Elementos del lenguaje script

Asignaciones y operadores: expresiones.

1) Asignaciones: expresiones.

El signo igual (=) se utiliza en JavaScript para indicar la acción de asignar un valor. Es decir, una sentencia de JavaScript podría ser por ejemplo:

```
Edad=3;
```

Los operadores que se utilizan en JavaScript son los mismos que se utilizan en cualquier otro lenguaje de programación.

Una expresión en JavaScript es una cosa que se pueda leer como una expresión booleana o una expresión numérica. Las expresiones contienen caracteres como “+” en lugar de expresarlo como “sumado a”. Cualquier combinación valida de valores, variables, operadores y expresiones constituyen una expresión.

Ejemplos de expresiones:

```
var Expresion1 = 3 * (4 / 5);
var Expresion2 = “Hola” + ” que tal”;
var Expresion3 = 4+6;
var Expresion4 = Expresion1 + Expresion3;
```

2) Operadores

Combinando variables y valores, se pueden formular expresiones más complejas. Las expresiones son una parte fundamental de los programas. Para formular expresiones se utilizan los operadores.

Operadores aritméticos

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
/	División
%	Resta de la división

Operadores de asignaciones

Operador	Descripción
= X=5;	Asigna a la variable de la parte izquierda el valor de la derecha. La variable obtiene el valor 5.
+= X+=2;	Suma los operandos de la izquierda y la derecha y asigna el resultado al operando de la izquierda. La variable obtiene el valor de 7.
-= X-=3;	Resta los operandos de izquierda y derecha y asigna el resultado al operando de la izquierda. La variable obtiene el valor de 4.
= X=4;	Multiplica los operandos de izquierda y derecha y asigna el resultado al operando de la izquierda. La variable obtiene el valor de 16
/= X/=2;	Divide los operandos de izquierda y derecha y asigna el resultado al operando de la izquierda. La variable obtiene el valor de 8.
%= X%=3;	Divide los dos operandos y asigna el resto de la división al operando de la izquierda. El resultado es 2 (la resta de 8/3 es 2).

Operadores de comparación

Operador	Descripción
==	Devuelve el valor TRUE cuando los dos operandos son iguales.
!=	Devuelve el valor FALSE cuando los dos operandos son diferentes.
>	Devuelve el valor TRUE cuando el valor de la izquierda es mayor que el de la derecha.
<	Devuelve el valor TRUE cuando el valor de la izquierda es menor que el de la derecha.
>=	Devuelve el valor TRUE cuando el valor de la izquierda es mayor o igual que el de la derecha.
<=	Devuelve el valor TRUE cuando el valor de la izquierda es menor o igual que el de la derecha.

Operadores lógicos

Operador	Descripción
&&	I lógica. Devuelve el valor TRUE cuando los dos operandos son TRUE.
||	O lógica. Devuelve el valor TRUE cuando uno de los dos operandos es TRUE.
!	Negación lógica. Devuelve el valor TRUE cuando el valor es falso.

Variables. Tipos y ámbitos. Conversiones.

Las variables son elementos del lenguaje que permiten almacenar diferentes valores en cada momento. Se puede almacenar un valor en una variable y consultar este valor posteriormente, también podemos modificar su contenido siempre que queramos.

Por ejemplo, podemos solicitar el nombre del usuario al entrar en una web y almacenarlo en una variable para posteriormente, cuando el usuario finalice la sesión despedirlo con una frase que incluya su nombre.

1) Nombre de las variables

Cada variable se identifica por un nombre. Al asignar un nombre a una variable hay que tener en cuenta estas reglas:

- Se pueden utilizar todas las letras del abecedario en mayúsculas y minúsculas, los números del 0 al 9 y el guion subrayado "_".
- Los nombres de las variables no pueden contener puntos ni espacios en blanco.
- El primer carácter deber ser una letra o el guion subrayado.
- En el nombre se distinguen mayúsculas y minúsculas.
- No se pueden utilizar como nombre ninguna de las palabras reservadas por JavaScript, es decir, no se pueden utilizar palabras que coincidan con las que pertenecen al lenguaje JavaScript.
- Oficialmente no hay restricciones en lo que respeta a la longitud del nombre, pero esta ha de caber en una sola línea.

Nombres correctos	Nombres incorrectos
Aux_Nombre	Aux Nombre
resultado	primer.resultado
mail	2mail
primer_apellido	document

2) Declaración de variables

Para declarar variables en JavaScript se utiliza la instrucción var. A cada variable se le asigna un nombre y opcionalmente un valor inicial. Si no se trata de una función, la instrucción var es opcional, pero se recomienda utilizarla.

```
var ejemplo;
Hola = "Hola"; //es lo mismo que var Hola = "Hola";
var resultat=3+7; //la variable tendrá el valor de 10
```

3) Tipos de variables

Los tipos de valores que puede contener una variable JavaScript son:

- **Números**: Pueden contener cualquier tipo de número, real o entero.
- **Operadores lógicos o booleanos**: Puede contener uno de los siguientes valores: TRUE, FALSE, 1 o 0;

- **Cadenas de caracteres o String**: Cualquier combinación de caracteres (letras, números, signos especiales y espacios). Las cadenas se delimitan mediante comillas dobles o simples. Las comillas simples, por norma, serán utilizadas dentro de fragmentos de código delimitados por comillas dobles o viceversa.

Ejemplos:

Tipos de variable	Variable
Numérica	**numero1=3;** **numero2 = 3.7;** **numero3 = 0.6;** **numero4 = 5+7;**
String	**cadena1 = "Pepe";** **cadena2 = "Bienvenido al site web";**
Booleana	**entrada = true;** **salida = false;** **auxentrada = 1;** **auxsalida = 0;**

En JavaScript existen también las matrices o vectores (**Arrays**), que son grupos de elementos que pueden ser de diferentes tipos. Se pueden declarar de diversas formas:

Usando el *constructor* **Array()**:

var coches = new Array();

coches[0] = "Volvo";

coches[1] = "Seat";

coches[2] = "Renault";

De forma condensada: var coches = new Array("Volvo", "Seat", "Renault");

De forma *literal*: var coches = ["Volvo", "Seat", "Renault"];

Las matrices pueden ser de diferentes *dimensiones*, donde los elementos se enumeran de 0 a *n-1*, donde *n* es el número total de elementos de la matriz:

- Una dimensión:

var coches = new Array("Volvo", "Seat", "Renault");

- Dos dimensiones:

```
var personas = new Array(new Array("John", "Doe"), new Array("Will", "Smith"), new Array("John", "Smith"))

// o también de la forma:

var personas= [["John", "Doe"], ["Will", "Smith"], ["John", "Smith"]]

// Obtener el dato de la fila 1 columna 0

var persona = personas[1][0]; // asigna el valor “Will”
```

4) Palabras reservadas en JavaScript

JavaScript tiene una serie de palabras reservadas que no pueden ser utilizadas como nombre de variables:

Break	False	In	This	Void
Continue	For	New	True	While
Declare	Function	Null	TypeOf	With
Else	If	Return	Var	

5) Caracteres especiales en JavaScript

JavaScript tiene también la opción de introducir caracteres especiales (acentos, <,>,”…). Para hacerlo se debe utilizar una codificación especial, en el caso de querer introducir caracteres especiales dentro del código HTML se ha de hacer:

\u00e1 -> á
\u00e9 -> é
\u00ed -> í
\u00f3 -> ó
\u00fa -> ú
\u00c1 -> Á
\u00c9 -> É
\u00cd -> Í
\u00d3 -> Ó
\u00da -> Ú
\u00f1 -> ñ
\u00d1 -> Ñ

6) Integración del código JavaScript

Como ya se ha comentado en apartados anteriores, para incluir un código JavaScript se utilizan las etiquetas <script></script> en el lugar del documento HTML que sea necesario.

Ejemplo:

```
<HTML>
        <HEAD>
        </HEAD>
        <BODY>
                <SCRIPT LANGUAGE=”Javascript”>
                        var ciudad=”Oviedo”;
                        var edad=6;carnet=true;
                </SCRIPT>
        </BODY>
<HTML>
```

Incluir el código JavaScript en medio del código HTML puede dar varios problemas y es recomendable hacerlo solo si no hay otro remedio. La otra opción de incluir el código es creando ficheros con la extensión “.js” donde se incluirá el código JavaScript y después se hará referencia a estos ficheros en el documento principal HTML; estas referencias se han de incluir en la cabecera del documento principal <head>.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
        <title>Control asistencia</title>
        <script src="../jscalendar/calendar.js" type="text/javascript" xml:space="preserve">
        </script>
        <script src="../jscalendar/lang/calendar-ca.js" type="text/javascript"
        xml:space="preserve">
        </script>
        <script src="../jscalendar/calendar-setup.js" type="text/javascript"
        xml:space="preserve">
        </script>
</head>
<body>
</body>
</html>
```

7) Rutas relativas y absolutes

A la hora de incluir un fichero JavaScript se debe indicar el lugar donde se encuentra este fichero. Dentro de la etiqueta <script> encontramos un parámetro que es SRC donde se indica precisamente esta información.

Las rutas que se incluyen en este parámetro pueden ser relativas o absolutas, se escogerá una opción en función de la aplicación que se está diseñando, aunque se aconseja utilizar siempre rutas relativas, de esta manera, si la aplicación se ha de migrar a otro servidor donde hay variación de rutas, no se deberá modificar ninguna ruta porque la página web, por defecto, cogerá las nuevas rutas.

Ejemplo de ruta relativa:

```
<script src="../jscalendar/calendar.js" type="text/javascript"
xml:space="preserve" />
```

En este caso el fichero "calendar.js" al que hace referencia esta parte del código se encuentra en la carpeta "jscalendar" la cual cuelga de otra carpeta que será la raíz donde se encuentra la página web.

Ejemplo de ruta absoluta:

```
<script src="/var/www/intraproven//jscalendar/calendar.js" type="text/javascript"
xml:space="preserve" />
```

En este caso se ha incluido la ruta absoluta, el fichero "calendar.js" se encuentra directamente en "**/var/www/intraproven//jscalendar/calendar.js**" y en el caso de que se realice una migración, la aplicación debería ser modificada para incluir las nuevas rutas