

Javascript

¿Qué es Javascript?

JS



Javascript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con Javascript podemos crear diferentes efectos e interactuar con nuestros usuarios.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

Es un lenguaje basado en acciones que posee menos restricciones.

Está centrado en escribir funciones que respondan a esas acciones o eventos como ser: movimientos del mouse, aperturas, utilización de teclas, cargas de páginas entre otros.

Javascript más difundido hoy se ejecuta en el navegador o dispositivo del usuario.

Ya hay implementaciones del lenguaje sobre el servidor, e incluso un modelo de base de datos basado en Javascript.

En HTML, el código JavaScript debe estar entre las etiquetas `<script>` y `</script>`

Se puede colocar dentro de `<head>`, dentro de `<body>` e incluso en archivos externos

Ejemplos

```
<body>  
  <script type="application/javascript">  
    alert("Hola Mundo!!!");  
  </script>  
</body>
```

```
11 <body>  
12   <h1>  
13     <script type="application/javascript">  
14       document.write("Escribimos en el documento!");  
15     </script>  
16   </h1>  
17 </body>
```

También lo podemos centralizar en un archivo externo

```
<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">  
  <title>Javascript</title>  
  <script type="application/javascript" src="js/aplicacion.js"></script>  
</head>
```


Ventajas de usar JS externos

Separamos el código JS y HTML

Ambos, son más legibles y más fáciles de mantener.

Pueden acelerar la carga de la página.

Los archivos externos no requieren etiquetas `<script>`

```
CLASE 5... [📄] [🔍] [🔄] [📄]
  ▾ js
    JS aplicacion.js
  <> index.html

index.html > html > body
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>Javascript</title>
9      <script type="application/javascript" src="js/aplicacion.js"></script>
10 </head>
11
12 <body>
13     <button type="button" onclick="saludo('Buen dia')">Mañana</button>
14     <button type="button" onclick="saludo('Buenas tardes')">Tarde</button>
15     <button type="button" onclick="saludo('Buenas noches')">Noche</button>
16 </body>
17
18 </html>
```

Variables

Se usan para almacenar y referenciar valores.

```
| var nombre;
```

Hemos declarado la variable llamada nombre, pero no le asignamos valor.

```
| var nombre = "Maria";|
```

Y Ahora le dimos un valor a esa variable.

Variables

Tenemos dos formas de declarar variables:

```
var nombre = "Sarah Connor";
```

```
let nombre = "John Connor";
```

Las variables pueden ser numéricas:

var edad=10;

Strings (texto y números)

let nombre="Pablo";

Booleanos

var habilitado = true;

Arrays (arreglo de valores)

var colores = ['rojo','amarillo','azul'];

Constantes

Se usan para almacenar y referenciar valores que no van a cambiar.

```
const nombre = "Trinity";
```

Se le asigna un valor que no será cambiado

Ámbitos Globales y Locales

Depende donde se declare, puede definirse como global o local

```
var i = "global";  
function foo() {  
    i = "local";  
    console.log(i); // local  
}  
foo();  
console.log(i); // local
```

Ámbitos Globales y Locales

Se puede declarar dos variables con el mismo nombre, pero diferentes ámbitos

```
var i = "global";  
function foo() {  
    var i = "local"; // Otra variable local solo para esta función  
    console.log(i); // local  
}  
foo();  
console.log(i); // global
```


Operadores aritméticos

precio = 128 //introduzco un 128 en la variable precio

unidades = 10 //otra asignación

factura = precio * unidades //multiplico precio por unidades,
obtengo el valor factura

resto = factura % 3 //obtengo el resto de dividir la variable
factura por 3

precio++ //incrementa en una unidad el precio (ahora vale 129)

Operadores de asignación

ahorros = 7000 //asigna un 7000 a la variable ahorros

ahorros += 3500 //incrementa en 3500 la variable ahorros,
ahora vale 10500

ahorros /= 2 //divide entre 2 mis ahorros, ahora quedan
5250

De cadena y negación

```
cadena1 = "hola"  
cadena2 = "mundo"  
cadenaConcatenada = cadena1 + cadena2 //cadena  
concatenada vale "holamundo"
```

```
estado = !false // ! Es la negación
```

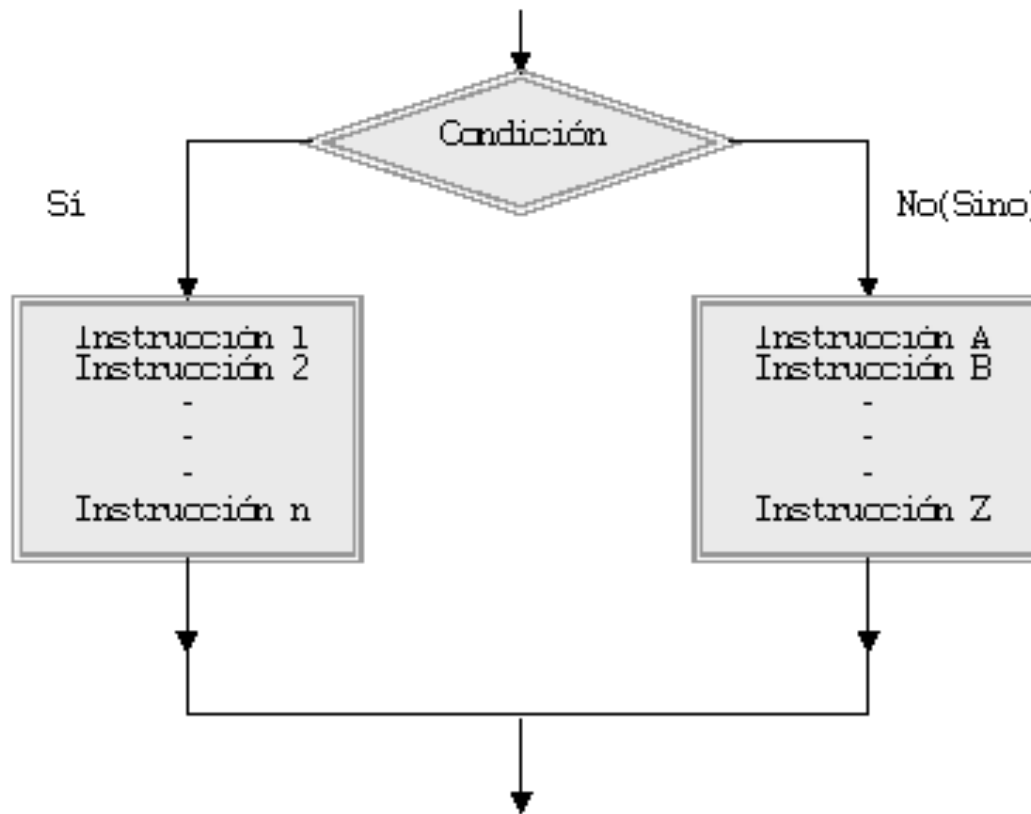
Template String

```
const nombre = "Sarah";  
const apellido = "Connor";  
  
// Template String  
const completo = `${nombre} ${apellido}`;  
  
console.log(completo);
```

Definir una variable numérica, asignarle un valor y sumarle 5.

Definir dos variables de cadenas, asignarles valores y concatenarlas.

Condicional IF



```
if (condición) {  
    instrucciones si  
    se cumple la  
    condición  
} else {  
    ... si no  
    se cumple  
}
```

Operadores lógicos

! Operador NO o negación. Si era true pasa a false y viceversa.

A == B Comprueba si dos valores son iguales.

A != B Comprueba si dos valores son distintos.

A > B Mayor que, devuelve true si el primer operando es mayor.

A < B Menor que, es true cuando el elemento de la izquierda es menor.

A >= B Mayor igual.

A <= B Menor igual.

&& Operador Y, si son los dos verdaderos vale verdadero.

|| Operador O, vale verdadero si por lo menos uno de ellos es verdadero.

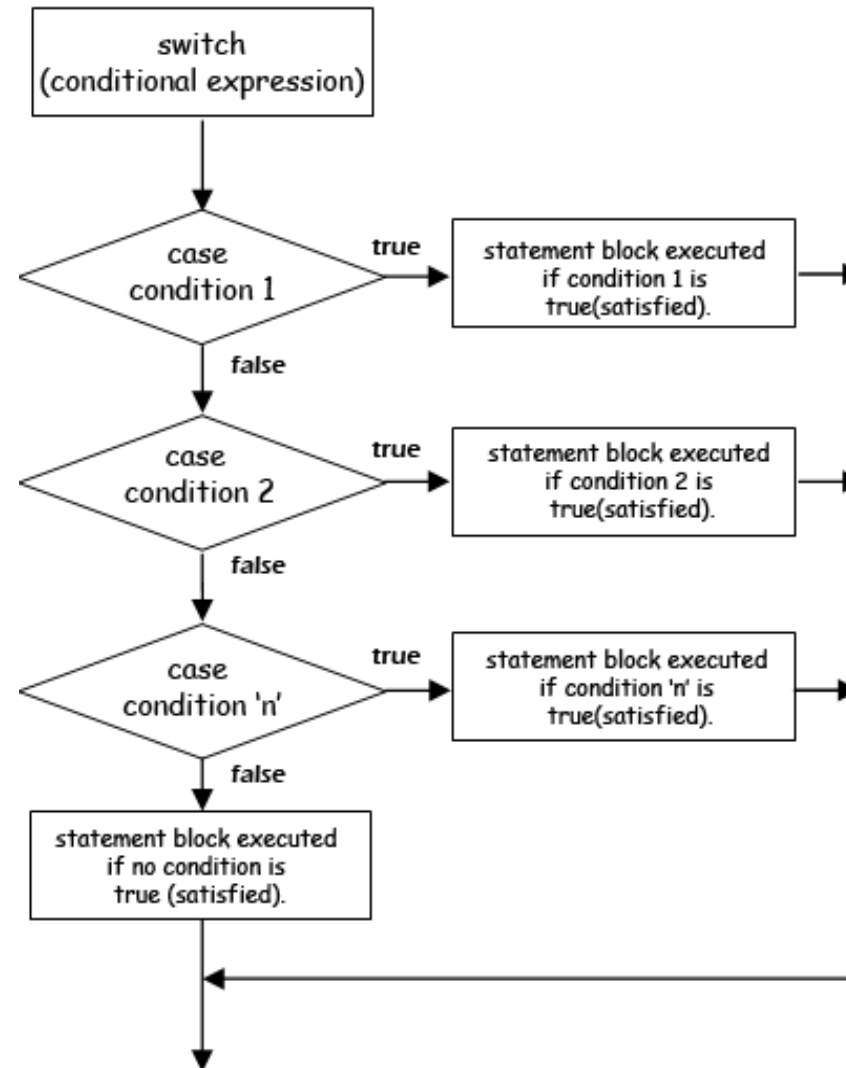
Los operadores lógicos siempre devuelven un valor TRUE o FALSE.

```
if ( esAlumno == true ) {  
    alert ("Hola alumno!");  
}
```

```
if (esAlumno == true ) {  
    alert ("Hola alumno!");  
} else {  
    alert ("Acceso sólo para alumnos");  
}
```


Evaluar si dos números son iguales,
diferentes, mayor o menor.

Switch



```
switch (expresión) {  
    case valor1:  
        Sentencias a ejecutar si expresión es valor1  
        break  
    case valor2:  
        Sentencias a ejecutar si expresión es valor2  
        break  
    case valor3:  
        Sentencias a ejecutar si expresión es valor3  
        break  
    default:  
        Sentencias a ejecutar si el valor no es  
ninguno de los anteriores  
}
```

Definir una variable numérica. Asignarle un valor entre 1 y 10

Mostrar a que grupo pertenece:

Grupo 1: del 1 al 3

Grupo 2: del 4 al 6

Grupo 3: del 7 al 10

Modifiquemos el ejercicio para que el número lo ingrese el usuario

```
a = prompt("ingrese un número");
```

Asegurémonos que se tome como entero!

```
a = parseInt(prompt("ingrese un  
número"));
```

Bucle FOR

```
for (var i=0; i<10; i++) {  
    console.log("El valor es: " + i);  
}
```

Bucle WHILE

```
var i = 0;
while (i < 10) {
    console.log("El valor es " + i);
    i++;
}
```


Bucle DO-WHILE

```
var i=0;  
do {  
    console.log("El valor es " + i);  
    i++;  
}  
while (i < 10);
```

Realizar la sumatoria de 0 a 10 y devolver el valor de la misma.

Funciones

Las funciones encapsulan una funcionalidad que se quiere utilizar.

Podemos llamar/invocar a esa función con un solo nombre y no tenemos que volver a escribir el código cada vez que la utilicemos

```
function name (parameter1, parameter2, parameter3) {  
    código a ser ejecutado  
}
```

Veamos

- ✓ ¿Cuál es el nombre de la función?
- ✓ ¿Dónde es invocada/llamada?
- ✓ ¿Cuáles son sus parámetros?
- ✓ ¿Cuál es el resultado de su invocación?
- ✓ ¿Dónde estará almacenado dicho resultado?

```
var x = myFunction(4, 3);  
  
function myFunction(a, b) {  
    return a * b;  
}
```

Veamos

- ✓ ¿Cuál es el nombre de la función?
- ✓ ¿Dónde es invocada/llamada?
- ✓ ¿Cuáles son sus parámetros?
- ✓ ¿Cuál es el resultado de su invocación?
- ✓ ¿Dónde estará almacenado dicho resultado?

```
var x = myFunction(4, 3);  
  
function myFunction(a, b) {  
    return a * b;  
}
```

```
// Definición de la función
function suma_y_muestra(n1, n2) {
    var resultado = n1 + n2;
    alert("El resultado es " + resultado);
}
```

```
// Declaración de las variables
var numero1 = 3;
var numero2 = 5;
```

```
// Llamada a la función
suma_y_muestra(numero1, numero2);
```

```
function nombreFuncion(parametro1) {  
    //Instruccion1  
    var nuevo = parametro1 + 5;  
    //InstruccionN  
    return nuevo;  
}
```

```
devuelto = nombreFuncion(4);
```

Funciones como constante

Otra forma de declarar funciones,
mas seguras en código


```
const saludo = function(mensaje) {  
    alert(mensaje);  
}
```


Funciones de flecha

Otra forma de declarar funciones

```
const saludo = (mensaje) => {  
    alert(mensaje);  
}
```

Ejercicios

1. Crear una función que recibas dos valores, y muestre la suma de ambos.
2. Crear una función que reciba dos cadenas de texto y retorne la concatenación de las mismas
3. Crear una función que reciba dos valores: cantidad y precio unitario. Si el total del ítem es mayor a 1000, aplicar un 10% de descuento. Devolver el valor total del ítem.
4. Crear una función que reciba un número y muestre tantos asteriscos como la cantidad de veces que se pasó como parámetro. EJ: 5  *****

Array

Colección de información que se encuentra dentro de una misma variable

```
// Array en JS  
var arreglo = [];  
var arreglo = [1, 2, 3, 4, 5, 6];
```

Array

Agregamos ítems a un array

```
arreglo.push(6,7,8);  
console.log(arreglo);
```

Array

Creamos un nuevo array en base a otro

```
var arreglo2 = [...arreglo, 5];
```

También podemos usar la función `map()` que permite utilizar funciones

```
var arreglo3 = arreglo2.map(function(valor) {  
    return valor * 2;  
});
```

Recorrer un array

```
var dato = [2, 6, 5, 1, 18, 44];  
  
// recorrido con for  
  
for (var i=0; i<dato.length; i++) {  
    console.log(dato[i]);  
}
```

Recorrer un array

```
var dato = ['n', 'q', 'x', 'c'];  
    // recorrido con for  
for (var i=0; i<dato.length; i++) {  
    document.write('La letra es:  
<b>' + dato[i] + '</b><br />');  
}
```

Generar un array con 10 números, recorrerlo e ir multiplicando todos los elementos, finalmente obtener el producto total.

Dentro de la página que se carga (document) puedo acceder a los elementos que tienen un ID, mediante **getElementById**

Ejemplo:

```
document.getElementById("num1").value
```

Como cualquier operación de asignación,
al poner a la izquierda del igual, voy a
setear el valor.

Si se lo pone a la derecha, se obtiene el
valor para poder usarlo.

Desarrollemos un portero eléctrico.

Tendrá dos visores, de dos posiciones el piso y una posición para dpto.

Los pisos van del 00 al 42.

Los dptos, del 1 al 8.

El botón **llamar**, muestra el mensaje de abajo. El botón **borrar** limpia los visores y el mensaje de abajo.

Si se hace referencia a un piso y/o dpto que no existe, mostrar el **error** en el visor de abajo

PISO		DPTO
1	2	3
4	5	6
7	8	9
0	Llamar	Borrar
Llamando al piso NN, dpto NN		

Desarrollemos una calculadora

Tendrá un solo display.

Botones de números del 0 al 9

Operaciones: + - * /

El boton = muestra el resultado en el display

Desarrollemos un teclado en pantalla

Cada línea del teclado debe hacerse en un array.

Al presionar cada tecla (botón) deberá mostrarse en el display.

La muestra estará centralizada en una sola función.

Debe existir un botón para borrar el display

Agregado:

- Botón Backspace...

Objetos literales

Los objetos literales son como variables que trabajan con pares de valores

```
// Objetos literales
const persona = {
    nombre: 'Tony',
    apellido: 'Stark',
    edad: 45
}
```

Desestructuración de objetos

Permite crear variables desde las propiedades de un objeto

```
// Objetos literales
const persona = {
  nombre: 'Tony',
  apellido: 'Stark',
  edad: 45,
  direccion: {
    ciudad: 'New York',
    codpos: '1234',
    calle: "Fifth Avenue",
    numero: 500
  }
};

const {nombre, apellido} = persona;

console.log(nombre, apellido);
```

Desestructuración de Array

Permite crear variables desde los valores un array

```
// Desestructuracion de array
const avengers = ['Ironman', 'Spiderman', 'Bruja Escarlata'];

const [p1, ,p3] = avengers;

console.log(p1, p3);
```