos una expresión javascript en un formulario en donde declaramos un botón con el texto presione. Al presionar el botón escribe en el documento la frase Hola Programador.
La expresión onClick es JavaScript, fíjese un detalle: luego de onClick = la expresión documentr.write está entre comillas dobles y el texto se encuentra en comillas simples.

Desde un evento para que muestre una ventana de alerta:

```
<html><body>
<FORM>
<INPUT TYPE="button" NAME="boton"
VALUE="Presione" onClick="alert('Hola
Programador!!!')">
</FORM>
</body></html>
```

Ejemplo 3

Este ejemplo es similar al anterior pero usando una ventana de alerta.

II - Elementos básicos

- Al escribir un programa JavaScript, se deben respetar las mayúsculas y las minúsculas.
- Las órdenes o instrucciones dadas a JavaScript se separan con punto y coma, si se escriben dos instrucciones en la misma línea.
- Otra forma de separarlas es mediante un salto de línea
- Se recomienda usar siempre el punto y coma después de cada instrucción.

Comentarios:

Cuando se escribe un programa, muchas veces es necesario documentar las tareas que realiza, muchas veces son líneas importantes que no hacen tareas de programación en sí:

```
<script type=”text/javascript”>
//comentario de una línea
/*Comentario que puede abarcar
Más de una línea*/
</script>
```

Ejemplo 4

Textos

Los textos que deseamos que se muestren en el programa deben ir encerrados entre comillas que pueden ser dobles o simples.

```
<HTML><HEAD></HEAD>
<script type=”text/javascript”>
document.write('texto que saldrá en el script')
</script></BODY></HTML>
```

Ejemplo 5

En el ejemplo propuesto, se realiza un script que escribe en el documento HTML una frase. Para que realice esta tarea escribimos la instrucción *document.write* que significa que escriba en el documento lo que se encuentra dentro del paréntesis. Cuando combinamos frases y contenido de variables, se usa el operador de concatenación +, como veremos mas adelante.
Uso de Document: Luego veremos que document es un objeto. El document es muy usado en JavaScript, en casos en que no es conveniente trabajar con ventanas.

Variables y Operadores en JavaScript

En este capítulo veremos algunos elementos muy importantes a la hora de trabajar con JavaScript acompañados de ejemplos que ustedes podrán realizar para probar lo que se dice. Los ítems a tratar son:

1. Qué son las Variables
2. Qué son los operadores

1. Qué son las variables:

Una **Variable** es un espacio de memoria en donde se alberga un dato. Las variables deben tener un nombre, que puede ser cualquiera, por ejemplo pepe, respuesta, consulta, etc.

- Se forman con caracteres alfanuméricos o guión bajo.
    - ejemplos: sumando_1, A, B, sumando2, _A, etc.

  - DEBEN comenzar con un carácter alfabético o guión bajo.
- Prohibidos:
  - +, $, espacio
  - palabras reservadas como for, return, while, if, var, etc

Se aconseja que el nombre represente al dato que alberga, de esta manera es fácil convocarlas.
Ejemplo:
A=14
B=13
Suma = A + B
En este ejemplo tenemos tres variables: A, B y suma.

Declaración de variables

No es obligatorio, aunque es conveniente, declarar una variable mediante la indicación de la palabra reservada var, de la siguiente manera:

- Var suma1
- Var suma2, suma3 → varias variables separadas por coma.

Se puede asignar el valor a una variable de la siguiente manera:

- Var suma1 = 14

Ambito de las variables

Ámbito de la variable, es el lugar en donde se encontrará disponible esa variable, existen variables globales y locales:

- Globales: se encuentran disponibles en toda la página, incluidos los manejadores de eventos.
- Locales: cuando son declaradas en un tramo de programa acotado por { }, por ejemplo una función o un bucle.

NOTA: Si no se usa la palabra var, la variable será global.

Tipos de datos

Los datos pueden ser de diferentes tipos, por ejemplo:

- Números
- Cadenas de caracteres: texto que puede estar compuesto de letras y números, encerrado en comillas dobles o simples.
- Valores booleanos: las variables de este tipo pueden tomar solo los valores true o false, o sea o es verdadera (true) o falsa (false).

Tipos de variables

El Operador typeof devuelve una cadena de textos que describe el tipo de variable que estamos comprobando.

```
Var confirma = true
Var numero = 67
document.write("tipo1:" + typeof confirma + "<br>")
document.write("tipo2:" + typeof numero + "<br>")
```

Ejemplo 6

2.Qué son los operadores

Los operadores son símbolos que se utilizan para trabajar con las variables, tal como se trabaja en la aritmética elemental, los signos mas y menos son

operadores. A continuación se listan los operadores usados en JavaScript con un ejemplo correspondiente.

Aritméticos:

- Suma: +
- Resta: -
- Multiplicación: *
- División: /
- Resto de la división: %

```
<HTML><HEAD><TITLE>LABORATORIO
1</TITLE></HEAD>
<script>
document.write('texto que saldrá en el script<br>')
var a=6 , b=2
c= a*b;
d= a/b;
e= a%b;
f= a+b;
document.write('a= '+ a + '<br>b= ' + b + '<br>c= ' + c);
document.write('<br>d= ' + d +'<br>e= ' + e + '<br>f='+f)
</script>
</BODY></HTML>
```

Ejemplo 7

NOTA1: en muchos navegadores, el operador de suma es interpretado como la concatenación de caracteres como se muestra en el siguiente ejemplo

```
<script type="text/javascript">
var a,b,s;
a=prompt("ingrese numero a","");
b=prompt("ingrese numero b","");
s=a+b;
alert(s);
</script>
```

Ejemplo 8

Vista en el Browser:

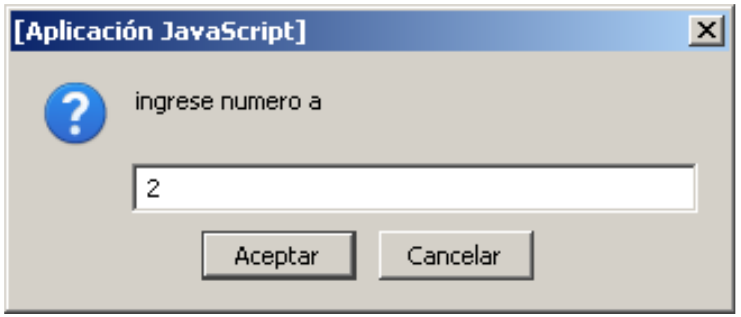


Figura 1

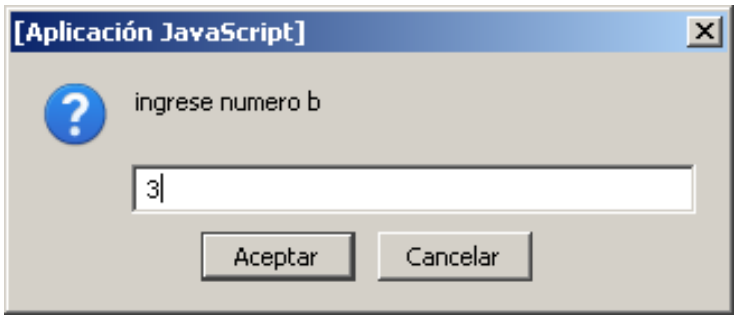


Figura 2

Resultado:

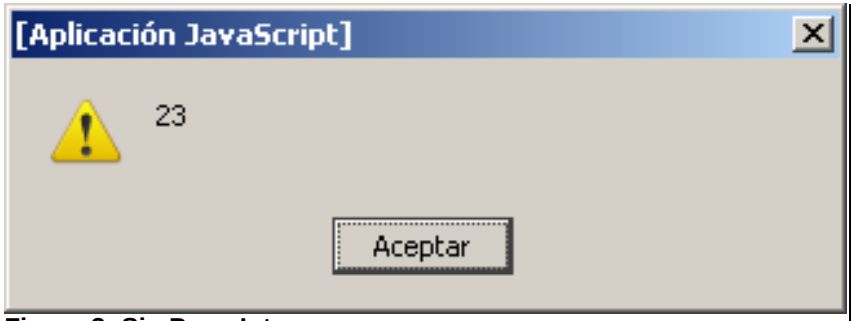


Figura 3: Sin ParseInt

Para solucionar momentáneamente se hará uso de 2 funciones propias de JavaScript: ParseInt o ParseFloat, las que se describen en el capítulo de funciones

```
<script type="text/javascript">
var a,b,s;
a=prompt("ingrese numero a","");
b=prompt("ingrese numero b","");
    s=parseInt(a)+parseInt(b);
    alert(s);
</script>
</body>
</html>
```

Ejemplo 9: con ParseInt

Unarios:

- Incremento de la variable en una unidad: ++, →(A++)
- Decremento de la variable en una unidad: -- → (A--)

Ejemplo de operadores unarios:

```
<HTML><HEAD>
<TITLE>LABORATORIO 1</TITLE>
        </HEAD>
<script type=”text/javascript”>
document.write('texto que saldrá en el script<br>')
var a=3 , b=1
a++; b--
//se incrementa la variable  a y se decrementa b
document.write('a= '+ a + '<br>b= ' + b)
</script>
</BODY>
</HTML>
```

Ejemplo 10

De Asignación

- = Asignación simple
- += Asignación con suma
- -= Asignación con resta
- *= Asignación con multiplicación
- /= Asignación con división
- %= Obtiene el resto y asigna

Ejemplo de asignación:

```
<html>
<body>
<script type="text/javascript">
num1 = 2; //asignación simple
document.write('Numero original--> '+num1+'<br>');
```

```
num1 += 4; // asignación con suma, en este caso se
suma 4 al número num1
document.write('Numero mas cuatro (num1 += 4)-->
'+num1+'<br>');
num1 -=1; // asignación con resta, en este caso se resta
1 al número num1
document.write('Numero menos uno (num1  -=1)-->
'+num1+'<br>');
num1  %=2;// resto de la división del número por dos
document.write('Resto de dividir en dos (num1  %=2)-->
'+num1+'<br>');
</script>
</body>
</html>
```

Ejemplo 11

Vista en el browser

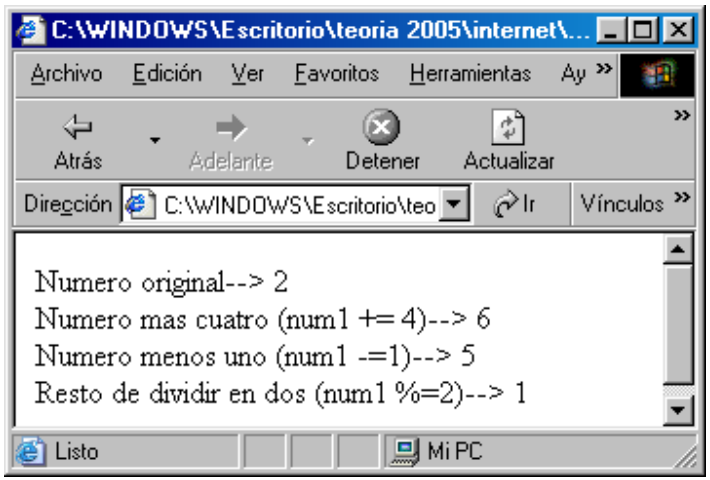


Figura 4: asignación

Operadores de incremento

Estos operadores se pueden colocar antes o después de la expresión, pero solo modifican si se encuentran delante, ejemplo:

```
<html>
<body>
<script type="text/javascript">
a = 1;
document.write('a = 1<br>')
b = ++a;
document.write('(b = ++a) a--> '+a +'<br>b-->'+ b +'<br>--
-------<br>');
a = 1;
document.write('a = 1<br>')
c  =a++;
document.write('(c = a++) a--> '+ a +'<br>c-->'+ c +' b--
>'+b);
</script>
</body>
</html>
```

Ejemplo 12

Operadores lógicos

- ! : operador de negación
- && : operador AND (si los dos valores son verdaderos, el resultado es verdadero)
- || : Operador OR (si uno de los valores es verdadero, devuelve el valor verdadero)

```
<html><body>
<script type="text/javascript">
num1 = true;
document.write('Num1: '+num1+'<br>-------<br>');
num1 = !num1;
document.write('Negación (num1 = !num1):
'+num1+'<br><b>---Vuelvo num1 al valor original---
</b><br>');
num1 = !num1;
num2  = true;
num3 = false;
document.write('Num1= '+num1+'<br>Num2=
'+num2+'<br>Num3= '+num3+'<br>--------<br>');
num4 = num1 && num2;
document.write('And con true (num4 = num1 && num2):
'+num4+'<br>-------<br>');
num4 = num1 && num3;
document.write('And con false (num4 = num1 && num3):
'+num4+'<br>-------<br>');

//or
num4 = num1 || num2;
document.write('OR dos true (num4 = num1 || num2):
'+num4+'<br>-------<br>');
num4 = num1 || num3;
document.write('OR con false (num4 = num1 || num3):
'+num4+'<br>--------<br>');

</script></body></html>
```

Ejemplo 13

```
<html><body>
<script type="text/javascript">
a = false;
b = false;
_a = !a;
_b = !b;
document.write('<center><table border=
2><caption><h2>Función AND</h2></caption>');
document.write('<tr><td>A</td><td>B</td><td>Y</td></tr
><tr><td>');
//00
document.write(a +'</td><td>');
document.write(b +'</td><td>');
c =  a&&b;
document.write(c +'</td></tr><tr><td>');
//0 1
document.write(a +'</td><td>');
document.write(_b +'</td><td>');
c =  a&&_b;
document.write(c +'</td></tr><tr><td>');
//1 0
document.write(_a +'</td><td>');
document.write(b +'</td><td>');
c =  _a&&b;
document.write(c +'</td></tr><tr><td>');
//1 1
document.write(_a +'</td><td>');
document.write(_b +'</td><td>');
c =  _a&&_b;
document.write(c +'</td></tr><tr><td>');
```

```
//fin de tabla
document.write('</table></center>');
</script></body></html>
```

Ejemplo 14

Operadores de Comparación

Se usan para comparar los valores de las variables.

- x == y Devuelve verdadero si x es igual a y
- x != y devuelve verdadero si x es distinto de y
- x > y devuelve verdadero si x es mayor que y
- x < y devuelve verdadero si x es menor que y
- x <= y devuelve verdadero si x ees menor o igual que y

Orden de Precedencia de los operadores

- Orden en el que se realizarán las operaciones
- Se evalúan de izquierda a derecha
  1. ( ) [ ] .: Paréntesis, corchete y punto de objeto
  2. ! - ++ --: Negación, negativo, incremento.
  3. * / %: Multiplicación, división, módulo
  4. + -: Suma, resta
  5. Operadores condicionales: menor, <=,mayor, >=
  6. Condicionales de igualdad y desigualdad: = =, !=
  7. Lógicos booleanos: && ||
  8. Asignación: =, +=, -=, *=, /=, etc

Anexo 1

Ya se vieron algunos ejemplos, en los que hemos usado la orden document.write(), que le dice a JavaScript que escriba lo que se encuentra entre paréntesis en un documento HTML.
Este anexo es para los más ansiosos y quieren ver un resultado inmediatamente.

Efectos rápidos

Entre los efectos rápidos haremos uso del objeto window, en el que se mostrarán ventanas de:

1. Alert
2. Confirm
3. Prompt

1. Uso de ventanas de alerta

Veamos algunos ejemplos simples y rápidos:

- Copie el siguiente código con el bloc de notas y guarde en una carpeta llamada JAVASCRIPT en su disquet, con el nombre segundoJS.html

```
<html>
<body>
<script type="text/javascript">
wdow.alert("Bienvenido a Laboratorio 1 - Carrera de
Programador!")
</script>
</body>
</html>
```

Ejemplo 15

en este caso incrustamos el código de javascript dentro del body de una página web.
Window es un objeto usado por javascript, y alert es un método de window que al decir window.alert le decimos

que muestre una ventana de alerta con el mensaje que se encuentra en el paréntesis, como en este caso se trata de un texto, se debe incluir entre comillas dobles o simples. Cabe aclarar que la orden de alert puede ir sola siempre y cuando se encuentre dentro del bloque <script>...</script>, ya que JavaScript asume que se trata de una ventana de alerta.
Al ejecutar la página, se verá la siguiente pantalla:

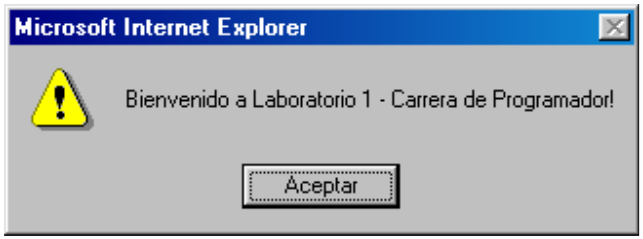


Figura 5: ventana de alerta

2. Interactuando con el usuario – ventana de confirmación

En este apartado trataremos de mostrar dos ejemplos simples en los que el programador interactúa con el usuario.

```
<html><body>
<script type="text/javascript">
resp = confirm("Carrera de Programador?");
alert(v)
</script>
</body></html>
```

Como en el caso anterior, confirm se puede usar con window.Confirm. El detalle que se debe tener en cuenta en estos casos, es que confirm solicita una respuesta que se carga en una variable que hemos llamado “resp”, los valores posibles que tomará la variable “resp” son True o False. Para saber el valor elegido, se muestra en una ventana de alert, la visualización se muestra de la siguiente forma:

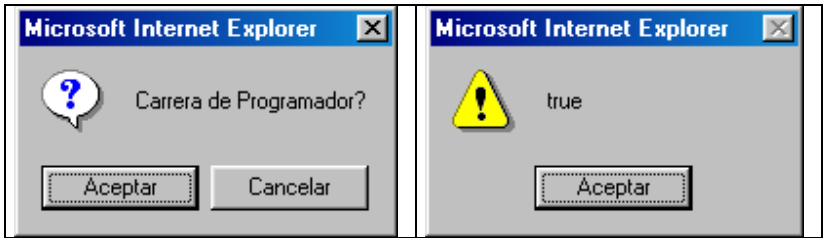


Figura 6: ventana Confirm

3. Ventana de Prompt

Otro ejemplo típico que permite interactuar con el usuario, es el de solicitar por pantalla el ingreso de datos. Esta tarea se realiza usando la ventana prompt de la siguiente forma:

```
<html><body>
<script type="text/javascript">
nombre = prompt("Ingrese su nombre", "")
alert(v)
</script>
</body></html>
```

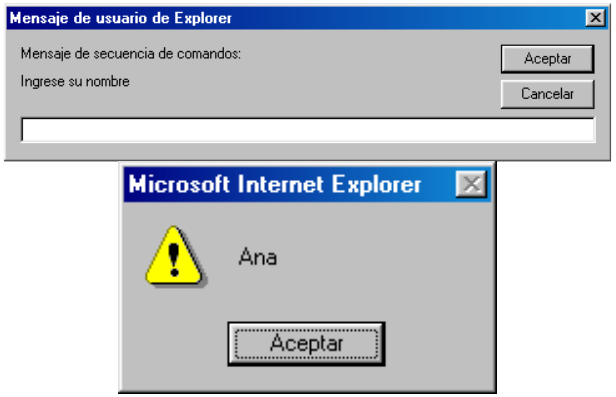


Figura 7: Ventana de Prompt

El comando Prompt posee dos lugares a los que llamamos parámetros.
Los parámetros, como se verá mas adelante, son variables que se ingresan a un programa, separados por coma:

- *Parámetro 1*: en este lugar se escribe la frase que deseamos ver en la pantalla del prompt, en el caso del ejemplo: “Ingrese su nombre”,
- *Parámetro 2*: en el que podemos escribir un texto que deseamos que se muestre como valor predeterminado. Este valor puede ser ignorado pero mostrará la palabra **undefined** o sea valor no definido, si no queremos que se muestre esta palabra, simplemente se abren y cierran las dobles comillas, en el caso que deseamos ver una palabra predeterminada, la escribimos entre comillas dobles.

III - Flujo del Programa

Estructuras de control

Toma de decisiones o bifurcación condicionales

Realiza una u otra acción dependiendo del valor de la variable que se evalúe, en esta categoría se encuentran las estructuras: IF y SWITCH.

Estructura IF

La estructura mas sencilla es la estructura IF y se utiliza de la siguiente manera:

```
If (expresión) {
        Acciones a realizar en caso positivo
} else {
Acciones a realizar en caso negativo
}
```

Ejemplo 16

En el siguiente ejemplo se solicita al usuario una respuesta, mediante el uso de una ventana de confirmación.
Debe tener en cuenta que: **if (*conf*)** significa → si la variable ***conf*** posee el valor true (verdadero), entonces escriba en el documento con la orden document.write: "Presionó Aceptar", de otro modo (**else**) escriba en el documento → document.write: "Presionó Cancelar"

Un detalle que debe tener en cuenta es que si dentro de la rama if posee mas de una orden, estas deben encerrarse entre llaves: { }, lo mismo sucede con el else, siendo de buena costumbre usar las llaves siempre tanto en if como en else.
Recuerde que el **else** es opcional.

```
<html><body>
<script type="text/javascript">
<!--
conf = confirm("Desea Seguir?");
document.write(conf+"<br>");
if (conf) {
document.write("Presionó Aceptar");
}
else {
document.write("Presionó Cancelar");
}
-->
</script>
</body></html>
```

Ejemplo 17

Estructura Switch

Bifurca según el valor que tome la variable, se utiliza para seleccionar un valor entre diferentes alternativas.

```
Switch (expresión) {
 case valor1:
                Tarea1
                break
```

```
case valor2:
            Tarea2
            break
Default:
            tarea
}
```

La expresión a evaluar se escribe junto a la palabra switch → Switch (expresión)

- Si la expresión tiene el valor1, realiza la tarea1.
- Para que salga de la estructura se incorpora la orden break, de otro modo continúa realizando la tarea2.
- Si no encuentra valor válido, ingresa a la opción default.

Pruebe el siguiente ejemplo:

```
<html>
<body>
<SCRIPT>
<!--
num = prompt("numfoto","");
switch (num){
case "1":
        fto = 'foto'+num+'.jpg';
        document.write(fto);
        document.write("<img src="+fto+">");
        break;
case "2":
        fto = 'foto'+num+'.jpg';
        document.write(fto);
        document.write("<img src="+fto+">");
        break;
case "3":
        fto = 'foto'+num+'.jpg';
        document.write(fto);
        document.write("<img src="+fto+">");
        break;
case "4":
        fto = 'foto'+num+'.jpg';
        document.write(fto);
        document.write("<img src="+fto+">");
        break;

default:
        alert("Esa foto no existe")
-->
</SCRIPT>
</body>
</html>
```

Ejemplo 18: Este ejemplo solicita el ingreso de un número de foto y la muestra

Vista en el Browser

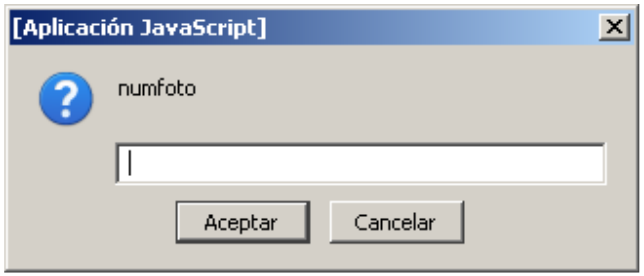


Figura 8: ejemplo con prompt y switch

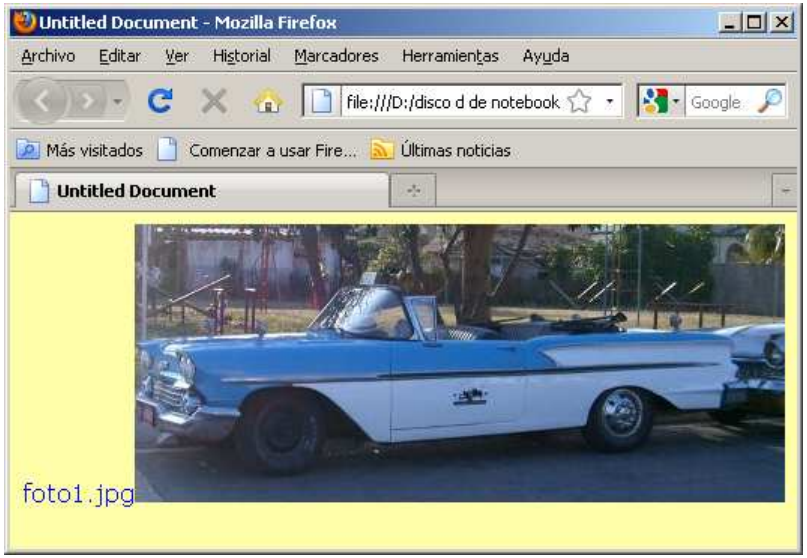


Figura 9: prompt, switch e imagen

Bucles

Se usan para realizar una tarea repetidamente, de acuerdo a una condición determinada. Se utilizan: FOR – WHILE – DO WHILE.

Bucle FOR

Se usa para realizar tareas repetitivas un número determinado de veces:
Sintaxis:

```
for (desde (inicio);hasta (condición);cómo_se_actualiza (incremento)) {
        Tareas a realizar
}
```

Ejemplo:

```
<html>
<body>
<SCRIPT>
<!--

for (i=1;i<6;i++){
        document.write("<H" + i +">Encabezado "+ i+ "</H><br>");
}
-->
</SCRIPT>
</body>
</html>
```

Ejemplo 19

En el ejemplo se dice:

- Para la variable i: **desde** i=1, **hasta** que la variable i sea menor que 6 → en 5 tendrá que salir, esta es la condición: en este punto se dirá hasta cuándo se realizará la iteración, **incrementando** i en 1 en cada iteración (cada vuelta del bucle)→ for (i=1;i<6;i++). El bucle for, se encarga de iterar o sea de realizar la tarea una y otra vez, incrementando la variable cada vez hasta que se cumpla la condición, una vez que cumple la condición sale del bucle y continúa con el programa.
- Escriba en el documento(...) → document.write("<H" + i +">Encabezado "+ i+ "</H>
");

Bucle WHILE

Se usa para repetir una tarea un número indefinido de veces, MIENTRAS se cumpla una condición.

Sintaxis:
while (condición){
    tarea a ejecutar
}

Ejemplo:

```
<HTML><BODY>
<SCRIPT>
<!--
var i=0;
while (i < 5){
    i += 1;
  document.write("<H" + i +">Encabezado "+ i+
"</H><br>");
}
-->
</SCRIPT>
</BODY></HTML>
```

Ejemplo 20

En el ejemplo se dice:

- Mientras i sea menor que 5, entonces:
    - Incremente i en 1
    - Escriba en el documento(..)→document.write("<H" + i +">Encabezado "+ i+ "</H>
")

Bucle while: primero se evalúa la condición y luego realiza la tarea.

Bucle do..while

Este bucle se ejecuta por lo menos una vez

Sintaxis:
do {
        tareas
} while (condición)

```
<html><body>
<SCRIPT>
<!--
var j=prompt("Cantidad de veces que ingresará al
bucle:","");
var i=1;
do {
document.write("<H" + i +">Encabezado "+ i+
"</H><br>");
i += 1;
} while (i <= j)
-->
</SCRIPT>
</body></html>
```

Ejemplo 21

El ejemplo dice:

- Do → Realice determinadas tareas, en este caso escribe el documento y luego incrementa j en uno → document.write("<H" + i +">Encabezado "+ i+ "</H>
"); j += 1;

- While → Mientras no se cumpla la condición, en el momento en que se cumpla la condición debe salir, en este caso da vueltas mientras la variable i sea menor que la variable j.

NOTA: En el caso del bucle **do..while**, la tarea se realiza al menos una vez, mientras que en el bucle **while**, evalúa la expresión y luego ingresa al bucle.

Realice el siguiente ejemplo para comprobar lo que se dice en la NOTA:

```
<script>
var num=0
while (num==1) {
alert("Ingreso al while");
}
do {
alert("Ingreso a do..while");
} while (num==1);
```

Ejemplo 22

Este ejemplo muestra claramente que NUNCA ingresará al bucle while, porque la variable num jamás toma el valor 1, pero sí ingresará al menos una vez al bucle **do.. while.**

IV - Funciones

Qué es una función?

- Una función es pequeño programa (subprograma) que realiza una o varias tareas determinadas.
    - Conj. de instrucciones englobadas en un mismo proceso: este proceso es una función y tiene un nombre.
    - Pueden ser convocados por su nombre desde diferentes lugares. De modo que sirven para evitar la repetición del código

Sintaxis

Las funciones poseen la siguiente estructura:
Function nombre(parámetro_1, parámetro_2, ...,parámetro_n) {
Código de tareas que realizará la función
}
Se escribe la palabra function, luego el nombre de la función y entre paréntesis los parámetros, si no los tiene, igual se incorporan los paréntesis() .
Ejemplo:

```
function alarma(){
Window.alert(“ERROR!!!!”)
}
```

Formas de ejecutar una función:
solo necesita invocar su nombre entre tags de script de la siguiente manera:

1. Luego de la función invocando su nombre: alarma()
2. En otra parte de la página
3. Desde un evento

Realice los siguientes ejemplos:

```
<html> <body>
<script type="text/javascript">
function alarma(){
alert(“ERRORRRR!")
}
alarma()
</script>
</body></html>
```

Ejemplo 23

Otra forma de llamar a la función:

```
<html><body>
<script type="text/javascript">
function alarma(){
alert("ERRORRRR!")
}
</script>

<h1>veamos que pasa</h1>

<script>
alarma()
</script>
</body></html>
```

Ejemplo 24

Llamar a la función desde un evento:

```
<html><body>
<script type="text/javascript">
function alarma(){
alert("ERRORRRR!")
}
</script>

<h1>veamos que pasa</h1>

<form>
<input type=button value="Presione el botón"
onclick=alarma()>
</form>
</body></html>
```

Ejemplo 25

A tener en cuenta cuando se trabaja con funciones:

- Declararlas entre etiquetas <script>....</script>
- Declarar la función antes de ser llamadas
- Lo conveniente es declarar las funciones en el head, o en programa aparte → prog.JavaScript.
- NO llamar a la función y luego declararla

Parámetros de las funciones

Las funciones tienen una entrada y una salida, que se pueden usar para recibir y devolver datos.
Los parámetros se usan para mandar valores a la función, con estos parámetros realizará acciones.
Entonces los parámetros:

- Se usan para enviar valores a la función

Sintaxis:

```
function hola(nombre){
alert("Hola "+nombre)
}
```

Nota: Para incorporar un parámetro en la función, tenemos que poner el nombre de la variable que almacenará el dato que se ingresa, por otro lado la variable tendrá vida durante la ejecución de la función y dejará de existir cuando la función se termine de ejecutar. Los parámetros pueden recibir cualquier tipo de datos, textos, números, boléanos, u objetos.

```
<html>
<head>
<script>
<!--
function alarma(nombre){
alert("Hola "+nombre)
}
-->
</script>
</head>
<body>
<h1>veamos que pasa</h1>
<script>
v1 = prompt("Escriba su nombre","");
alarma(v1)
</script>
</body></html>
```

Ejemplo 26

Múltiples parámetros:

- Se pueden ingresar mas de un parámetro, separados por coma: ‘,’.

```
function alarma(num1,num2){
document.write("producto= "+num1*num2)
}
```

- Llamado:

```
v1 = (prompt("num1",""));
v2 = (prompt("num2",""))
alarma(v1,v2)
```

Ejemplo:

```
<html><head>
<script>
<!--
function alarma(num1,num2){
document.write("producto= "+num1*num2)
}
-->
</script>
</head>
<body>
<h1>veamos que pasa</h1>
<script>
        v1 = (prompt("num1",""));
        v2 = (prompt("num2",""))
        alarma(v1,v2)
</script>
</body></html>
```

Ejemplo 27

Nota: Los parámetros de las funciones se pasan por valor, o sea que aunque se modifiquen los parámetros en una función, la variable original no cambia su valor. Veamos en un ejemplo:

```
Function valor(mivar){
mivar= 10
document.write(“Variable dentro de la funcion: “+ mivar)
}
var mivar= 3
valor(mivar)
document.write(“Variable fuera de la funcion: “+ mivar)
```

Ejemplo 28

Analice el ejemplo.

Retorno de la función:

Los parámetros de una función JAVASCRIPT se pasan por valor, o sea que si dentro de la función se cambia el valor de la variable, este cambio no se ve reflejado a menos que se utilice la palabra reservada return

- La función realiza un conjunto de acciones y devuelve un valor.
- Se usa la palabra return seguida por el valor que se desea retornar.
- Cuando se la convoca se asigna a una variable.

Pruebe los siguientes Ejemplos:

Sin_return.htm

```
<html><body>
<script type="text/javascript">
```

```
function suma(num) {
num++;
alert(num);
}
var a=1;
d=suma(a);
alert(d);
alert(a)
</script>
</body></html>
```

Ejemplo 29

Con_return.htm

```
<html><body>
<script type="text/javascript">
function suma(num) {
num++;
alert(num);
return num;
}
var a=1;
d=suma(a);
alert(d);
</script>
</body></html>
```

Ejemplo 30

Múltiples retornos:

- Dentro de una función se pueden usar mas de un return
- Lógicamente sólo habrá un valor que cumple con las condiciones de la función y ese es el valor que retornará.

Funciones básicas

JAVASCRIPT posee funciones predefinidas algunas de las cuales enumeramos a continuación:

- eval(string): recibe una cadena de caracteres y la evalúa.

```
<html><body>
<script type="text/javascript">
var miTexto = "3 * 5 + 4”
document.write(‘Var miTexto: ‘+miTexto + "= ");
eva= eval(miTexto);
document.write(eva)
</script>
</body></html>
```

Ejemplo 31

- parseInt(cadena,base): Devuelve un valor numérico resultante de convertir la cadena en un número entero en la base indicada.

```
<html><body>
<script type="text/javascript">
<!--
valor = prompt("Ingrese el valor 1","");
valor2 = prompt("Ingrese el valor 2","");
//los valores serán concatenados
document.write("Sin ParseInt (concatena) " + valor + " +
"+ valor2 +" = ");
```

```
sum_str = valor+valor2;
document.write(sum_str + "<br>");

//los valores serán sumados
document.write("Con ParseInt (suma) "+ valor + " + "+
valor2 +" = ");
p_i_valor = parseInt(valor);
p_i_valor2 = parseInt(valor2);
sum_val = p_i_valor+p_i_valor2;
document.write(sum_val + "<br>")
-->
</script>
</body></html>
```

Ejemplo 32

- parseFloat(cadena): Convierte la cadena en un número (no necesariamente entero) y lo devuelve.

```
<html><body>
<script type="text/javascript">
<!--
valor = prompt("Ingrese el valor 1","");
valor2 = prompt("Ingrese el valor 2","");
//los valores serán concatenados
document.write("Sin ParseFloat (concatena) " + valor + "
+ "+ valor2 +" = ");
sum_str = valor+valor2;
document.write(sum_str + "<br>");

//los valores serán sumados
document.write("Con ParseFloat (suma) "+ valor + " + "+
valor2 +" = ");
p_i_valor = parseFloat(valor);
p_i_valor2 = parseFloat(valor2);
sum_val = p_i_valor+p_i_valor2;
document.write(sum_val + "<br>")
-->
</script>
</body></html>
```

Ejemplo 33

- isNaN(número): (is Not a Number = Si no es un número) Devuelve un booleano dependiendo de lo que recibe por parámetro. Si no es un número devuelve un true, si es un numero devuelve false.

```
<html><body>
<script type="text/javascript">
<!--
miInteger = parseInt("A3.6");
pp=isNaN(miInteger);
document.write("No es un número: "+pp)
mifloat = parseInt("3.6");
isNaN(mifloat);
document.write("<br>Es un Número: "+isNaN(mifloat))
-->
</script>
</body></html>
```

Ejemplo 34

V - Objetos en JavaScript

JavaScript utiliza objetos, de hecho hemos venido trabajando con ellos, cuando decimos *document.write()* estamos trabajando con el objeto document. Haremos ahora un pequeño resumen de lo que necesitamos para trabajar con los objetos de JavaScript, sin complicarnos demasiado.

Qué es un objeto?

Aunque hemos venido trabajando con objetos, en este apartado trataremos de explicar en forma simple e intuitiva qué son los Objetos. Básicamente el modelo de objetos se basa en los objetos del mundo real: un perro, una planta, una ventana, etc.
Los objetos de la vida real poseen dos características: propiedades y comportamientos, los **propiedades** del perro podrían ser: nombre, color, casta y el **comportamiento** sería menear la cola, ladrar, etc.

Los objetos de software son modelos tomados de objetos de la vida real y también poseen propiedades y comportamiento.

Un objeto de software:

- Mantiene sus propiedades en una o más variables. Recuerde que una variable es un ítem de datos que posee un nombre o identificador.
- Y su comportamiento es implementado mediante **métodos**, un método es una función o subrutina asociada al objeto. A menos que se diga lo contrario una función no es estática.

En resumen se puede decir que un *objeto* de software es un *conjunto de variables* (llamadas propiedades) y *funciones* (llamadas métodos) relacionadas. Recuerde que los métodos son funciones que manipulan las variables, o sea las propiedades.
Los objetos se dividen en *clases* e *instancias*, una *clase define a los objetos en forma general* por ejemplo la clase perro, pero *su* perro es una *instancia* de la clase perro.

Una clase es una descripción de la forma y funcionamiento de los objetos, con las clases no se realiza ningún trabajo, sino con los objetos que creamos a partir de una clase.

Cómo instanciar un objeto?

Como se dijo en el párrafo anterior una clase define a un objeto en forma general, o sea posee la definición de sus características y su funcionalidad.

Sabemos ya que no se trabaja con clases, son solo definiciones, para trabajar con una clase de objetos, se debe crear un objeto instanciado de esa clase.

La creación de objetos de una determinada clase se realiza mediante el uso de la instrucción new. En el siguiente ejemplo crearemos un objeto de la clase Date, que permite manejar fechas y horas, y lo cargaremos en la variable fecha.
fecha = new Date();

Al no ingresar parámetros, el objeto fecha contendrá fecha y hora actuales. Pruebe el siguiente ejemplo:

```
<html>
<body>
<script type="text/javascript">
        fecha=new Date();
        dia = fecha.getDate();
        año = fecha.getYear();
document.write('fecha: '+ fecha+ '<br>Dia: '+dia +
'<br>Año: '+ año)
</script>
</body>
</html>
```

Ejemplo 35

En este ejemplo, se crea un objeto fecha y se utilizan los métodos getDate y getYear para obtener día y año.

Cómo se accede a las propiedades y métodos de un objeto?

Ya hemos estado accediendo a métodos de objetos, el que más hemos usado es el objeto document y el método write: document.write('texto'). Generalizando entonces, para acceder a un método se escribe: *objeto.metodo(parámetros)*, si no se tienen parámetros para ingresar, igual hay que incorporar los paréntesis.
De la misma forma se accede a las propiedades, pero tenemos que tener en cuenta que las propiedades son variables, por lo tanto no tenemos parámetros, entonces se simplifica de la siguiente manera: *objeto.propiedad*.

Clases predefinidas

JavaScript posee clases predefinidas, en este apartado mostraremos las más usadas.
Las clases se escriben con la primera letra en mayúscula.

Clase Array

Esta clase permite crear vectores que son tipos de datos complejos, es un conjunto ordenado de elementos en donde cada elemento es una variable.
Por ejemplo se puede tener un Array de días de la semana, el Array contendrá siete elementos, del cero al seis (los arrays comienzan siempre en la posición cero), cada uno correspondiente a un día de la semana.

Declaración de un Array
Siguiendo con el ejemplo de los días de la semana, declaramos el Array (la primera letra con mayúscula) semana de la siguiente manera:
Semana = new Array(7)

En este caso decimos que el array poseerá siete (7) elementos aunque se puede dejar abierta la posibilidad, sin asignar la cantidad de elementos.

Asignar valores al Array:

Para asignar valores al array, hay que tener en cuenta que se debe indicar el índice de la posición del dato entre corchetes como se muestra a continuación:

Semana[0] = lunes

Semana[1] = martes
.....
semana[6] = domingo

Captar valores de un Array

Para obtener los valores de las diferentes posiciones del array se procede de igual manera, poniendo entre corchetes el índice de la posición a la que queremos acceder.
Realice el siguiente ejemplo:

```
<html><body>
<script type="text/javascript">
        var semana = new Array(7)
        semana[0] = "lunes"
        semana[1] = "martes"
        semana[2] = "miercoles"
        semana[3] = "jueves"
        semana[4] = "viernes"
        semana[5] = "sabado"
        semana[6] = "domingo"
for (i=0;i<7;i++){
document.write('Indice ' + i + ' de la semana: ' +
semana[i] + '<br>') }
</script>
</body></html>
```

Ejemplo 36

Tipos de datos en un Array

Se pueden introducir cualquier tipo de datos en un Array, los datos del tipo carácter deben ir entre comillas. A pesar de la improlijidad, los arrays pueden contener dentro de sus casillas distintos tipos de datos, o sea en una casilla se puede introducir texto, en otro un valor booleano, en otra un número, etc.
Longitud de un Array
Los arrays además de almacenar el valor de cada posición, almacena el número total de posiciones o sea en el caso del Array semana, la longitud es de 7 posiciones.
Para obtener esta longitud usamos una propiedad *length* del objeto array, recuerde que la propiedad de un objeto es una variable, veamos en un ejemplo:

```
<html><body>
<script type="text/javascript">
        var semana = new Array(7)
semana[0] = "lunes"
semana[1] = "martes"
semana[2] = "miercoles"
semana[3] = "jueves"
semana[4] = "viernes"
semana[5] = "sabado"
semana[6] = "domingo"

for (i=0;i<7;i++){
document.write('Indice ' + i + ' de la semana: ' +
semana[i] + '<br>')
 }

document.write("<h1>Longitud del array: " +
semana.length + " </h1>" )
</script>
```

```
</body></html>
```

Ejemplo 37

Otra forma de inicializar el Array semana sería:

Semana = new Array(“lunes”, “martes”, ”miércoles”, ”jueves”, ”viernes”,“sabado”, “domingo”)

En este caso se reduce la tarea y se crea un array con 7 posiciones y el contenido de cada posición se declara entre paréntesis, separada con comas.
Ejemplo:

```
<html><body>
<script type="text/javascript">
        semana = new Array("lunes", "martes","miércoles","jueves","viernes","sabado",
"domingo")

for (i=0;i<7;i++){
document.write('Indice ' + i + ' de la semana: ' + semana[i] + '<br>')
 }

document.write("<h1>Longitud del array: " + semana.length + " </h1>" )
</script>
</body></html>
```

Ejemplo 38

Propiedades del Array

length: propiedad que contiene la cantidad de elementos del vector

Métodos del Array

- *concat*(*vector2*): método que concatena los elementos del vector2 al final del vector convocante. No funciona con Explorer 3
- *sort*(*compara*): método que ordena alfabéticamente los elementos del vector. Se puede añadir como parámetro una función de comparación (compara), para ordenar de acuerdo a ésta. La función debe aceptar dos parámetros:
    - devuelve cero si son iguales
    - menor que cero si el primer parámetro es menor que el segundo
    - mayor que cero si el segundo parámetro es menor que el primero

Clase Date

Permite manejar fechas y horas.

Declaración de un objeto de la clase Date

Fecha = new Date();
En este caso, como vimos anteriormente si no agregamos parámetros, el objeto Fecha contendrá la fecha y horas actuales, obtenidos según el reloj del sistema.
Si queremos asignar una fecha a una variable se procede utilizando parámetros, se debe tener en cuenta que los meses comienzan en cero.

Ejemplo: cumple = new Date(2000,11,23)

Métodos de Date

El objeto Date utiliza en sus métodos, para asignar valores el prefijo set y para obtener valores el prefijo get.

- *getTime*() - *setTime*(mls): obtiene y asigna, respectivamente, fecha y hora tomados como milisegundos transcurridos desde el 1 de enero de 1970
- *getYear*() – *setYear*(año):obtiene y asigna el año de la fecha, entre 1900 y 1999 en dos dígitos, no es recomendable.
- *getFullYear*() – *setFullYear*(año): obtiene y asigna el año de la fecha, en cuatro dígitos.
- *getMonth*() – *setMonth*(mes)
- *getDate*() – *setDate*(dia)
- *getHours*() – *setHours*(horas)
- *getMinutes*() – *setMinutes*(minutos)
- *getSeconds*() – *setSeconds*(segundos)
- *getDay*(): devuelve el día de la semana en formato número del 0 al 6.

Clase Image

Esta clase representa una imagen. Está disponible desde Netscape 3 y Explorer 4.

Propiedades de Image

- *SRC*: contiene la dirección del archivo con la imagen
- *LOWSRC*: es la imagen que se cargue antes que la imagen final, generalmente se establece una imagen pequeña.
- *WIDTH*: ancho de la imagen
- *HEIGTH*: alto de la imagen
- *BORDER*: ancho del borde que se establece a la imagen
- *ALT*: utilizado para navegadores que no visualizan la imagen
- *HSPACE*: espacio horizontal entre imágenes
- *VSPACE*: espacio vertical entre imágenes

Declaración del objeto imagen:

Activa = new Image() →la primera letra en mayúscula.

Luego se establece el lugar en donde se encuentra la imagen. Si se encuentra en el mismo directorio de la pagina, se procede de la siguiente manera:
Activa.src = "nombrImagen.gif"
Para saber si el navegador entiende los objetos de imagen, se usa la siguiente línea:
if (document.images), si devuelve true (verdadero), entonces realiza la tarea

Ejemplo de Rollover o imágenes de sustitución:

en este ejemplo se cuenta con dos imágenes, una de las cuales se carga junto a la página. Al pasar el mouse por la imagen se cambia por una segunda.

```
<HTML><HEAD>
<SCRIPT LANGUAJE="JavaScript">
<!--
if  (document.images) {
        var activa = new Image();
        activa.src = "left.gif";
```

```
        var noactiva = new Image();
        noactiva.src = "right.gif";
        }
function activar(nombreimg) {
if  (document.images) {
        document[nombreimg].src= activa.src;}
}
function noactivar(nombreimg) {
if  (document.images) {
        document[nombreimg].src= noactiva.src;}
}
-->
</SCRIPT>
</HEAD>
<BODY>
<A HREF = "valida.htm"
onMouseOver="activar('prueba');"
onMouseOut="noactivar('prueba');">
<IMG NAME ="prueba" SRC = "right.gif" BORDER = 0>
</A>
</BODY></HTML>
```

Ejemplo 39

Esto se hace dentro de un vínculo → <A HREF...., la razón es que los eventos onMouseOver y onMouseOut son admitidos por la etiqueta **<IMG.**
No olvidar el atributo **NAME=”nombre_de_imagen”,** esto es para identificarla posteriormente.
Tal vez este código parezca un poco reiterativo, ya que se declara dos veces *document[nombreimg]*, pero recuerde que cada vez que se convoca una imagen, el navegador debe traer el archivo desde un lugar remoto, por tanto es preferible haberlas precargado, en busca de optimizar los tiempos de acceso.

Clase String

Todas las variables de tipo texto son objetos de la clase string, por lo tanto no es necesario definirlas con la orden new como se hacía con los objetos vistos anteriormente (aunque puede hacerlo, pero no funcionará en JavaScript 1.0) y todas las variables de este tipo “tienen derecho” de hacer uso de una serie de propiedades y métodos, por ejemplo: saber la longitud en caracteres o extraer una porción del texto o cambiar el texto a mayúsculas, etc.

Declaración del objeto String:

Cadena = “Esta es un objeto de la clase string”

Propiedades de String

- *length*: Esta clase solo tiene la propiedad length, que guarda el número de caracteres del string.

Métodos de String

Presentamos acá los métodos más usados para manipular los strings, existen otros que sirven para aplicar estilos a los textos, y es como usar lenguaje HTML.

- *charAt(indice)*: devuelve el carácter que hay en la posición indicada por *indice,* las posiciones comienzan en cero.

- *indexOf(carácter,desde)*: Devuelve la posición en que se encuentra el *carácter* , el segundo parámetro es opcional ya que indica desde el lugar en que hará la búsqueda.
- *lastIndexOf(carácter,desde)*: Devuelve la posición en que se encuentra el *carácter* pero realiza la búsqueda desde el final del texto, o sea de derecha a izquierda.
- *replace(texto_cambiar,nuevo_texto)*: Reemplaza el texto_a_cambiar por el nuevo_texto. Implementado en javascript 1.2
- *split(separador)*: Crea un vector a partir de un texto en el que cada elemento está separado por el *separador* indicado por el parámetro. Implementado para jascript 1.1 y posterior.
- *substring(lugar_comienzo,lugar_fin)*: devuelve el substring que se inicia en el carácter de *lugar_comienzo* y termina en el *lugar_fin* (lugar_comienzo y lugar_fin son números).
- *toLowerCase()*: Pasa todos los caracteres a minúscula.
- *toUpperCase()*: Pasa todos los caracteres a mayúscula.
- *toString()*: sirve para convertir cualquier objeto en cadenas de caracteres.

Ejemplo de string:

```
<HTML><HEAD><TITLE>
Cátedra de Laboratorio 1 - Carrera de Programador
Universitario
</TITLE></head>
<body>
<script>
<!--
var hola="Trabajando con string en Laboratorio 1" ;
var texto = "";
//largo del string
largo = hola.length;
document.write("Largo del string: "+largo+ "<br>");
//-------------------
/*En la variable letra cargamos cada caracter del string
hola, en texto vamos sumando cada letra separada por
un guión*/
for (i=0;i<largo;i++) {
 letra = hola.charAt(i);
 texto = texto + letra +"-"
 }
document.write("Texto separado por guiones:"+texto +
"<br>");
//-------------------
//Dónde se encuentra la s?
ese = hola.indexOf("s");
document.write("La letra S se encuentra en el lugar:
"+ese + "<br>");
//-------------------------
//uso de replace
re=hola.replace("string","<b>texto</b>");
document.write("En la variable re, cambio el texto: " + re
+"<br>")
//--------------------
//uso de substring
substr=hola.substring(1,6);
document.write("Subtexto: "+substr+"<br>");
//--------------------
```

```
//Paso todo a mayuscula
mayu = hola.toUpperCase()
document.write("Texto en mayuscula: "+mayu);
-->
</script>
</body></HTML>
```

Ejemplo 40

Clase Math en JavaScript

Proporciona mecanismos para realizar operaciones matemáticas complejas. Trabaja directamente con la clase Math, por lo que no usaremos la instrucción new.

Propiedades de Math

- E: constante de Euler.
- LN2: Logaritmo neperiano de 2
- LN10: Logaritmo neperiano de 10
- LOG2E: Logaritmo en base 2 de E
- LOG10E: Logaritmo en base 10 de E
- PI: número PI → 3,1416...
- SQRT1_2: Raiz cuadrada de ½
- SQRT2: Raiz cuadrada de 2

Métodos de Math

- abs(): valor absoluto de un número sin signo
- acos(): arcocoseno de un número en radianes
- asin(): arcoseno de un número en radianes
- atan(): arcotangente de un número
- ceil(): devuelve el entero igual o el siguiente de un número.
- floor(): devuelve el número o el inmediato anterior.
- cos(): coseno de un número
- exp() resultado de elevar el número E por un número
- log(): Logaritmo neperiano de un número
- max(): devuelve el mayor de dos números dados.
- Min(): devuelve el menor de dos números dados.
- Pow(): trabaja con dos parámetros y devuelve el primer número elevado al segundo.
- random(): Devuelve un número aleatorio entre 0 y 1.
- round(): Redondea al entero más próximo
- sin(): devuelve el seno del número en radianes
- sqrt(): raiz cuadrada de un número
- tan(): Tangente de un número en radianes

Ejemplo de la clase math:

```
<HTML><HEAD><TITLE>
Cátedra de Laboratorio 1 - Carrera de Programador
Universitario
</TITLE></head>
<body>
<script>
<!--
neper = Math.E;
_pi = Math.PI;
document.write("Euler: "+neper+"<br>"+"PI:
"+_pi+"<br>");
document.write("Usando pow elevamos 2 al cubo:
"+Math.pow(2,3)+"<br>")
document.write("Redondea 2.6: "+
Math.round(2.6)+"<br>");
document.write("Redondea 2.4: "+ Math.round(2.4));
```

```
-->
</script>
</body></HTML>
```

Ejemplo 41

VI - Jerarquía de Objetos del navegador

Al cargar una página, el navegador crea una jerarquía de objetos en memoria que permite controlar los elementos de dicha página. Es fundamental conocer esta jerarquía para poder controlar la página con Javscript.
Encabeza esta jerarquía el objeto window, que ofrece una serie de propiedades y métodos para controlar el aspecto de la ventana, la barra de estado, abrir ventanas secundarias, etc. Por otro lado da acceso a otros objetos como el objeto document que, como ya hemos visto, trabaja sobre la página web que se está visualizando, el historial o los distintos frames de la ventana.
JavaScript sabe que todos los objetos comienzan en window, aunque no lo especifiquemos, podríamos escribir: *window.document.write("hola")* y estaría muy bien.
En la figura se muestra un esquema elemental de la jerarquía de objetos:

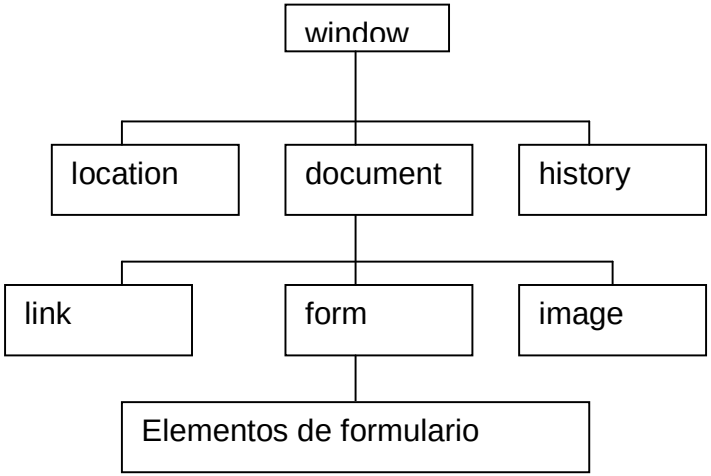


Figura 10: Jerarquía de objetos

Propiedades de los objetos

Las propiedades de los objetos pueden ser otros objetos, el ejempllo más claro es el objeto document, como describimos anteriormente.

Objeto Window

El principal objeto de JavaScript es el objeto window, de hecho ya hemos trabajado con él, y es el único que puede ejecutar tareas sin ser llamado, por ejemplo decimos *alert('algo')* en lugar de decir *window.ALER('algo').*
Este objeto define ventanas e incluye los objetos referentes a la barra de tareas, al documento, etc.

Propiedades del objeto window

Solo nombraremos las propiedades más usadas:

- *closed*: ventana cerrada
- *defaultStatus*: Texto por defecto en la barra de estado
- *document*: contiene la página web sobre la que se está trabajando
- *frame*: frame de una página wweb se accede por su nombre.

- *history*: objeto historial de las páginas visitadas.
- *locationbar*: objeto barra de direcciones de la ventana.
- *menubar*: objeto barra de menús de la ventana.
- *name*: Nombre de la ventana
- *self*: ventana o frame actual
- *status*: texto de la barra de estado
- *toolbar*: objeto barra de herramientas.

Muchas de estas propiedades no funcionan en algunos navegadores

Métodos de window

- *[variable] = window.open*(URL, nombre, propiedades) Esta frase permite crear y abrir una nueva ventana. En este caso, que es opcional, se asigna la ventana a una variable para poder acceder a ella.

Qué significa lo que está entre paréntesis?

- Recuerde que la dirección URL es la dirección de la ventana que estamos abriendo.
- El nombre es el que queremos que se utilice como parámetro de un TARGET.
- Propiedades
    - *Toolvar[= yes/no]*
    - *Location[=yes/no]*
    - *directories[=yes/no]*
    - *status[=yes/no]*
    - *menubar[=yes/no]*
    - *scrollbars[=yes/no]*
    - *resizable[=yes/no]*
    - *width[=yes/no]*
    - *heigth[=yes/no]*

- *Close(mensaje):* cierra la ventana.
- *Alert(mensaje)*: Muestra una ventana de diálogo, como se explicó anteriormente
- *Confirm(mensaje)*: ventana de diálogo con el mensaje y con dos posibles opciones.
- *Prompt(mensaje,sugerencia)*: anteriormente usado. Muestra el mensaje, siendo sugerencia el valor inicial. La opción OK el método devuelve la cadena ingresada, si oprime Cancelar, devuelve el valor null (false)
- *SetTimeout('función',tiempo)*: Llama a función cuando haya pasado el tiempo establecido en milisegundos.
- *back():* ir una página atrás en el historial
- *blur():* Quitar el foco de la ventana actual
- *focus():* coloca el foco de la aplicación en la ventana.
- *forward():* Ir una página adelante
- *home():* ir a la página de inicio del explorador.
- *MoveBy(pixelsX,pixelsY):* mueve la ventana de acuerdo a los pixeles que se indican por parámetro.
- *print(): muestra la ventana de impresión del navegador.*
- *resizeBy(ancho,alto):* redimensiona la ventana del navegador para que ocupe el espacio indicado por parámetros

Barra de Estado

Sirve para controlar la barra de estado. Si deseamos que la página tenga un mensaje en la barra de estado, podemos hacerlo de dos formas diferentes:

- Con precarga en la declaración de body:

```
<BODY onLoad="window.defaulStatus='Esta es mi barra de estado';return true">
```

- O dentro de un script como se muestra a continuación:

```
<script>
        window.status='Barra de estado!!!!'
</script>
```

Ejemplo 42: Barra de estado

- Por medio de un botón:

```
<HTML>
<BODY>
<form>
<input type="button" value="Mirame"
onclick="window.status='barra de estado'">
</form>
  </BODY>
</HTML>
```

Ejemplo 43

Objeto document

Representa al documento HTML en que estamos trabajando.

Propiedades de document

Se muestran a continuación las propiedades más usadas:

- *alinkColor*: color de los enlaces activos
- *bgColor*: color de fondo del documento
- *Anchor*: ancla o marcador de una página, trabaja como en HTML.
- *fgColor:* color del texto
- *lastModified:* fecha de la última modificación
- *linkColor:* Color de los enlaces
- *vlinkColor: color de los enlaces visitados.*

Métodos de document

Solo nos ocuparemos de los métodos más usados.

- *close()*: cierra el flujo del documento
- *open()*: abre el flujo del documento.
- *write(): Escribe dentro de la página web*
- *writeln():* escribe en el documento e inserta un salto de línea al final.

Pruebe los siguientes ejemplos:
1) Ultima modificación: lastModifiied

```
<html><HEAD>
<script type=”text/javascript”>
function ultima(){
document.write('Ultima modificación:
'+document.lastModified)
}
</SCRIPT>
</HEAD>
<body>
<script>
ultima()
</script>
```

```
</body></html>
```

Ejemplo 44

VII Ejemplos con Imágenes

Una forma fácil de trabajar con imágenes es embebiendo código HTML con JavaScript de la siguiente forma:

```
<html>
<body>
<script type="text/javascript">
cual=parseInt(prompt("ingrese un número:",""));
p=isNaN(cual);
/*en este caso debe ingresar un número entre 1 y 4, en
caso de que no ingresó un número o está fuera de
rango,  devuelve error*/
if (p) {
        alert("error no ingresó numero")
        }else{
if (cual<1||cual>4){
        alert("fuera de rango")
        }else{
        document.write('<img
src="fotos/img'+cual+'.jpg">')
}
}
</script>
</body>
```

Ejemplo 45

En el ejemplo, se supone que se dispone de cuatro imágenes, la salida sería:

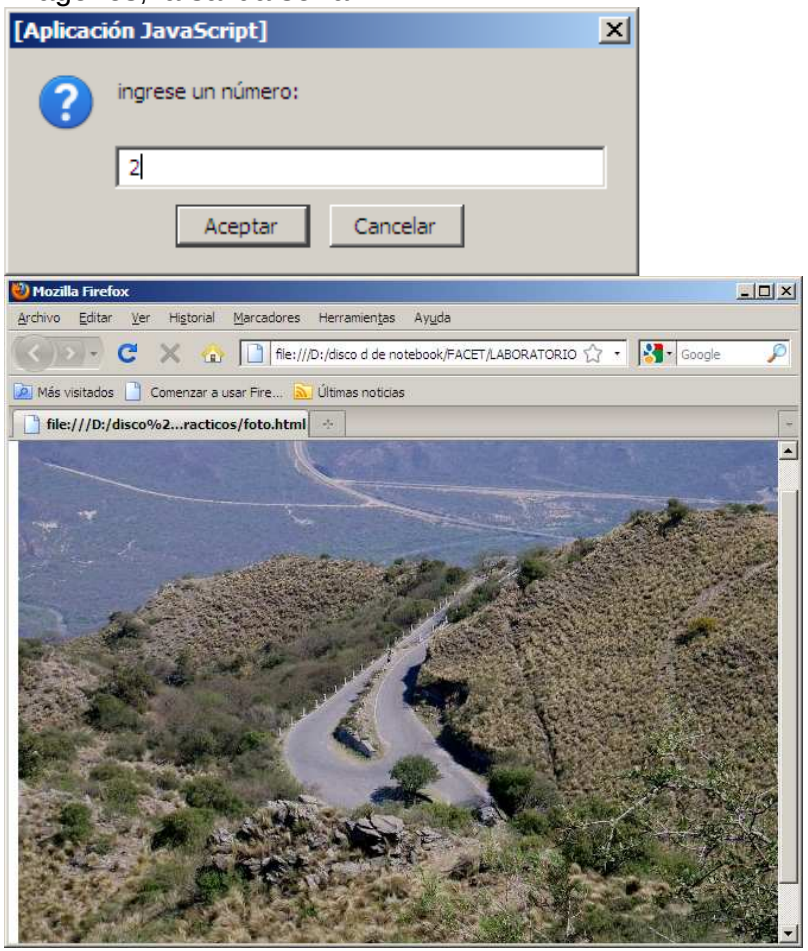


Figura 11: HTML + JavaScript

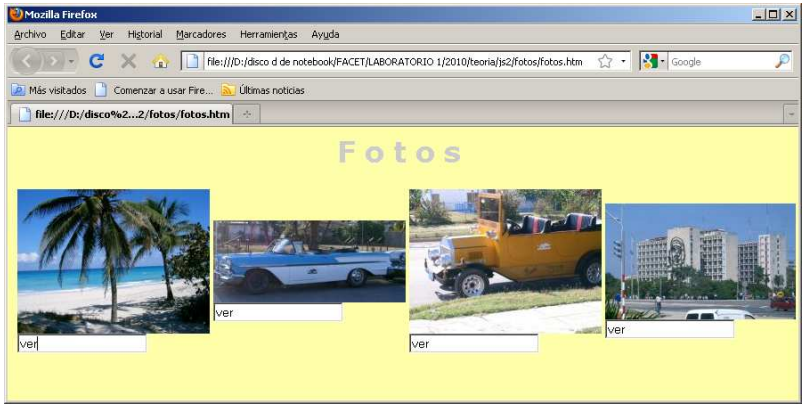
Con botones y window.open

En este ejemplo se muestra un catálogo de fotos que permiten motrar otra, al presionar el botón “ver”.

```
<html><head>
<link rel="StyleSheet" href="estilo.css" type="text/css">
</head>
<body>
<h1><b>Fotos</b></h1>
<table>
<tr><td>
<img src="fondo_chico.jpg"><br>
<input type= “button” value=ver
onclick="window.open('fondo.jpg','ventana1','width=500,h
eight=300')">
</td><td>
<img src="auto_chico.jpg"><br>
<input type= “button” value=ver
onclick="window.open('auto1.1.jpg','ventana2')"><br>
</td><td>
<img src="auto2_chico.jpg"><br>
<input type= “button” value=ver
onclick="window.open('auto2.jpg','ventana3',')">
</td><td>
<img src="che_chico.jpg"><br>
<input type= “button” value=ver
onclick="window.open('che.jpg','ventana4','width=500,hei
ght=400, top=0, left=0, toolbar=no,location=no,
status=no,menubar=no,scrollbars=no, resizable=no')">
</td><tr></table>
</body>
</html>
```

Ejemplo 46

Vista en el Browser:



Al presionar el botón ver:

Figura 12: Foto activada luego de presionar botón ver

Convocar imágenes mediante document
Una forma fácil de convocar imágenes es mediante el uso de document, de la siguiente manera:

```
document.nombre_imagen.src=”imagen.jpg”
```

Siendo nombre_imagen, coincidente con el nombre de la imagen convocada en el documento html:

<img src="img1.jpg" name=" nombre_imagen ">

Ejemplos con funciones – Carrusel de Fotos
En este ejemplo se trabaja con las siguientes variables:
Foto: definirá la foto que se verá
Total: es el número total de fotos
Mas: definirá si el carrusel va hacia adelante o hacia atrás.

```
<script type="text/javascript">
foto = 1;
function cambia(mas, total){
        foto = foto + mas;
if (foto > total){
                foto = 1}
if (foto < 1){
        foto = total;}
document.kkk.src = "img"+foto+".jpg"
}
</script>
```

Ejemplo 47: función para navegar entre las fotos

```
<body text=white>
 <center><h1>Fotos</h1></center>
<form>
<center>
 <img src="img1.jpg" width="600" heigth="75"
name="kkk"><br>
 <input type=button name="_foto" value="<<--"
onClick="cambia(-1,6)"> <input type=button
name="_foto" value="-->>" onClick="cambia(1,6)">
</center>
 </form>
</body>
 </html>
```

Ejemplo 48: Carrusel de fotos

Vista en el Browser

Figura 13: vista en el browser del carrusel

Carrusel de fotos con función en archivo externo

```
<html>
<head>
<script src="_EXT.JS">
</script>

</head>
<body text=white>
<center><h1>Fotos</h1></center>
<form>
<center>
<img src="img1.jpg" width="600" heigth="75"
name="kkk"><br>
<input type=button name="_foto" value="<<--"
onClick="cambia(-1,6)">
<input type=button name="_foto" value="-->>"
onClick="cambia(1,6)"></center>
</form>
</body>
</html>
```

Ejemplo 49: HTML que llama a EXT.JS

```
foto = 1;

function cambia(mas, total){
        foto = foto + mas;
if (foto > total){
                foto = 1}
if (foto < 1){
        foto = total;}
document.kkk.src = "img"+foto+".jpg"
}
```

Ejemplo 50: EXT.JS

Indice

Tabla de Ejemplos

Tabla de Figuras