DISEÑO Y DESARROLLO WEB

VARIABLES, OPERADORES Y MÉTODOS

¿QUÉ SON LAS VARIABLES?

Una VARIABLE es un elemento que se emplea para almacenar y hacer referencia a otro valor. Gracias a las variables es posible crear "programas genéricos" que funcionan siempre igual independientemente de los valores concretos utilizados.

Técnicamente, son un espacio reservado en memoria CON NOMBRE para guardar UN VALOR que puede cambiar.

En programación no se podrían hacer programas útiles sin las variables. Por ejemplo un programa que suma dos números podría escribirse como:

```
resultado = 3 + 1
```

Este programa es tan poco útil que sólo sirve cuando el primer número de la suma sea el 3 y el segundo número el 1. En cualquier otro caso, el programa obtiene un resultado incorrecto.

USANDO VARIABLES

El programa se puede rehacer de la siguiente manera utilizando variables para almacenar y referirse a cada número:

```
numero_1 = 3
numero_2 = 1
resultado = numero_1 + numero_2
```

Los elementos numero_1 y numero_2 son variables que almacenan los valores que utiliza el programa. Si se modifica el valor de las variables el programa sigue funcionando correctamente.

CREACIÓN DE LAS VARIABLES

Las variables en JavaScript se crean mediante la palabra reservada var seguida por el nombre que queramos darle:

```
var numero_1 = 3
var numero_2 = 1
var resultado = numero_1 + numero_2
```

A esto también se le llama DECLARACION de una variable.

SINTAXIS DE LAS VARIABLES

Al crearla, como todavía no tiene nada guardado dentro, su contenido es indefinido:

```
var nombre;
```

sin contenido, es undefined

Una vez declarada, NO se vuelve a usar la palabra var cada vez que se utilice la variable.

SINTAXIS DE LAS VARIABLES

Puede usarse el español para nombrarlas, pero el nombre NO puede:

- empezar con un número
- contener espacios
- contener acentos
- contener eñes
- contener caracteres especiales, solo se pueden usar el $ y el _.
- utilizar palabras reservadas del lenguaje

SINTAXIS DE LAS VARIABLES

Si queremos nombres de más de una palabra, utilizamos los sistemas de notación camelCase o snake_case, por ejemplo:

```
var primerNombre
```

o

```
var primer_nombre.
```

GUARDANDO VALORES EN VARIABLES

- Técnicamente, inicializar una variable es guardarle su primer valor dentro, en ese momento se define su TIPO DE DATO (que no cambiará).
- Se llama PROCESO DE ASIGNACIÓN DE VALOR al proceso mediante el cual se guarda un valor dentro de una variable (sea el primer valor o para reemplazar el que tenga actualmente).
- Se utiliza el operador =

```
nombre = “Juan”; //nombre de la variable “nombre” con valor “Juan”
```

GUARDANDO VALORES EN VARIABLES

SIEMPRE “lo que esté” a la derecha del = se guarda en “lo que esté” a su izquierda:

```
var apellido = ‘Perez’;  //un valor directo

var total = 7 + 5;  //se calcula y se asigna el resultado

var iva = precio * 0.21; //en el cálculo se puede usar el valor de otra variable
```

TIPOS DE VARIABLES

- Numéricas enteras (INTEGER) o decimales (FLOAT, usan el . como separador)
- Cadenas de texto (STRING, se encierran entre comillas simples o dobles)
- Vectores (ARRAYS, se encierran varios valores separados por comas entre [ ])
- Booleanos (true o false)

Las variables SOLO pueden contener UN valor. Si luego le asignamos otro, se reemplaza el anterior.

- crear un documento HTML
- escribir un bloque script el mensaje, almacenándolo en una variable:
  - Hola mundo.
  - Estamos “Programando con JavaScript”
- Mostrarlo por consola y con un alert
- Comentar el código explicando

OPERADORES

Se utilizan para manipular el valor de las variables, realizar operaciones matemáticas con sus valores y compararlas.

De esta forma, los operadores permiten a los programas realizar cálculos complejos y tomar decisiones lógicas en función de comparaciones y otros tipos de condiciones.

Son caracteres que el lenguaje conoce y con los que realiza una acción a partir de ellos.

OPERADORES: ASIGNACIÓN

El operador de asignación es el más utilizado y el más sencillo. Este operador se utiliza para guardar un valor específico en una variable. El símbolo utilizado es =

```
var numero1 = 3;
var numero2 = 4;
numero1 = 5; // Ahora, la variable numero1 vale 5
numero1 = numero2; // Ahora, la variable numero1 vale 4
```

OPERADORES: INCREMENTO Y DECREMENTO

Estos dos operadores solamente son válidos para las variables numéricas y se utilizan para incrementar o decrementar en una unidad el valor de una variable.

```
var numero = 5;
++numero; // numero = 6
var numero2 = 4
--numero2; // numero2 = 3
```

OPERADORES: MATEMÁTICOS

Estos cinco operadores solamente son válidos para las variables numéricas y se utilizan para realizar cálculos con números enteros o flotantes:

```
var suma = 1 + 2; // 3
var resta = 4 - 3; // 1
var multiplica = 4 * 4; // 16
var divide = 9 / 3; // 3
var restoDeDivision = 10 % 3; // 1
```

OPERADORES: MATEMÁTICOS

Los operadores matemáticos también se pueden combinar con el operador de asignación:

```
var numero1 = 5;
numero1 += 3;  // numero1 = numero1 + 3 = 8
numero1 -= 1;  // numero1 = numero1 - 1 = 4
numero1 *= 2;  // numero1 = numero1 * 2 = 10
numero1 /= 5;  // numero1 = numero1 / 5 = 1
numero1 %= 4;  // numero1 = numero1 % 4 = 1
```

OPERADORES: RELACIONALES

Los operadores relacionales definidos por JavaScript son idénticos a los que definen las matemáticas: mayor que (>), menor que (<), mayor o igual (>=), menor o igual (<=), igual que (==), exactamente igual que (===) y distinto de (!=).

El resultado de todos estos operadores SIEMPRE es un valor BOOLEANO.

OPERADORES: RELACIONALES

```
var numero1 = 3;
var numero2 = 5;
resultado1 = numero1 > numero2; // resultado1 = false
resultado2 = numero1 < numero2; // resultado2 = true
```

OPERADORES: RELACIONALES

```
numero1 = 5;
numero2 = 5;
resultado1 = numero1 >= numero2; // resultado1 = true
resultado2 = numero1 <= numero2; // resultado2 = true
resultado3 = numero1 == numero2; // resultado3 = true
resultado4 = numero1 != numero2; // resultado4 = false
```

OPERADORES: LÓGICOS

Los operadores lógicos son imprescindibles para realizar aplicaciones complejas, ya que se utilizan para tomar decisiones sobre las instrucciones que debería ejecutar el programa en función de ciertas condiciones.

El resultado de cualquier operación que utilice operadores lógicos SIEMPRE es un valor lógico o BOOLEANO.

OPERADORES: LÓGICOS

NEGACIÓN

Uno de los operadores lógicos más utilizados es el de la negación. Se utiliza para obtener el valor contrario al valor de la variable:

```
var visible = true;

alert(!visible);  // Muestra "false" y no "true"
```

OPERADORES: LÓGICOS

AND

Obtiene su resultado combinando dos valores booleanos. El operador se indica mediante el símbolo && y su resultado solamente es true si los dos operandos son true:

```
var valor1 = true;
var valor2 = false;
var valor3 = true;
resultado1 = valor1 && valor2; // resultado1 = false
resultado2 = valor1 && valor3; // resultado2 = true
```

OPERADORES: LÓGICOS

OR

La operación lógica OR también combina dos valores booleanos. El operador se indica mediante el símbolo || y su resultado es true si alguno de los dos operandos es true:

```
var valor1 = true;
var valor2 = false;
var valor3 = false;
resultado1 = valor1 || valor2; // resultado1 = true
resultado2 = valor2 || valor3; // resultado2 = false
```

MÉTODOS:

Método alert( ):

Ventana emergente (solo botón aceptar) para mostrar datos de variables o mensajes (string). Se detiene la ejecución del script hasta que se cierre: [NO devuelve datos]

```
alert(“Esto es un mensaje”); //pasa un string
alert(nombre); //pasa el valor de la variable nombre
```

MÉTODOS:

Método console.log( )

Imprime mensajes o variables en la consola del navegador, entre los paréntesis va lo que queremos imprimir:

```
console.log(“Esto es un mensaje”); //pasa un string
console.log(numero); //pasa el valor de la variable numero
```

MÉTODOS:

Método prompt( ):

Ventana emergente (botones aceptar/cancelar, campo de ingreso de texto y mensaje) para ingresar datos, se necesita guardar el ingreso en un variable para no perderlo: [devuelve el dato ingresado siempre como string o null si se cancela o cierra]

```
var ingreso = prompt( “Mensaje a mostrar” , “valor por defecto” );
//el valor por defecto es OPCIONAL
```

MÉTODOS:

Métodos parseInt( ) [entero] parseFloat( ) [decimal]:

Convertir texto a número [“2” (caracter 2) → 2 (número 2)]

```
ingreso= parseInt(ingreso);
edad = +prompt( “Ingrese su edad”);
//el operador + convierte el ingreso directamente a número
```

MÉTODOS:

Método confirm()

Ventana emergente de confirmación (botones aceptar/cancelar y mensaje). Devuelve un valor booleano de acuerdo al botón presionado, por lo que es necesario guardarlo en una variable. [aceptar → true / cancelar → false]

```
var respuesta = confirm( ‘Acepta los términos?‘ );
```

MÉTODOS:

Método document.write( )

Escribe el código fuente html, se pueden generar etiquetas html tratadas como string (de lo contrario generan un error al poseer en su sintaxis los operadores <>).

```
document.write(edad);
document.write(“<p>Esto es un párrafo</p>”);
document.write(“<p>Tu edad ingresada es: “ + edad + ”</p>”);
```

MANOS A LA OBRA