

## VI. JAVASCRIPT

Es un lenguaje de programación que te permite crear contenido nuevo y dinámico, controlar archivos de multimedia, crear imágenes animadas, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario, entre otros [4] [5].

JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. Es decir, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios [4].

Es muy empleado en la Web para realizar diferentes aplicaciones para ello es importante conocer previamente conceptos generales de HTML y CSS.

JavaScript es un lenguaje levemente tipado, que se presta bien para aprender a programar, y es relativamente sencillo. [6]

### A. ¿CÓMO INCLUIR JAVASCRIPT EN UN PROYECTO?

Un archivo JavaScript se encierra entre etiquetas `<script>`, tiene la extensión `.js` y se enlaza en un proyecto HTML considerando 3 posibilidades:

#### 1. Incluir JavaScript dentro del mismo documento:

El código JavaScript se incluye en cualquier parte del documento. Pero se recomienda que se lo haga dentro de `<head>`

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Primer Proyecto con Javascript</title>
  <script type="text/javascript">
    alert("Un mensaje de prueba");
  </script>
</head>
```

El atributo **type** esta estandarizado y para el caso de JavaScript, el valor correcto es `text/javascript`.

**Desventajas:** Si se quiere hacer una modificación en el bloque de código, es necesario modificar todas las páginas que incluyen ese mismo bloque de código JavaScript.

#### 2. Definir JavaScript en un archivo externo

Las instrucciones JavaScript se pueden incluir en un archivo externo de tipo JavaScript que los documentos HTML mediante la etiqueta `<script>`.

```
<head>
  <meta charset="UTF-8" />
  <title> EJERCICIO JAVASCRIPT </title>
  <script type="text/javascript" src="js/funciones.js"></script>
</head>
```



El atributo **src** nos indicara la ruta de nuestro archivo JavaScript que para el caso anterior se denomina funciones.js y se ubica dentro del directorio js.

**Ventajas:** Enlazar un archivo JavaScript externo permite reutilizar el mismo código JavaScript en todas las páginas del sitio web y que cualquier modificación realizada en el archivo se ve reflejada inmediatamente en todas las páginas HTML que lo enlazan

Cada etiqueta <script> solamente puede enlazar un único archivo, pero en una misma página se pueden incluir tantas etiquetas <script> como sean necesarias.

### 3. Incluir JavaScript en los elementos HTML

Consiste en incluir trozos de JavaScript dentro del código HTML de la página.  
Por ejemplo:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1">
6   <title>Primer Proyecto con Javascript</title>
7 </head>
8 <body>
9   <p onclick="alert('Un mensaje de prueba')">Un párrafo de texto.</p>
10 </body>
11 </html>
```

**Desventajas:** Ensucia el código y complica las tareas de mantenimiento.

## B. SINTAXIS DE JAVASCRIPT

**Sintaxis:** se define como el conjunto de reglas que deben seguirse al escribir el código fuente de los programas para considerarse como correctos para ese lenguaje de programación [4].

La sintaxis de JavaScript es muy similar a la de otros lenguajes de programación. A continuación, se muestran Las normas básicas que definen la sintaxis de JavaScript [4]:

- No se tienen en cuenta los espacios en blanco y las nuevas líneas
- Se distinguen las mayúsculas y minúsculas
- No se define el tipo de las variables
- No es necesario terminar cada sentencia con el carácter de punto y coma. Pero por buenas prácticas se recomienda hacerlo.
- Se pueden incluir comentarios de una o múltiples líneas.

Considerando la forma 2 para incluir a JavaScript desde un archivo externo se considera el siguiente ejemplo.

```
index.html x funciones.js
// Comentarios de una sola linea

a = 1
b = 2
A = 10
B = 30
c = a + b
C = A + B

alert("El resultado de a+b es = " + c);
alert("El resultado de A+B es = " + C);

/*
Comentarios de Multiples Lineas
Suma de Dos Números
*/
```

Comentarios de una sola linea

Diferencia entre mayúsculas y minúsculas

No es obligatorio usarlas

Comentarios

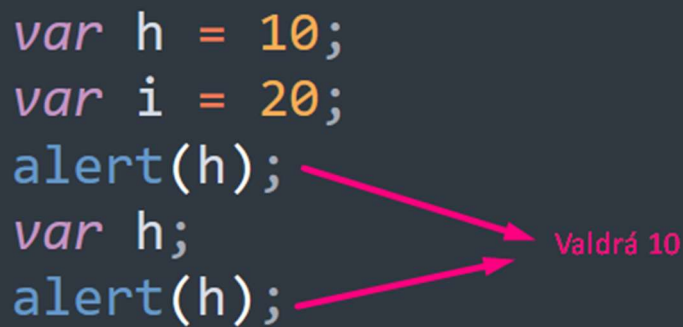
Comentarios de Multiples Lineas

## C. VARIABLES

Una variable es un elemento que se emplea para almacenar y hacer referencia a otro valor. Las variables en JavaScript se crean mediante la palabra reservada **var**, **let** o **const**.

### 1. Uso de var

La palabra clave **var** se usaba en todo el código JavaScript desde 1995 hasta 2015 [7]. Si vuelve a declarar una variable de JavaScript declarada con **var**, no perderá su valor.



```
var h = 10;
var i = 20;
alert(h);
var h;
alert(h);
```

Valdrá 10

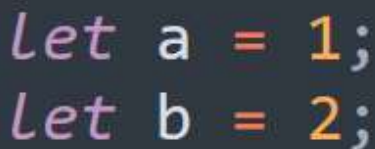
El diagrama muestra un código JavaScript con cinco líneas. Las primeras dos líneas declaran y asignan valores: `var h = 10;` y `var i = 20;`. La tercera línea llama a `alert(h);`. La cuarta línea redeclara la variable `h` sin asignar un nuevo valor: `var h;`. La quinta línea llama nuevamente a `alert(h);`. Dos flechas rojas indican que ambas llamadas a `alert(h);` muestran el valor 10, ya que la redeclaración con `var` no cambia el valor almacenado.

Actualmente se utiliza la expresión **let** o **const**.

### 2. Uso de let

Si el valor de la variable va a cambiar se debe usar **let**.

La palabra reservada **let** se debe indicar al definir por primera vez la variable, lo que se denomina declarar una variable.



```
let a = 1;
let b = 2;
```

El diagrama muestra dos líneas de código JavaScript: `let a = 1;` y `let b = 2;`.

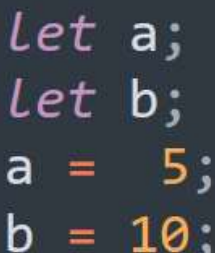
Cuando se utilizan las variables en el resto de instrucciones del script, solamente es necesario indicar su nombre.



```
let r = a + b;
```

El diagrama muestra una línea de código JavaScript: `let r = a + b;`.

En JavaScript no es obligatorio inicializar las variables, ya que se pueden declarar por una parte y asignarles un valor posteriormente.



```
let a;
let b;
a = 5;
b = 10;
```

El diagrama muestra cuatro líneas de código JavaScript: `let a;`, `let b;`, `a = 5;` y `b = 10;`.

**Nota:** Las variables definidas con **let** no se pueden volver a declarar. Las variables definidas con **let** deben declararse antes de su uso [7].

### 3. Uso de Const

Si desea aplicar una regla general se debe crear siempre las variables con **const**.

```
const price1 = 5000;
const price2 = 1000;
let total = price1 + price2;
```

En el ejemplo anterior se detalla que las variables price1 y price2 se declaran con la palabra clave **const**. Esto implica que sus valores son constantes y no se pueden cambiar.

Mientras que la variable total se declara con la palabra clave **let** y este es un valor que si puede cambiar.

**Nota: Las variables definidas con const no se pueden volver a declarar. Las variables definidas con const no se pueden reasignar [7].**

Se recomienda usar **const** cuando se declara:

- Una nueva matriz
- Un nuevo objeto
- Una nueva función
- Una nueva expresión regular

#### Reglas al nombrar una variable

El nombre de una variable también se conoce como identificador y debe cumplir las siguientes normas [4] [7]:

1. El identificador de las variables es único.
2. El identificador puede ser corto o largo.
3. Los nombres pueden contener letras, dígitos, guion bajo (\_) y signos de dólar (\$).
4. El primer carácter no puede ser un número.
5. Los nombres distinguen entre mayúsculas y minúsculas.
6. Las palabras reservadas (como las palabras clave de JavaScript) no se pueden usar como nombres

```
// Las siguientes variables tienen nombres correctos:
let $numero1;
let _$letra;
let $$$otroNumero;
let $_a__$4;

//Sin embargo, las siguientes variables tienen identificadores
//incorrectos:
let 1numero; // Empieza por un número
let numero;1_123; // Contiene un carácter ";"
```

## Declarar múltiples variables

JavaScript permite declarar muchas variables en una declaración. Para ello se debe usar la declaración con **let** y separar las variables con comas.

```
let person = "John Doe", carName = "Volvo", price = 200;
```

## Tipos de variables

Tipo	Descripción
Numéricas	Se utilizan para almacenar valores numéricos enteros o decimales. En este caso, el valor se asigna indicando directamente el número entero o decimal. Los números decimales utilizan el carácter punto en vez de coma. <pre>let number = 10, number2 = 20;</pre>
Cadena	Se utilizan para almacenar caracteres, palabras y/o frases de texto. Para asignar el valor a la variable, se encierra el valor entre comillas dobles o simples. <pre>let name = "Mi nombre es Laura";</pre>
Booleano	Almacena un tipo especial de valor que solamente puede tomar dos valores: true (verdadero) o false (falso). No se puede utilizar para almacenar números y tampoco permite guardar cadenas de texto. <pre>let flat = true; let flat2 = false;</pre>

Elaboración propia basado en [4]

Nota: Para mostrar el valor de una variable se puede hacer usando la función alert()

```
let flat = true;  
let name = "Mi nombre es Laura";  
  
alert(flat); // Se visualiza en una ventana los datos de la variable flat  
alert(name); // Se visualiza en una ventana los datos de la variable name
```

Resultado en el navegador:

Esta página dice

true

Aceptar

Esta página dice

Mi nombre es Laura

Aceptar

## D. OPERADORES

Operador	Nombre	Utilidad
=	Asignación	Este operador se utiliza para guardar un valor específico en una variable
++	Incremento	Es válido para las variables numéricas y se utilizan para incrementar en una unidad el valor de una variable
--	Decremento	Es válido para las variables numéricas y se utilizan para decrementar en una unidad el valor de una variable
!variable	Negación	Empleado en operadores booleanos se utiliza para obtener el valor contrario al valor de la variable.
&&	AND	La operación lógica AND obtiene su resultado combinando dos valores booleanos. Su resultado solamente es true si los dos operandos son true.
	OR	La operación lógica OR combina dos valores booleanos. Su resultado es true si alguno de los dos operandos es true.
+	Suma	Operadores matemáticos básicos
-	Resta	
*	Multiplicación	
/	División	
%	Módulo	Calcula el resto de la división entera de dos números.
>	Mayor	Operadores relacionales
<	Menor	
>=	Mayor igual	
<=	Menor igual	
==	Igual que	
!=	Diferente	



## E. ARREGLOS

Un array es una colección de variables, que pueden ser todas del mismo tipo o cada una de un tipo diferente.

### Sintaxis:

```
const array_name = [elemento1, elemento2, ...];
```

Es una práctica común declarar arreglos con la palabra clave `const`.

Opción 1:

```
const animales = ["perro", "gato", "pajaro"];
```

Opción 2:

```
const autos = new Array("Saab", "Volvo", "BMW");
```

Así mismo se puede crear un array y luego proporcionar los elementos:

Ejemplo:

```
const vestuario = [];  
vestuario[0] = 'Gorra';  
vestuario[1] = 'Saco';  
vestuario[2] = 'Pantalón';
```

### Acceso a los elementos del array

Para acceder a un elemento de un arreglo se puede lograr haciendo referencia al número de índice:

```
const cars = ["Ford", "Mazda", "Chevrolet"];  
alert(cars[0]);
```

O bien:

```
const cars = ["Ford", "Mazda", "Chevrolet"];  
let car = cars[0];  
alert(car);
```



## Cambio de un elemento de un array

```
const animales = ["perro", "gato", "pajaro"];
animales[2] = "loro";
alert(animales[2]);
```

Esta página dice

loro

Aceptar

## Acceda a la matriz completa

Con JavaScript, se puede acceder a la matriz completa haciendo referencia al nombre de la matriz:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Primer Proyecto con Javascript</title>
</head>
<body>
  <p id="text"></p>

  <script>
    const cars = ["Ford", "Mazda", "Chevrolet"];
    document.getElementById("text").innerHTML = cars;
  </script>
</body>
</html>
```

Resultado:

Ford,Mazda,Chevrolet

## Propiedades de los Array

Propiedad	Función
array.length	Obtiene la longitud del array
array.sort()	Ordena el arreglo
array.push(Elemento)	Añadir un elemento al final de un Array
array.pop()	Eliminar el último elemento de un Array
array.unshift(Elemento)	Añadir un elemento al principio de un Array
array.shift()	Eliminar el primer elemento del array.
array.indexOf("Ford")	Encontrar el índice de un elemento del Array

## ESTRUCTURAS DE CONTROL

### 1. Estructura IF

Se emplea para tomar decisiones en función de una condición [4]. Su definición formal es:

Opción 1:

```
Let Mensaje = true;  
if(Mensaje) {  
    alert("Hola Mundo");  
}
```

Opción 2:

```
Let Mensaje = true;  
if(Mensaje == true) {  
    alert("Hola Mundo");  
}
```

### 2. Estructura If – Else

Normalmente las condiciones suelen ser del tipo "si se cumple esta condición, hazlo; si no se cumple, haz esto otro" [4].

```
Let edad = 18;  
if(edad >= 18) {  
    alert("Puedes votar");  
}  
else {  
    alert("Todavía eres menor de edad y no puedes votar");  
}
```

### 3. Estructura While

El ciclo While recorre un bloque de código siempre que una condición específica sea verdadera.

```
let texto = "";
let j = 0;
while (j < 10) {
  texto += "<br>The number is " + j;
  j++;
}

document.getElementById("texto").innerHTML = texto;
```

### 4. Estructura For

Se emplea cuando se desea ejecutar de forma repetitiva una instrucción.

```
mensaje = "Hola Mundo";
for(var i = 0; i < 5; i++) {
  alert(mensaje);
}
```

### 5. Estructura For ... in

Si se quieren recorrer todos los elementos que forman un array, la estructura for...in es la forma más eficiente de hacerlo [4].

```
const cars = ["Ford", "Mazda", "Chevrolet"];
for(i in cars) {
  alert(cars[i]);
}
```

## F. PROPIEDADES A TRABAJAR DENTRO DE JAVASCRIPT

A continuación, se detallará algunas propiedades de JavaScript que se manejará dentro de la clase.

Propiedad	Descripción	Ejemplo
<code>document.getElementById()</code>	Encuentra un elemento HTML cuyo id sea el que se indique.	<code>document.getElementById("text")</code>
<code>.innerHTML</code>	Método de acceso rápido al contenido completo de un contenedor en HTML. Permite recuperar el contenido actual de un contenedor o insertar nuevo contenido en ese contenedor.	<code>document.getElementById("text").innerHTML = "MI NUEVO TEXTO"</code>
<code>.value</code>	Obtiene el valor de la entrada de texto.	<code>document.getElementById("text").value</code>
Propiedad	Descripción	Ejemplo
<code>console.log()</code>	Visualiza el mensaje en la consola del navegador.	<code>console.log ("MENSAJE EN CONSOLA");</code>
<code>document.write()</code>	Si usa <code>document.write()</code> después de cargar un documento HTML, eliminará todo el HTML existente	<code>document.write("MENSAJE ELIMINADO");</code>
<code>alert()</code>	Permite visualizar una ventana de alerta para mostrar datos.	<code>alert("HOLA SOY UNA VENTANA");</code>
<code>prompt(text, defaultText)</code>	Se usa para mostrar un cuadro de diálogo con un mensaje que solicita al usuario que ingrese algún texto o información.	<code>13 = prompt("Ingrese número");</code> <code>alert(13);</code>

### G. PALABRAS RESERVADAS DE JAVASCRIPT

Al igual que otros lenguajes de programación JavaScript trabaja sus palabras reservadas como parte de la sintaxis del mismo y la cual permitirá realizar diferentes funcionalidades.

Keyword	Description
break	Terminates a switch or a loop
continue	Jumps out of a loop and starts at the top
debugger	Stops the execution of JavaScript, and calls (if available) the debugging function
do ... while	Executes a block of statements, and repeats the block, while a condition is true
for	Marks a block of statements to be executed, as long as a condition is true
function	Declares a function
if ... else	Marks a block of statements to be executed, depending on a condition
return	Exits a function
switch	Marks a block of statements to be executed, depending on different cases
try ... catch	Implements error handling to a block of statements
var	Declares a variable

Fuente [7]

## Bibliografía

- [1] Codigofacilito.com, «Qué Es Git,» 2015. [En línea]. Available: <https://codigofacilito.com/articulos/que-es-git>.
- [2] Git.com, «Inicio - Sobre el Control de Versiones - Fundamentos de Git,» [En línea]. Available: <https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Fundamentos-de-Git>.
- [3] Hostinger Tutoriales, «¿Qué es GitHub y para qué se utiliza?,» 2019. [En línea]. Available: <https://www.hostinger.co/tutoriales/que-es-github/>.
- [4] J. Eguíluz Pérez, Introducción a Javascript, 2009.
- [5] developer.mozilla.org, «What is JavaScript?,» [En línea]. Available: [https://developer.mozilla.org/es/docs/Learn/JavaScript/First\\_steps/Qu%C3%A9\\_es\\_JavaScript](https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qu%C3%A9_es_JavaScript).
- [6] <https://desarrolloweb.com/>, «JavaScript,» [En línea]. Available: <https://desarrolloweb.com/home/javascript>.
- [7] W3SCHOOL, «Javascript,» [En línea]. Available: [https://www.w3schools.com/js/js\\_statements.asp](https://www.w3schools.com/js/js_statements.asp).