



JavaScript



Módulo 3



1. Variables y constantes: tipología, etapas y alcance.
2. Operadores: clasificación, rangos y usos.



Variables JavaScript.

Las variables son contenedores para almacenar datos
(almacenar valores de datos).





Variables JavaScript.

Existen 4 formas de declarar una variable en JavaScript:

1. Usando **var**
2. Usando **let**
3. Usando **const**
4. Usando nada

Igual que en el álgebra:

```
1  
2  var x = 10;  
3  var y = 5;  
4  var z = x + y;  
5
```

Veamos ejemplos.



Uso de var.

- Declare siempre las variables de JavaScript con **var**, **let** o **const**.
- La palabra reservada **var** se usa en todo el código JavaScript desde 1995 hasta 2015.
- Las palabras clave **let** y **const** se agregaron a JavaScript en 2015.
- Si desea que su código se ejecute en un navegador anterior, debe usar **var**.



Constantes JavaScript.

Las variables constantes presentan un **ámbito de bloque** (block scope) igual que las variables definidas usando la instrucción **let**.

La diferencia es que el valor de una constante no puede cambiarse a través de la reasignación. Las constantes **no se pueden redeclarar**.
Veamos un ejemplo.





Identificadores JavaScript.

Todas las **variables** de JavaScript deben **identificarse** con **nombres únicos** .

Lo ideal es que los **identificadores** sean **descriptivos**, por ejemplo: **edad**, **suma**, **volumen**, **total**, son nombres que dicen algo por sí mismos.

```
1  
2  var x=10;  
3  var suma=x+2;  
4  var total=suma*1.16;  
5
```



Reglas generales para nombramiento de variables.

1. Los nombres pueden contener letras, dígitos, guion bajo y signo de dólar.
2. Los nombres deben comenzar con una letra.
3. Los nombres también pueden comenzar con \$ y _.
4. Los nombres distinguen entre mayúsculas y minúsculas.
5. Las palabras reservadas no se pueden usar como nombres.

Existen ciertos estilos de
naming, veamos algunos.



Uso de `let`.

- La palabra reservada `let` se introdujo en ES6 (2015).
- Las variables definidas con `let` no se pueden volver a declarar.
- Las variables definidas con `let` deben declararse antes de su uso.
- Las variables definidas con `let` tienen ámbito de bloque.

Veamos ejemplos.



Scope de una variable en JavaScript.

Dependiendo del lugar donde se declare una variable, ésta puede ser temporal, ya sea dentro de un bloque o de una función.

En programación la zona en la que está activa una variable se llama *ámbito* o *alcance(scope)*.

Clasificación de el scope de
una variable en JavaScript.

Global scope.

Function scope.

Block scope



Function scope.



Solo se puede acceder a ellas desde dentro de la función.

```
2  // Aquí no puedes usar la variable carName
3
4  function myFunction() {
5      let carName = "Volvo";
6      // Aquí sí puedes usar la variable carName
7  }
8
9  // Aquí no puedes usar la variable carName
10
```



Global scope.



Todos los scripts y funciones de una página web pueden acceder a ella.

```
2  let carName = "Volvo";  
3  // Aquí sí puedes usar carName  
4  
5  function myFunction() {  
6    // Aquí también puedes usar carName  
7  }  
8
```



Block scope.

- Antes de ES6 (2015), JavaScript solo tenía **Global Scope** y **Function Scope**.
- ES6 introdujo dos nuevas e importantes palabras reservadas de JavaScript: **let** y **const**.
- Estas dos palabras clave proporcionan **Block Scope** en JavaScript.
- No se puede acceder a las variables declaradas dentro de un bloque { } desde fuera del bloque.

Veamos un ejemplo.



Tipos de datos en JavaScript.

Las variables de JavaScript pueden contener diferentes tipos de datos:
números, cadenas, objetos y más...

Sin tipos de datos, una computadora no puede resolver esto de manera segura:

```
3 let x = 16 + "Volvo";
```



Los tipos de datos en JavaScript son dinámicos.

Esto significa que la misma variable se puede utilizar para contener diferentes tipos de datos:

```
3  let x;           // Por ahora x está indefinida.
4  x = 10;          // Por ahora x es un número.
5  x = "Erick";     // Por ahora x es una cadena.
```



Cadenas JavaScript.

Una cadena (o una cadena de texto) es una serie de caracteres como “Hola a todos”.

```
2 let carName1 = "Mustang Mach-E"; // Usando comillas dobles
3 let carName2 = 'Mustang Mach-E'; // Usando comillas simples
```




Números JavaScript.

JavaScript tiene un solo tipo de números.
Los números se pueden escribir con o sin decimales:

```
3 let x1 = 34.00;    // Número con decimales
4 let x2 = 34;       // Número sin decimales
```

Los números extra grandes o extra pequeños se pueden escribir con notación científica (exponencial):

```
3 let y = 456e5;     // 45600000
4 let z = 456e-6;    // 0.000456
```



Booleanos JavaScript.

Los valores booleanos solo pueden tener dos valores: true o false.

```
2 let a = 8;
3 let b = 8;
4 let c = 7;
5 let valor1=(a == b) // Se asigna true (verdadero)
6 let valor2=(a == c) // Se asigna false (falso)
```

Los booleanos se utilizan a menudo en pruebas condicionales.



Operadores JavaScript.

```
3 let x = 5;      // Asignando el valor 5 a x
4 let y = 2;      // Asignando el valor 2 a y
5 let z = x + y;  // Asignando el valor 7 a z (5 + 2)
```

El operador de **asignación** = asigna un valor a una variable.

JavaScript tiene tres tipos de operadores *binarios* y *unarios*, y un operador *ternario* especial, el operador condicional.

Un operador binario requiere dos operandos, uno antes del operando y otro después del operador:

operando1 operador operando2



Operadores aritméticos.

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
**	Potencia (Desde ES2016)
/	División
%	Modulo (División resto)
++	Incremento paso 1
--	Decremento paso 1



Ejercicios.

1. Pedir un monto sin IVA e imprimir en la pantalla del navegador el monto con IVA usando cualquier método.
2. Pedir un monto e imprimir en la ventana del navegador el monto ya con el 15% de descuento aplicado.



Operadores de asignación.

Operador	Ejemplo	Equivalencia
=	$x = y$	$x = y$
+=	$x += y$	$x = x + y$
-=	$x -= y$	$x = x - y$
*=	$x *= y$	$x = x * y$
/=	$x /= y$	$x = x / y$
%=	$x \% = y$	$x = x \% y$
**=	$x ** = y$	$x = x ** y$



Ejercicios.

1. Asignar un ahorro por cada día de la semana y acumular la suma en una variable, finalmente imprimir el ahorro en la ventana del navegador.



Operadores de cadenas.

El operador + también se puede usar para agregar (concatenar) cadenas.

```
2 let nombre = "Pedro";  
3 let apellido = "Dorantes";  
4 let nombreCompleto = nombre + " " + apellido;
```

Veamos el resultado.



Ejercicio.

1. Crear 3 variables que guarden datos de una persona e imprimir en consola sus datos concatenándolos en una sola variable.

Ejemplo:





Operadores de comparación.



Operador	Nombre
<code>==</code>	Igual que
<code>===</code>	Identico
<code>!=</code>	Diferente
<code>!==</code>	No identico
<code>></code>	Mayor que
<code><</code>	Menor que
<code>>=</code>	Mayor o igual que
<code><=</code>	Menor o igual que
<code>?</code>	Operador ternario



Operadores lógicos.

Operador	Nombre
&&	AND
	OR
!	NOT

Hagamos sus tablas de verdad.