

JS



PROFE VIDERMID

# Programación en Java Script

## (Parte 1)

Facilitador: Ing. Esp. Vidermid Sánchez



@vidermid



+584147106623



@ingenieriadigitalsc



+584147464801





**JS**



PROFE VIDERMID

**BIENVENIDOS A LA  
PROGRAMACIÓN  
CON**

**JS**

# ¿QUÉ ES ECMASCRIPT 6 (ES6)?

Ecma International es una organización internacional basada en membresías de estándares para la comunicación y la información. Adquirió el nombre Ecma International en 1994, cuando la European Computer Manufacturers Association (ECMA) cambió su nombre para expresar su alcance internacional.

Cuando hablamos de ECMAScript, nos referimos al estándar que ya va por la versión ES6 y determina cómo emplear el lenguaje Javascript, que permite a los fabricantes de software desarrollar las herramientas adecuada para interpretarlo correctamente.

Hoy en día, existen múltiples plataformas que interpretan y procesan Javascript, como por ejemplo NodeJS o los navegadores web. También es utilizado para el desarrollo de aplicaciones de diferentes sistemas operativos. Es el estándar ECMAScript el que marca cómo deberá ser interpretado el lenguaje en cada una de estas tecnologías.



**alert()** permite generar ventanas emergentes para mostrar mensajes y/o valores.  
Ejemplo: `alert('Hola María');`

**prompt()** permite generar ventanas emergentes con cuadro de diálogo.  
Ejemplo: `prompt('Por favor indicar su Nombre:');`

**document.write()** permite escribir mensajes y/o valores dentro del documento html.  
Ejemplo: `document.write('Bienvenido a mi página web');`

**console.log()** permite escribir mensajes y/o valores dentro de la consola del navegador web.  
Ejemplo: `console.log('Esta es la consola');`

En Java Script existen tres formas de expresar mensajes de pantalla, sin importar que sean con las funciones `prompt()`, `alert()`, `document.write()` o `console.log()`, ésto se logra con:

Apostrofes → ‘ ‘ *código ASCII alt+39*

Comillas → “ ” *código ASCII alt+34*

Tilde invertida → ` ` *código ASCII alt+96 (no se utiliza para entrada de datos con la función `prompt()`; )*  
(Acento Grave)

Ejemplos:

`console.log(“La suma es: ”+suma);`      `document.write(“La suma es: ”+suma);`      `alert(“La suma es: ”+suma);`

`console.log(‘La suma es: ’+suma);`      `document.write(‘La suma es: ’+suma);`      `alert(‘La suma es: ’+suma);`

`console.log(`La suma es: ${suma}`);`      `document.write(`La suma es: ${suma}`);`      `alert(`La suma es: ${suma}`);`



# MANEJO, ÁMBITO DE CONSTANTES Y VARIABLES (SCOPE)



Java Script es un lenguaje de tipado dinámico, es decir, el intérprete asigna a las variables un tipo durante el tiempo de ejecución basado en su valor en ese momento.

Java Script es Case Sensitive, es decir, diferencia entre mayúsculas y minúsculas, por lo cual habrá que tener mucha precaución a la hora de escribir las expresiones, constantes y variables.

**Constantes:** herramienta que permite almacenar un valor de forma fija mientras el programa está en ejecución. No se puede modificar su contenido. (const)

**Variables:** herramienta que permite almacenar valores y mantenerlos mientras el programador lo establezca. Si se puede modificar su contenido. (var ó let)

**Nombre Valido:** no debe iniciar por números o caracteres especiales, pero si puede contenerlos dentro de su nombre. Debe iniciar por una letra. (los caracteres especiales varían según el lenguaje de programación). No deben contener espacios.

**Scope:** en JS se pueden declarar con let o con var (alcance de la variable).

**Tipos:** cadena, carácter, decimales, enteras, lógicas/booleanas, Undefined, Null, Nan.





Puede realizarse en html5 con las etiquetas `<script>` `</script>` desde el head de la página:

```
<head>  
  <meta charset="UTF-8">  
  <meta http-equiv="X-UA-Compatible" content="IE=edge">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <script>alert('Desde la Cabecera del HTML');</script>  
  <title>JS</title>  
</head>
```

No recomendado porque el html no está estructurado completamente

Puede realizarse en html5 con las etiquetas `<script>` `</script>` en el inicio del body de la página:

```
<body>  
  <script>alert('Desde el inicio del body del HTML');</script>  
  <h1>JS con HTML5</h1>  
  <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Praesentium, et.</p>  
</body>
```

No recomendado porque el html no está estructurado completamente

**Puede realizarse en html5 con las etiquetas `<script>` `</script>` al final del body de la página:**

```
<body>
  <h1>JS con HTML5</h1>
  <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Praesentium, et.</p>
  <script>alert('Desde el final del body del HTML');</script>
</body>
```

Aquí el html ya está estructurado completamente, pero no permite reutilización del código.



**Puede realizarse en html5 referenciando mediante `<script src="lib.js">` `</script>` al final del body de la página:**

```
<body>
  <h1>JS con HTML5</h1>
  <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Praesentium, et.</p>
  <script src="js/lib.js"></script>
</body>
```

Es una buena practica que contribuye a la reutilización del código, además, en el script de JS es mas limpia la sintaxis.



**En el script JS quedaría:**

```
alert('Desde un script JS independiente');
```





Asignación simple (=) Asigna un contenido a una variable o a un objeto. Por ejemplo: `mivariable = "Saludos"`

En JavaScript el operador de asignación tiene la particularidad de que puede combinarse con algunos de los operadores aritméticos, dando lugar a toda una familia de nuevos operadores:

<code>A += B</code>	Ejemplo equivalente:	<code>A = A + B</code>
<code>A -= B</code>	Ejemplo equivalente:	<code>A = A - B</code>
<code>A /= B</code>	Ejemplo equivalente:	<code>A = A / B</code>
<code>A *= B</code>	Ejemplo equivalente:	<code>A = A * B</code>
<code>A %= B</code>	Ejemplo equivalente:	<code>A = A % B</code>
<code>A &amp;= B</code>	Ejemplo equivalente:	<code>A = A &amp; B</code>
<code>A ^= B</code>	Ejemplo equivalente:	<code>A = A ^ B</code>
<code>A  = B</code>	Ejemplo equivalente:	<code>A = A   B</code>

Suma (+). Suma dos números:  $3 + 2 = 5$  . Si en lugar de números se suman cadenas, como por ejemplo "A" + "B" = "AB" se produce una concatenación. Si alguno de los elementos a concatenar no es una cadena de texto, queda convertido a cadena: "AB" + 123 = "AB123" .

Resta (-). Halla la diferencia entre dos números. Ejemplo A:  $3 - 2 = 1$  . Ejemplo B:  $(-1) - (-2) = 1$  .

Negativo (-). Indica el signo negativo de un número o una expresión: -3.

Multiplicación (\*). Multiplica dos números:  $3 * 2 = 6$  .

División (/). Divide dos números, obteniendo un cociente de coma flotante:  $5 / 2 = 2.5$  . Fíjate en que el separador decimal es el punto.

Módulo aritmético %. Divide dos números, obteniendo un resto entero:  $5 \% 2 = 1$  .

Incrementos y decrementos (++ --). Incrementa o decrementa el valor de una variable numérica en una unidad. No puede aplicarse a un literal. Pueden darse dos casos:

```
let A, B; B = 2;
```

```
A = ++B;
```

```
A = B++;
```

En el primer caso,  $A = 3$  Es decir, que el incremento de B ha sido el correcto, pero no así el segundo, en el que  $A = 2$ . Esto se debe a que, en el primer caso, el incremento se efectúa antes de que la expresión sea evaluada, y en el segundo, se incrementa después de la evaluación según indica el orden de precedencia de los operadores

De igual forma, para decrementos:

```
let A, B; B = 2;
```

```
A = --B;
```

```
A = B--;
```

En el primer caso resulta:  $A = 1$  y en el segundo:  $A = 2$

**Igualdad (==)** Verifica la igualdad de dos expresiones sin tener en cuenta el tipo de dato.

*Por ejemplo:* `2 == "2"` devuelve True

**Igualdad estricta (===)** Hace lo mismo que el anterior, pero verificando también que coincidan los tipos de datos

*Por ejemplo:* `2 === "2"` devuelve False

**Desigualdad (!=)** Funciona de la misma forma que la igualdad, pero negándola.

*Por ejemplo:* `2 != "2"` devuelve False

**Desigualdad estricta (!==)** Lo mismo que la igualdad estricta, pero negándola.

*Por ejemplo:* `2 !== "2"` devuelve True

Y estos cuatro, seguro que ya sabes cómo funcionan:

### **Mayor que (>)**

*Por ejemplo:  $2 > 1$  devuelve True*

### **Mayor o igual que (>=)**

*Por ejemplo:  $2 \geq 1$  devuelve True*

### **Menor que (<)**

*Por ejemplo:  $2 < 1$  devuelve False*

### **Menor o igual que (<=)**

*Por ejemplo:  $2 \leq 1$  devuelve False*

**Negación lógica ! (Not).** Establece una negación lógica en una expresión, es decir, que ante una expresión, cuyo estado lógico es True (verdadero), el operador hará que devuelva False (falso).

*Ejemplo:* El hierro es un metal = True.

*Ejemplo:* ! El hierro es un metal = False.

**Conjunción lógica && (And).** Establece una conjunción lógica de dos expresiones, es decir, que han de resultar True (verdadero) las dos expresiones para que el resultado final también lo sea.

*Ejemplo:* El hierro es un metal = True.

*Ejemplo:* El hierro es duro = True.

*Ejemplo:* El hierro es un metal && El hierro es duro = True.

**Disyunción lógica || (Or).** Establece una disyunción lógica de dos expresiones, es decir, que el resultado se dará evaluando una expresión u otra.

*Ejemplo:* El hierro es un metal = True.

*Ejemplo:* El hierro es duro = True.

*Ejemplo:* El hierro es un metal || El hierro es duro = True.



**Typeof:** Este operador nos indica el tipo de dato contenido en un variable, un literal o el resultado de una expresión. Puede devolver seis valores diferentes: number, string, object, function, boolean o undefined.

Ejemplo: `typeof 2` devuelve `number`

**Void:** Este es un curioso operador que se limita a impedir que todo funcione normalmente, es decir, que una vez evaluadas la instrucciones, sus efectos o presentación de resultados serán anulados. Por ejemplo, bloqueando un formulario donde no se han cumplimentado algunos campos obligatorios, o si algún valor es erróneo. El siguiente ejemplo genera un link que no funciona: Este link no funciona

Y se escribe: `<a href="javascript:void(0)">Este link no funciona</a>`

**New:** Sirve para crear una instancia de un objeto definido por el usuario, o también, para crear alguno de los objetos intrínsecos de JavaScript, como son: Array, Boolean, Date, Function, Math, Number o String. Lo veremos con más detalle en la parte del curso de objetos.








*Como puedes ver, JavaScript dispone de una gran variedad de operadores. Cuando se escriben expresiones lógicas complejas es recomendable usar los paréntesis para cada sub expresión participante, y no olvides tener siempre en cuenta el orden de evaluación de los operadores. Es relativamente fácil equivocarse al escribir expresiones complicadas, y es buena idea probar siempre lo escrito con cuantas variaciones sea posible para asegurarse de que no se producen resultados inesperados.*

# ORDEN DE PRECEDENCIA DE LOS OPERADORES (JERARQUÍA)



Operador	Descripción
. [] ()	Acceso a campos, índice de matrices y llamada a funciones.
++ -- - ~ ! delete new typeof void	Incremento +1, decremento -1, negativo, NOT, NOT lógico borrado, crear objeto, mostrar tipo, indefinido
* / %	Multiplicación, división, módulo de división (resto)
+ - +	Suma, resta, concatenación de cadenas
<< >> >>>	Bit shifting
< <= > >=	menor que, menor que o igual, mayor que, mayor que o igual
== != === !==	Igualdad, desigualdad, identidad, no identidad
&	AND
^	XOR
	OR
&&	AND lógico
	OR lógico
?:	Condicional
=	Asignación
,	Evaluación múltiple



- **isNaN()** El método devuelve true si un valor no es un número.  `isNaN(0/0); isNaN(""); isNaN('A');  
isNaN(true); isNaN(false);`
- **Number.isNaN()** devuelve true si un número no es un número.  `Number.isNaN('Hello');`
- **Number()** método que convierte un valor en un número.  
Si el valor no se puede convertir, devuelve NaN.  `Number(true); Number(false);`
- **parseFloat()** método que analiza un valor como una cadena y devuelve el primer número decimal  `parseFloat(67); parseFloat("89");  
parseFloat("25.47");`
- **parseInt()** método que analiza un valor como una cadena y devuelve el primer entero.  `parseInt(" 51"); parseInt(" 73.00");  
parseInt(" 97.81"); parseInt(" 11 22 99");  
parseInt(" 71 ");`
- **String()** método que convierte un valor en una cadena.  `String(Boolean(0)); String([1,2,3,4]);  
String(new Date()); String("12345");`
- **toFixed()** método que redondea la cadena a un número específico de decimales.  `let num = 5.56789;  
let n = num.toFixed(2);`

La sentencia *if* no es más que una sentencia condicional, o sea, que si se cumple "x" condición, el programa hace una cosa, y si no se cumple, el programa hace otra.

Las sentencias *if* se construyen de la siguiente forma:

```
if (condición)
{
    instrucciones...
}
```

Dichas instrucciones solo se ejecutarán si se cumple la condición del IF.

**Ejemplo:**

```
if (a>b)
{
    alert(a+'Es el Mayor');
}
```

A diferencia del *if* simple, el programa hace una cosa, pero en caso contrario podemos especificar lo que queremos que haga o ejecute.

Las sentencias *if* se construyen de la siguiente forma:

**Sintaxis:**

```
if (condición)
{
    instrucciones...
}
else
{
    otras instrucciones...
}
```

**Ejemplo:**

```
if (a>=b)
{
    alert(a+'Es el Mayor o igual');
}
else
{
    alert(b+'Es el Mayor');
}
```

La sentencia switch ejecuta un bloque de código dependiendo de los diferentes casos.

La declaración de cambio es una parte de las declaraciones "condicionales" de JavaScript, que se utilizan para realizar diferentes acciones basadas en diferentes condiciones. Use el interruptor para seleccionar uno de los muchos bloques de código que se ejecutarán. Esta es la solución perfecta para sentencias if/else largas y anidadas

**Sintaxis:**

```
switch(expresión) {  
  case m:  
    bloque de código;  
  break;  
  case n:  
    bloque de código;  
  break;  
  default:  
    bloque de código;  
}
```

**Ejemplo:**

```
switch(carromarca) {  
  case "Ford":  
    text = "Muy duraderos";  
  break;  
  case "Chevrolet":  
    text = "Son buenos y mas económicos";  
  break;  
  case "Ferrari":  
    text = "Lo mejor del mundo";  
  break;  
  default:  
    text = "Marca no registrada o muy baja en ventas";  
}
```



El *for* tiene la siguiente sintaxis:

```
for (<inicialización>; <condición>; <incrementador/decrementador>) {  
    // el cuerpo del ciclo, el código que se repite hasta llegar al numero final de repeticiones  
}
```

Ejemplo:

```
for (var i=0; i < 850; i++) {  
    console.log("Hola mundo");  
}
```

Un ciclo *while* en JavaScript tiene la siguiente sintaxis:

```
while (<condición>) {  
    // acá va el cuerpo del ciclo, el código que se va a repetir mientras la condición se cumpla  
}
```

**Ejemplo:**

```
let res=true;  
while (true) {  
    console.log("Hola Mundo");  
}
```

Con el bucle *do while*, se ejecuta una secuencia de código mientras la condición entre paréntesis sea verdadera. Hay una diferencia notable respecto al bucle *while*. Aquí la secuencia de código se ejecuta al menos una vez, porque la comprobación de la condición se realiza al terminar la primera vuelta del bucle,

***sintaxis:***

```
do
{
    secuencia de código
} while(condición)
```

***Ejemplo:***

```
let a=0;
do
{
    a=a+1;
    document.write(a);
} while(a<20);
```

Elabore un Script JS que permita al usuario ingresar el nombre, el año actual y su año de nacimiento. Se desea calcular, mostrar por pantalla la edad aproximada y un mensaje que indique si es Mayor o menor de edad, según sea el caso.

Elabore un Script JS que permita al usuario ingresar dos números enteros positivos. Se requiere calcular el producto de los valores conocidos y mostrarlos por pantalla. Realizar este proceso 4 veces.

Elabore un Script JS que permita al usuario ingresar número de cédula, nombre, la cantidad de productos de un mismo valor monetario que desea adquirir un cliente y el precio unitario. Se requiere calcular y mostrar por pantalla en monto a cancelar, además sus datos de identificación.

Elabore un Script JS que permita al usuario ingresar el nombre del estudiante, las 4 notas obtenidas en las cuatro asignaturas cursadas en el semestre (la escala debe ser del 1 al 20). Se requiere calcular y mostrar por pantalla el promedio, además visualizar sus datos personales, indicando si está aprobado o reprobado mediante un mensaje adicional, según sea el caso. Realizar el proceso para cada estudiante mientras el usuario lo desee.

Nota: Validar todos los datos de entrada.