JavaScript

ASPECTOS BÁSICOS

¿Qué es un lenguaje Script?

- Extienden las capacidades de la aplicación con la que trabajan
- Raramente se usan para algoritmos complejos
- Pueden ejecutarse en el cliente o en el servidor
- En el lado del cliente…
  - **JavaScript** fue desarrollado por Netscape
  - Poco después Microsoft saco su propio lenguaje de script, **VBScript**, para hacer frente a NetScape.
  - Posteriormente IE pasaría a soportar JavaScript, pero no el de Netscape, sino una interpretación del mismo: **JScript**
  - **ECMAScript**: versión estándar

JavaScript y los Navegadores

- Cada navegador acepta una versión de JavaScript ligeramente diferente (o no lo acepta en absoluto)

¿SOLUCIÓN?

- No existe…
  - Podemos probarlo en un número razonable de navegadores
    - Dreamweaver permite comprobar compatibilidad con los navegadores que le indiquemos
  - Podemos incluir código para detectar el navegador, así como diferentes versiones de algunas funciones, una para cada navegador
    - Uso de librerías estándar

¿Qué se puede hacer con JavaScript?

- Validar formularios
- Personalización de las páginas Web
- Hacer interactiva una página Web
- Manipular los elementos HTML de una página
- Inclusión de datos del sistema (hora…). Crear relojes animados…
  - Ej. Reloj que sigue al ratón
- Juegos
  - Ej. Tres en raya
- Chorradas
  - Ej. Matilda
- …

Funcionamiento

- Cuando un navegador solicita una página, el servidor manda tanto el código HTML como los scripts incluidos en dicho código
  - Al igual que hace con las hojas de estilo
- El navegador lee la página de arriba abajo, mostrando el código HTML y ejecutando los scripts en el orden en que aparecen

La etiqueta <script> </script>

- Sintaxis

```
<script type="text/javascript">
//instrucciones javaScript
</script>
```

- ¿Dónde se colocan los scripts?
  - en general, dentro de <head> ... </head>
  - si genera una salida, dentro de <body> ... </body>
    - si hace referencia a un elemento HTML, después de dicho elemento
  - en algunos casos, en los atributos de algunas etiquetas (eventos)
  - en un fichero externo con extensión **.js**

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

Navegadores que no aceptan JavaScript

- Resulta aconsejable escribir los scripts así:

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

- También podemos incluir la etiqueta no script para dar opción a visitar una página alternativa que no use javascript :

```
<noscript>
Su página no acepta Javascript. Pruebe con
<a href="no_script.html">esta página</a>
</noscript>
```

Mi primer script

- ¡Hola Mundo!

```
<html>
<head>
  <script type="text/javascript">
  <!--
  alert("Hola mundo!")
  //-->
  </script>
</head>
<body>
</body>
</html>
```

Mostrar mensajes

- Popups
  - Tipo alert
    - alert("sometext")
  - Tipo confirm
    - confirm("sometext")
  - Tipo prompt
    - prompt("sometext","defaultvalue")
- Escribir en la página HTML
  - document.write

Ejemplo con document.write

```
<html>
<head><title>Escribir datos en la página</title></head>
<body>
  Texto HTML normal.<br><br>

  <script type="text/javascript">
  <!--
  document.write(“<h3>Texto generado con JavaScript:</h3> <br>")
  document.write("La página se modificó por última vez con fecha
  document.lastModified + "<br><br>")
  //-->
  </script>

  Este vuelve a ser texto HTML normal.
</body>
</html>
```

Visualización de errores

- En IE:
  - Herramientas/Opciones de Internet/Opciones avanzadas/seleccionar “Mostrar una notificación sobre cada error de script”
- En Firefox:
  - Tareas/Herramientas/Consola de JavaScript
- Depuración mediante mensajes

Comentarios

- Facilitan:
  - Mantenimiento
  - Reutilización
  - Compartición de código
- Una línea:

```
// comentario de una línea
```

- Varias líneas:

```
/*
comentario de varias líneas
*/
```

Tipos de datos

- Tipos disponibles elementales:
  - Numérico (42, 3.14159, etc)
  - Booleano (true y false)
  - String (“Hola mundo”)
  - null, que denota el valor nulo
  - undefined/NaN, que denota un valor indefinido
- Tipos compuestos:
  - Funciones
  - Object
- Tipado “dinámico”:
  - no es necesario declarar el tipo de las variables
  - se pueden convertir automáticamente de un tipo a otro durante la ejecución

Variables

- Los nombres son ***case sensitive***
- Deben comenzar por una letra o guión bajo. No deben coincidir con palabras reservadas
  - Se podrían definir como variables:
    - Nombre
    - _Opción15
    - Mes3
  - Estarían mal definidas las siguientes variables:
    - 7opcion
    - &inicio
    - ¿nombre

- Declaración:
  - Asignándole un valor:
    - x = 42
  - Con la palabra reservada “var”:
    - var x
  - O bien ambos:
    - var x = 42

Números

OPERADORES UNARIOS	
-	negativo
+ +	incremento
- -	decremento

OPERADORES BINARIOS	
+	suma
-	resta
*	multiplicación
/	división
%	Módulo o resto de una división entera

OPERADORES DE ASIGNACIÓN			
operador	**expresión**	**ejemplo**	**equivale a**
=	a = b	a = 3	
+=	a += b	a += 3	a = a+3
-=	a -= b	a -= 3	a = a - 3
*=	a *=b	a *= 3	a = a * 3
/=	a /=b	a /= 3	a = a / 3

Ejemplo

- Código que obtiene una cantidad y le suma un impuesto del 7%

```
<script type="text/javascript">
<!—
venta = prompt("introduce el importe: ")
impuestos = venta * 0.07
total = venta + impuestos

alert("El total es: " + total)
//-->
</script>
```

- Alternativa:

```
alert("El total es: " + (venta + impuestos))
```

Cadenas de texto

- Se introduce delimitada por comillas simples o dobles

```
var nombre = "Juan";
var apellidos = 'García Fernández';
```

- Con las cadenas de texto podemos realizar operaciones
  - **Concatenación de cadenas** o unión de cadenas:

```
var nombre_completo = nombre + " " + apellidos;
```

  - La propiedad ***length*** indica el número de caracteres que contiene una cadena

```
alert(nombre.length); /* Muestra 5,número de caracteres
                         que contiene la variable nombre*/
```

  - Podemos averiguar la posición de una subcadena dentro de una cadena mediante el método ***indexOf( )***. La primera posición equivale al índice cero
  - Con el método ***substring(inicio,fin)*** extraemos una subcadena a partir de otra cadena mayor
  - Otros métodos para tratamiento de cadenas como ***toUpperCase()*** y ***toLowerCase()***, que convierten una cadena a letras mayúsculas y minúsculas respectivamente

Conversión de tipos cadena y número

- A veces hay información alfanumérica en un formato que no es el adecuado para realizar una operación
  - El valor “113” puede ser el número representado como ciento trece, o bien, una cadena de texto compuesta por dos unos y un tres
    - Si “113” es un número, la expresión 113 + 4 retornará el valor 117 como número.
    - Si “113” es una cadena de texto, la expresión “113” + 4 retornará el valor “1134” como cadena.
- Convertir de cadena a número
  - Para usar una variable cadena como un número entero **parseInt( )**
  - **parseFloat()** convierte una cadena de texto en número flotante
- Convertir un número en cadena
  - Método **toString()**

Booleanos

- Para tratar expresiones lógicas utilizaremos operadores relacionales y lógicos
  - El resultado de una expresión que utilice estos operadores será un valor booleano (true o false)

OPERADORES RELACIONALES		
operador	**expresión ejemplo**	**equivale a**
= =	A = = B	¿Es A igual a B?
! =	A != B	¿Es A distinto de B?
>	A > B	¿Es A mayor que B?
<	A < B	¿Es A menor que B?
>=	A >= B	¿Es A mayor o igual que B?
<=	A <= B	¿Es A menor o igual que B?

  - Los operadores lógicos pueden conectar entre sí varias operaciones relacionales:

OPERADORES LÓGICOS		
operador	**expresión ejemplo**	**equivale a**
!	! (A > = B)	A < B
&&	(A > B) && (B > C)	¿Es A mayor que B y B es mayor que C?
||	(A > B) || (B > C)	¿Es A mayor que B o B es mayor que C?

Estructuras de control

- El código no siempre es secuencial
- A veces se bifurca en función del valor de una condición: SENTENCIAS CONDICIONALES
  - if else
  - switch case
- A veces ciertas instrucciones necesitan ser ejecutadas mientras se cumpla una condición: BUCLES
  - do while
  - while
  - for

Sentencias condicionales

```
 if (condicion){
//código si se cumple la condición
}
else{
//código si no se cumple
}
```

```
switch (expresion){
  case etiqueta1:
       //código si expresion = etiqueta1
       break
  case etiqueta2:
       //código si la expresion = etiqueta2
       break
  default:
       //código si la expresión no es
          ninguna de las anteriores
}
```

Sentencias condicionales: Ejemplos

```
//si la hora es menor que las 10 mostrar buenos dias, sino buenas tardes
var d = new Date()
var time = d.getHours()
if (time < 10) {
   document.write("Buenos dias!")
}
else{
   document.write("Buenas tardes!")
}
```

```
//Recibir diferentes saludos segun el dia de la semana
var d=new Date()
theDay=d.getDay()
switch (theDay){
case 5:
  document.write("Ya es viernes")
  break
case 6:
case 0:
  document.write("Es fin de semana!")
  break
default:
  document.write("¿Cuando llegará el fin de semana?")
}
```

Bucles

- While

```
while (condicion){
        //código
        }
```

- Do ... while

```
do {
        //código
        }
while(condicion)
```

- For

```
for (inicializacion; condicion; incremento){
        //código
        }
```

```
For in  for (variable in object) {
        code to be executed
        }
```

Bucles: Ejemplos

```
var numero = 1;
document.write(“Voy a contar hasta diez <br/>”);
        do{
                document.write(numero+ “<br/>”);
                ++ numero;
        } while (numero <= 10);
```

```
var numero;
document.write(“Voy a contar hasta diez <br/>”);
for (numero=1;numero<=10; numero ++){
        document.write(numero+ “<br/>”);
        }
```

Funciones

- Las líneas de código JavaScript se agrupan en unidades que denominamos ***funciones***, que se ejecutan al ser invocadas
  - Reducen la redundancia del código
  - Favorecen la reutilización y el mantenimiento del mismo
- Declaración:

```
function mifuncion(argument1,argument2,etc)
{
//Codigo
}
```

- Invocación
  - Cuando invocamos una función podemos traspasarle una **lista de parámetros** con los valores que debe calcular

```
mifuncion (valor1, valor2,etc)
```

- Las funciones también pueden devolver valores. Esto se lleva a cabo mediante la sentencia **return**.

Tiempo de vida de las variables

- Las variables pueden ser
  - Locales: dentro de una función
    - Su valor perdura dentro de ellas
  - Globales: fuera de las funciones
    - Disponible desde cualquier parte de código JavaScript que haya en la página
- El uso de var para declarar una variable global es opcional…
  - Aunque si se introduce una variable en una función y no se declara, tendrá ámbito global!
- Para las variables locales, el uso de var es muy recomendable para evitar problemas…

Objetos

- JavaScript es un lenguaje **basado en objetos**
  - Un objeto puede contener propiedades y métodos
    - Las propiedades son los atributos del objeto
      - Un ejemplo de objeto es *document*
    - Los métodos son las acciones que el objeto puede realizar
      - *Un ejemplo de método del objeto document es write()*
- Los objetos pueden ser:
  - Predefinidos como String, Date o Array
  - Cliente proporcionados por el DOM
  - Definidos por el usuario: El programador también puede definir sus propios objetos

El objeto Date

- A partir de una fecha (objeto **Date**) podemos obtener sus componentes hora, minuto y segundo mediante una colección de métodos.
- Para declarar una variable con la fecha y hora actual se crea un objeto de tipo Date mediante el operador **new**

```
var fecha = new Date();
fecha.getHours(); //obtener las horas
fecha.getMinutes(); //obtener los minutos
fecha.getSeconds(); //obtener los segundos
fecha.getYear();  fecha.getFullYear();//obtener el año
fecha.getMonth()+1; //obtener el mes. Se suma 1 porque cuenta desde cero
fecha.getDay(); //obtener el día de la semana: 0=domingo, 1=lunes, …
fecha.getDate(); //obtener el día del mes
fecha.getTime(); //obtener la hora
```

- Para crear un objeto de tipo fecha con datos personalizados:

```
miFecha = new Date(año,mes,dia,hora,minutos,segundos);
miFecha = new Date(año,mes,dia);
```

El objeto Array

- Un **array** es una estructura que almacena una matriz de datos. En JavaScript existe el objeto predefinido **Array** para realizar esa labor
- Para declarar un array se crea un objeto mediante el operador **new:**

```
colores = new Array(5); //Array de cinco elementos sin inicializar
colores = new Array('rojo','verde','amarillo','azul','rosa');
//Declarar e inicializar un Array de cinco elementos
```

- También se puede crear asignándole directamente los valores entre corchetes:

```
colores = ['rojo','verde','amarillo','azul','rosa'];
```

- Para hacer referencia a un elemento del array se utiliza la forma ***NombreArray[índice]***
  - colores[0] hace referencia al primer elemento del array
- Propiedades:
  - ***length*** que informa del tamaño del array
  - **sort()**: ordena elementos
  - **concat():** concatena dos arrays
  - **toString():** devuelve una cadena que representa al objeto

Recorrer un Array

```
var x;
var coches = new Array();
coches[0] = "Saab“;
coches[1] = "Volvo“
coches[2] = "BMW"

for (x in coches){
    document.write(coches[x] + "<br/>")
}
```

Referencias

- W3schools. http://www.w3schools.com
- "Standard ECMA-262: ECMAScript Language Specification". http://www.ecma-international.org/publications/standards/Ecma-262.htm
  - Explorer: http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/reference/dhtml_reference_entry.asp
  - Mozilla: http://developer.mozilla.org/en/docs/Gecko_DOM_Reference
  - Tabla de compatibilidad: http://www.quirksmode.org/