Introducción a JavaScript

Francisco Javier Arce Anguiano

2. Variables y tipos de datos

Objetivos: Al finalizar la unidad el participante conocerá el manejo de variables en JavaScript.

Todos los lenguajes de programación necesitan en algún momento cargar en memoria los datos que se van a procesar. Las variables son fundamentales.

2.1. Introducción a las variables en JavaScript

JavaScript admite prácticamente cualquier tipo de nombre para definir una variable, no obstante, hay una serie de consideraciones que se deben tener presentes:

El primer carácter debe ser siempre una letra o el guión subrayado (_). Los restantes caracteres pueden ser letras, números o el guión subrayado, teniendo como precaución no dejar espacios entre ellos.

El nombre de la variable no debe coincidir con las palabras reservadas de JavaScript. JavaScript diferencia entre mayúsculas y minúsculas. Para declarar variables se utiliza la palabra clave ***var*** seguida del nombre de la variable. Las siguientes variables serán reconocidas como tales por JavaScript.

```
var nombre;
var dirección;
var entrada_valor_concreto;
var variable_numero_12;
```

Ahora se muestran otras variables que no serán reconocidas por JavaScript al no cumplir algunas de las reglas de definición vistas anteriormente.

```
var 1dato;
 var entrada datos;
 var while;
 var new;
```

Se recomienda utilizar siempre la misma pauta para definir los nombres de las variables. Se puede escribir en minúsculas, o bien la primera mayúscula y las demás minúsculas. Aunque las siguientes variables parezcan iguales, JavaScript las interpretará como diferentes.

```
 var resultadosuma
 var Resultadosuma
 var resultadoSuma
 var RESULTADOSUMA
 var resultado suma
 var resultadosumA
```

2.2. Tipos de datos en JavaScript

JavaScript puede manejar tres tipos de datos distintos decidiendo por nosotros el tipo de variable que deberá emplear durante la ejecución del *script*.

Los tres tipos de variables son:

- Variables de cadena
- Variables numéricas
- Variables booleanas

Un cuarto tipo podrían ser los datos Nulos (***null***). Estos se utilizan para comprobar si a una variable se le ha asignado un valor o no. ***Null*** representa un valor nulo para cualquier tipo de variable; por el contrario, una variable que no ha sido iniciada tiene un valor *undefined*.

2.3. Cadenas en JavaScript

Una variable de cadena es aquella que contiene texto. Las cadenas de texto en JavaScript se delimitan mediante comillas dobles o simples y pueden contener cualquier tipo de carácter. También pueden tener un valor nulo. Ejemplo:

```
var pais="México";
var entrada_codigo;
entrada_codigo="54054";
var valor=" ";
```

Las comillas simples serán utilizadas dentro de fragmentos de código delimitados por comillas dobles o viceversa.
Ejemplo:

```
document.write("Que quiere decir la palabra 'nuncupatorio' ")
alert('Pulsa la tecla "amigo" ')
```

Hay una serie de caracteres que permiten representar funciones especiales, como podría ser un salto de línea en un texto o, por ejemplo, las comillas. A continuación se presenta una lista:

```
    \b    carácter anterior
    \f    salto de página
    \n    salto de línea
    \r    retorno de carro
    \t    tabulador
    \\    carácter
    \'    comilla simple
    \"    comilla doble
```

```
<html>
<head>
```

```
 <title>Ejemplo de Cadena de texto</title>
 <script language="javascript">
 var cadena1="Esto es una cadena de texto que no utiliza ningún
caracter especial";
 var cadena2="Esto es una cadena \nde texto que si utiliza
\ncaracateres especiales";
 alert (cadena1);
 alert (cadena2);
  </script>
  </head>
 <body>
 </body>
 </html>
```

2.4. Tipos numéricos en JavaScript

Las variables numéricas son aquellas que contienen números enteros o de coma flotante.

Enteros

JavaScript puede utilizar tres bases distintas para números enteros: la decimal (base 10), la hexadecimal (base 16) y la octal (base 8). Por defecto el sistema es el decimal.

```
 var numero;
 numero = 100;
 numero = -1000;
```

Si queremos especificar un número en base octal deberemos anteponer un cero 0 al número. Recordemos que los números en base octal solo pueden contener los dígitos del 0 al 7.

```
 var numero_octal;
 numero_octal = 03254;
```

```
numero_octal = 02;
```

Para un valor hexadecimal deberemos anteponer al número el prefijo 0x. Los números en hexadecimal incluyen los dígitos del 0 al 9 y las letras comprendidas entre la A y la F ambas inclusive.

```
var numero_hex;
numero_hex = 0xff;
numero_hex = 0x12f;
```

Coma flotante

Un valor de coma flotante se compone de un valor entero seguido de un punto y de una fracción decimal. El exponente se indica mediante una e o E seguida por un entero positivo o negativo.

```
var numero_coma_flotante;
numero_coma_flotante=10.236;
numero_coma_flotante=43.433e+2;
numero_coma_flotante= -56.25;
```

2.5. Variables Booleanas

Las variables booleanas o lógicas se especifican mediante los valores verdadero (true) o falso (false).

```
var estoy_contento;
estoy_contento=false;
estoy_contento=true;
```

La utilización de tipos booleanos es de suma importancia cuando se quieren comparar datos o tomar decisiones.

2.6. Operadores matemáticos

Los operadores aritméticos son los encargados de realizar las operaciones básicas como sumar, restar, multiplicar y dividir.

- Suma (+)
- Resta (-)
- Multiplicación (*)
- División (/)
- Módulo (%). Resto de la división.
- Incremento, Preincremento, Posincremento (++)
- Decremento, Predecremento, Posdecremento (--)
- Negación (!)

Operador Aritmético	Código ejemplo	Descripción
+	valor1 + valor2	Suma
-	valor1 – valor2	Resta
*	valor1 * valor2	Multiplicación
/	valor1 / valor2	División
%	valor1 % valor2	Resto división
++	++ valor1	Preincremento
++	valor1 ++	Posincremento
--	-- valor1	Predecremento
--	valor1 --	Posdecremento
-	- valor1	Negación

```
var valor1, valor2, valor3;
 var suma, resta, multiplicacion, division, resto_division, varios;
 valor1=50;
 valor2=10;
 valor3=20;
 suma=valor1+valor2;
 resta=valor1-valor2;
 multiplicacion=valor1*valor2;
 division=valor1/valor2;
 resto_division=valor1%valor2
 varios=(valor1+valor3)*valor2;
 document.write("El resultado de la suma es"+suma+"<br>");
 document.write("El resultado de la resta es"+resta+"<br>");
 document.write("El resultado de la multiplicacion
es"+multiplicacion+"<br>");
 document.write("El resultado de la division es"+division+"<br>");
 document.write("El resto de la division es"+resto_division+"<br>");
 document.write("El resultado de la variable varios
es"+varios+"<br>");
```

Las variables valor1, valor2, valor3 son las encargadas de contener los números con los que se van a realizar las operaciones aritméticas básicas. Los resultados de dichas operaciones aritméticas básicas. Los resultados de dichas operaciones se guardan en una variable a la que se le ha dado el mismo nombre que el operador utilizado. Así pues, la variable que contiene el resultado de sumar dos números se llama suma, y el resultado de la sustracción se almacena en resta y así sucesivamente.

2.7. Operadores de comparación

Los operadores de comparación son similares a los lógicos, solo que estos no tiene porque ser booleanos. Son los clásicos mayor que, menor que.

== Devuelve true si los dos operandos son iguales.
!= Devuelve true si los dos operandos son diferentes.
> Devuelve true si el operando de la izquierda es mayor que el de la derecha
< Devuelve true si el operando de la izquierda es menor que el de la derecha.

>= Devuelve true si el operando de la izquierda es mayor o igual que el de la derecha.
<= Devuelve true si el operando de la izquierda es menor o igual que el de la derecha.
=== Igualdad estricta
!== Desigualdad estricta

Operador Comparación	Código ejemplo	Descripción
==	valor1 == valor2	Igualdad
! =	valor1 != valor2	Distinto de
<	valor1 < valor2	Menor que
<=	valor1 <= valor2	Menor o igual que
>	valor1 >= valor2	Mayor que
>=	valor1 >= valor2	Mayor o igual que

2.8. Operadores lógicos

Los operadores lógicos o booleanos se emplean para que un programa tome una decisión en función de un cálculo lógico. Los valores que se obtienen son true o false. Los operadores son los siguientes:

- && Suma lógica (Y o And). Este operador devuelve un valor true cuando las dos condiciones comparadas se cumplen. En el supuesto de que una de ellas sea false, el resultado será siempre false.
- || Producto lógico (O u Or). Este operador devuelve true siempre que una de las dos condiciones sea verdadera. En caso contrario devuelve false.
- ! Negación (No o Not). Este operador devuelve siempre el valor contrario, es decir, si la condición o variable es true devuelve false y viceversa.

La verdadera utilidad de estos operadores se ve al trabajar con estructuras condicionales. A continuación verá un resumen con todos los operadores lógicos.

```
true && true   devuelve true
true && false devuelve false
false && false devuelve false
true || true     devuelve true
true || false    devuelve true
false || false   devuelve false
```

Operador lógico	Código ejemplo	Descripción
&&	valor1 && valor2	AND lógico
||	valor1 || valor2	OR lógico
!	! valor1	NOT lógico

2.9. Operadores unarios y atajos

Otro tipo de operadores aritméticos son los incrementales o decrementales, que se aplican antes o después del operando.

```
variable++  //Devuelve el valor de variable y después incrementa su valor en uno.
++variable  //Aumenta el valor de variable en uno y después devuelve su valor.
variable --  //Devuelve en valor de variable y después disminuye su valor en uno.
-- variable  //Disminuye el valor de variable en uno y después devuelve su valor.
```

Veamos la eficacia de este tipo de operadores:

```
 pesos=pesos+1; //Aumenta el valor de pesos en 1
 pesos++;  //Aumenta el valor de pesos en 1
          //El último operador aritmético es la negación.
numero+=100;
```

```
numero= -numero;
```

Un operador de asignación se utiliza para asignar un valor a una variable. Veamos cuáles son, y su sintaxis.

+=	valor1 += valor2	valor1 = valor1 + valor2
-=	valor1 -= valor2	valor1 = valor1 – valor2
*=	valor1 *= valor2	valor1 = valor1 * valor2
/=	valor1 /= valor2	valor1 = valor1 / valor2
%=	valor1 %= valor2	valor1 = valor1 % valor2

Operador Asignación	Código ejemplo	Descripción
+=	valor1 += valor2	valor1=valor1+valor2
-=	valor1 -= valor2	valor1=valor1-valor2
*=	valor1 *= valor2	valor1=valor1*valor2
/=	valor1 /= valor2	valor1=valor1/valor2
%=	valor1 %= valor2	valor1=valor1%valor2

2.10. Palabras reservadas

En la gran mayoría de los lenguajes de programación contamos con ciertas palabras que no las podemos utilizar para crear variables o funciones, las cuales las llamamos como palabras reservadas para el lenguaje. En el caso de JavaScript no podemos utilizar las siguientes palabras como variables, funciones u objetos:

abstract	boolean	break	byte
case	match	char	class
const	continue	default	do

doublé	else	extendí	false
final	finally	float	for
function	goto	int	implements
input	in	instanceof	interface
long	native	new	null
package	private	protected	public
return	short	static	super
switch	this	throw	synchronized
throws	transient	true	try
var	val	while	with

2.11. Manejo de espacios en blanco

Podemos colocar espacios en blanco entre los operadores y los operandos para mayor legibilidad del código.
Los espacios entre líneas no afectan, pero no son recomendables.
https://jscompress.com o https://javascript-minifier.com .

2.12. Operadores de bits

Un ***operador bit a bit*** trata a sus operandos como un conjunto de ***32 bits*** (ceros y unos), en lugar de números decimales, hexadecimales u octales. Por ejemplo, el número decimal ***nueve*** tiene una representación binaria de ***1001***. Los operadores bit a bit realizan sus operaciones en tales representaciones binarias, pero devuelven valores ***numéricos estándar*** de JavaScript.

Operador Bit a bit	Código ejemplo	Descripción
&	valor1 & valor2	AND de bits
|	valor1 | valor2	OR de bits

^	valor1 ^ valor2	XOR de bits
>>	valor1 >> valor2	Shift a la derecha
<<	valor1 << valor2	Shift a la izquierda
>>>	valor1 >>> valor2	Shift sin signo

```
var a = 5; //00000000000000000000000000000101
var b = 2; //00000000000000000000000000000010
console.log(a&b);
console.log(a|b);
```

2.13. El operador typeof()

El operador ***typeof*** devuelve una cadena que describe el tipo de dato del operando. Así pues, podremos saber si el operando es una cadena, una variable, método, etc. Su funcionamiento es sencillo, basta con escribir a continuación de ***typeof*** la variable o elemento.

```
var variable = "Pepe";
var numero = 1;
var fecha = new Date();
document.write(typeof variable + "<br>");
document.write(typeof numero + "<br>");
document.write(typeof fecha + "<br>");
```

El resultado es:

```
string
number
object
```