Programación con JavaScript I

# Fundamentos de JavaScript

# Introducción

El lenguaje de programación JavaScript es el más utilizado hoy en día para entornos web. Junto con HTML y CSS, le da vida a la gran mayoría de los sitios web que utilizamos. Sin embargo, JavaScript ya no es exclusivo de la web, gracias a proyectos como **NodeJS**, **Electron** o **React Native** han llevado al ámbito de servidores algunos programas de escritorio y de las aplicaciones móviles. Además de esto, cada día nacen nuevos proyectos para expandir las posibilidades de JavaScript.

JavaScript se puede definir como un lenguaje de tipo interpretado, que está basado en el estándar ECMAScript, débilmente tipado y dinámico. También se define como orientado a objetos. Según la encuesta realizada todos los años por Stack Overflow (Merino, 2022), JavaScript ha sido el lenguaje de programación más utilizado durante los últimos 10 años.


Jeff Atwood, afamado autor del blog Coding Horror y cofundador de Stack Overflow, menciona que el éxito de este lenguaje se debe a cuatro factores (LOS TIEMPOS CAMBIAN, 2013):

- Ubicuidad
- Increíbles mejoras de rendimiento
- Código abierto (*Open Source*)
- Experiencia de usuario (*Web User Experience*)

En este tema aprenderás los ambientes en que se desarrolla, se administra el código y dónde se ejecuta este lenguaje, así como sus principales características.

# Explicación

De acuerdo con Flanagan (2020), JavaScript es un lenguaje de programación de la web e indica que la mayoría de los sitios web usan este lenguaje. Esto es posible porque es un lenguaje de guion (*scripting*) del lado del cliente.

Su enfoque primordial es ayudar a los desarrolladores a permitir la interacción del usuario con la página web, así como con el navegador mismo. Además, permite una comunicación asíncrona y puede actualizar partes de una página o todo el contenido si así se desea.

Aunque JavaScript se basa un poco en el lenguaje de programación Java, en su sintaxis y metodología es completamente incorrecto considerar que sea una versión ligera de Java. El nombre deriva por cuestiones de mercadeo, ya que su nombre original era “Live Script”, pero al momento de su lanzamiento, el lenguaje Java estaba en auge, por lo que su creador, Netscape, cambió su nombre de último momento junto con un release para que fuera más “parecido” a Java.

Entrando más a detalle, JavaScript no puede considerarse del todo como un lenguaje de programación como tal, en lugar de eso, es considerado como un lenguaje de guion o scripting, dado que utiliza al navegador web para realizar el trabajo sucio. Por ejemplo, si se requiere que una imagen sea reemplazada por otra cada determinado tiempo, JavaScript le pedirá al navegador que lo haga solo escribiendo algunas líneas de código relativamente simples. Por este tipo de acciones es que JavaScript es un lenguaje un tanto fácil para comenzar. No obstante, no quiere decir que en JavaScript no se puedan programar códigos complejos.

Un punto que ha favorecido el crecimiento de JavaScript es que todos los navegadores web modernos lo soportan, sin embargo, no todos son compatibles con todo el funcionamiento que se puede implementar. Esta es una de las razones por las que se dice que es fácil aprender JavaScript básico, pero esta afirmación no es tan válida cuando se tienen que desarrollar códigos más complejos, ya que pueden aparecer errores inesperados dependiendo del navegador web y su versión.

Para poder programar en JavaScript necesitarás un **entorno de desarrollo**, es decir, un software que te facilite la escritura y edición del código. Puedes utilizar un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés). Aunque hay varios **niveles** y **características**, dos de las características más importantes que debes tener en cuenta son el resaltado de sintaxis y autocompletado.

# Explicación

Algunos editores gratuitos son los siguientes:

- Aptana - http://www.aptana.com
- Eclipse - https://www.eclipse.org
- Visual Studio Code - https://code.visualstudio.com/docs/languages/javascript

Ejemplos de editores de pago:

- Intellij Webstorm - https://www.jetbrains.com/webstorm
- Komodo IDE - http://komodoide.com
- MicroSoft Visual Studio - http://www.visualstudio.com

Ejemplos de editores simples:

- JEdit: http://www.jedit.org/
- Komodo Edit: http://komodoide.com/komodo-edit/
- Notepad++: http://notepad-plus-plus.org/

Todo el código que desarrolles se ejecuta en navegadores web, por lo tanto, necesitas tener al menos uno instalado en tu equipo, sin embargo, es una buena práctica poder probarlo en varios de los navegadores más populares para asegurar un buen funcionamiento. Algunos de los navegadores más populares son Firefox, Chrome, Safari, Opera, Edge e Internet Explorer.

Además del ambiente de desarrollo, se sugiere que utilices un software controlador de versiones (VCS, por sus siglas en inglés). Esta herramienta te permitirá colaborar con otros desarrolladores en un proyecto sin que exista la posibilidad de sobrescribir el trabajo de otros. Otra ventaja es que puedes volver a una versión anterior si existe un problema más tarde.

Git es un ejemplo de este tipo de herramientas y es el más popular actualmente. Por otra parte, GitHub es considerado como uno de los mejores sitios web que proporciona alojamiento para repositorios de control de versiones, además de que ofrece varias herramientas para trabajar con ellos.

# Explicación

Una vez que tengas estas herramientas instaladas y configuradas, es momento de poner manos a la obra y de activar el F12. Dicha tecla te permite tener acceso a las herramientas del desarrollador, mostrando el código HTML, CSS y JavaScript del que disponga la página que se esté visualizando en el navegador web.

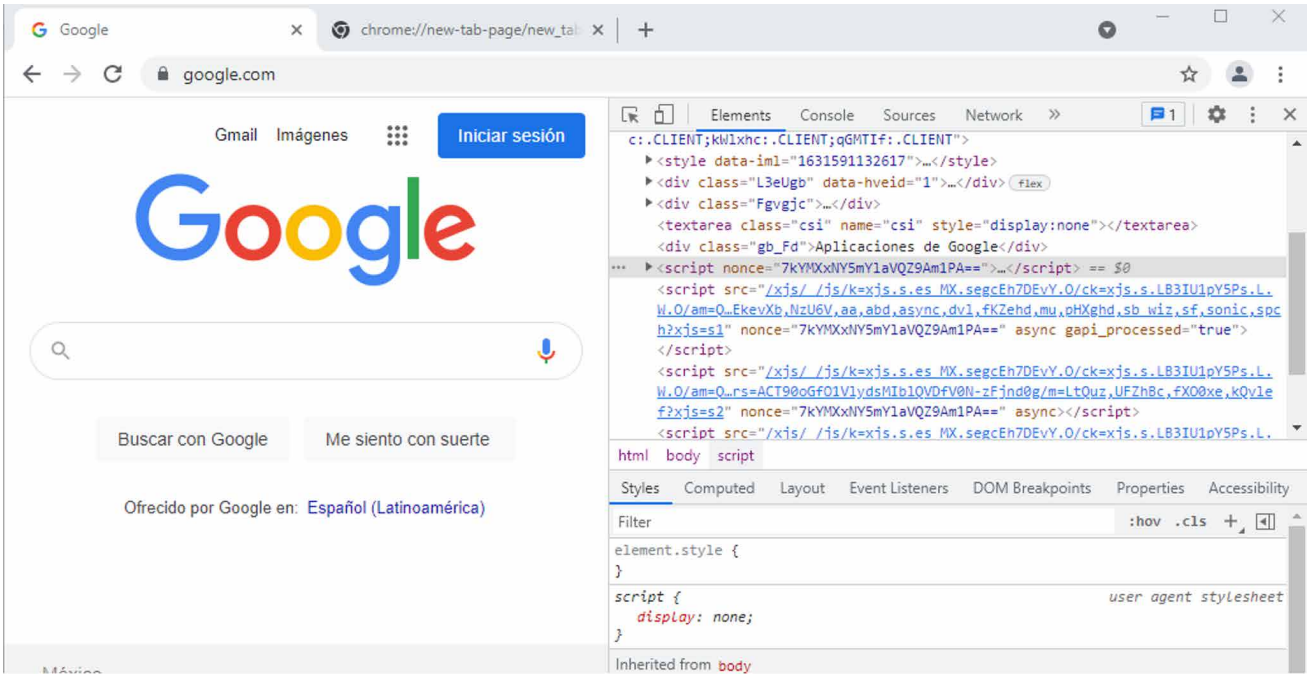



Figura 1. Ejemplo de consola en página.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

## Uso de JavaScript

Para poder entender este código, debemos saber que existen dos maneras de desarrollar JavaScript, las cuales son integrado e independiente.

La primera se realiza utilizando la etiqueta <SCRIPT> para indicar al navegador que el texto que continúe a esta etiqueta será código JavaScript.

```
<SCRIPT language = "JavaScript">
// Aquí va código JavaScript
</SCRIPT>
```

Tabla 1. Uso de JavaScript en código HTML.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

# Explicación

En el documento HTML se coloca este código debajo de la etiqueta <html> y arriba de la etiqueta <body>, sin embargo, también se puede poner dentro del cuerpo de la página.

```
<html>
<head>
<script type="text/javascript">
...
</script>
</head>
<body>
<script type="text/javascript">
...
</script>
</body>
</html>
```

Tabla 2. Ubicación del código JavaScript en código HTML.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

La otra manera de realizarlo es con código fuera del HTML, es decir, utilizando un código externo, lo que significa que se tendría que crear un nuevo archivo e invocarlo desde el archivo HTML.

```
...
<script src="index.js"></script>
</head>
```

Tabla 3. Invocación de JavaScript desde código externo.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Y el cuerpo del archivo .js deberá contener el código en JavaScript.

```
function saludo() {
  alert("Hola Mundo. Hola Javascript");
}
```

Tabla 4. Código JavaScript.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Y entonces podremos utilizar las funcionalidades del código JavaScript desde el HTML.

```
<body onload="saludo();">
```

Tabla 5. Uso de JavaScript desde HTML.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Otra manera para visualizar el código JavaScript es mediante la consola del navegador web.

# Explicación

**Características de JavaScript**

La sintaxis más destacable del lenguaje JavaScript es la siguiente:

- **Palabras reservadas:** son aquellas palabras que utiliza el lenguaje y que no pueden ser utilizadas para ningún otro propósito.
- **Tipos de datos:** a diferencia de otros lenguajes de programación, las variables no requieren definir un tipo de dato, pueden almacenar cualquier tipo sin problema.
- **Textos:** cuando se asigna un texto a una variable dentro de una función, este debe estar encerrado entre comillas (dobles o simples).
- **Declaración de variables:** las variables son declaradas anteponiendo la palabra reservada var. El nombre de una variable debe ir en minúsculas y tiene que ser posible identificar su contenido a través de su nombre. No se deben usar caracteres especiales.
- **Asignación de datos:** para asignar un dato, siempre se debe anteponer el signo igual.
- **Números y booleanos:** los números asignados a una variable o función deben estar escritos sin comillas, lo mismo sucede para los valores booleanos (True o False).
- **Final de una sentencia:** cada línea de código que finaliza debe concluir con punto y coma (;).

Las reglas para crear funciones son las siguientes:

- **Funciones personalizadas:** cuando se crean funciones en JS, se antepone la palabra reservada function, seguida del nombre.
- **Declaración de parámetros:** al crear un nombre de función se termina con paréntesis. Los parámetros se definen dentro de los paréntesis.
- **Uso de llaves:** cuando se crean funciones, todo el contenido debe ir encerrado entre llaves {...} que funcionan como limitadoras de contenido.
- **Funciones integradas:** son funciones parte del lenguaje y tienen definida la tarea y los parámetros que se reciben.

Para el trabajo con objetos existen las siguientes definiciones:

- **Funciones integradas:** son parte del lenguaje y tienen definida la tarea y los parámetros que reciben.
- **Trabajo con objetos:** los objetos se declaran tan fácilmente como cuando se crea una simple variable.
- **Acceder a propiedades:** el consumo de las propiedades y métodos de un objeto es muy simple: propiedad.objeto

# Explicación

JavaScript es un lenguaje sensitivo a las mayúsculas, esto significa que las palabras clave del lenguaje, variables, nombre de funciones y otros identificadores siempre deben escribirse de la misma manera. Por ejemplo, se deberá escribir **while** no **While** o **WHILE**. Lo mismo ocurre con **online**, **Online**, **OnLine** u **ONLINE**, son distintos los nombres de variables.

En JS se pueden utilizar espacios para indentar o pueden utilizarse tabuladores. Adicionalmente, reconoce los caracteres de control ASCII, así como nuevas líneas y retornos de carro/saltos de línea como determinadores.

Los comentarios se pueden realizar de dos formas diferentes:

- Cualquier texto entre // y el final de la línea es ignorado por Javascript.
- Cualquier texto entre los caracteres /* y */ también serán tratados como comentarios. Estos comentarios pueden abarcar varias líneas, pero no pueden estar anidados.

```
// Este es un comentario simple.
/* este también es un comentario */ // y aquí está otro comentario.
/*
* Este es un comentario multi línea. El caracter * extra al inicio de
* cada línea no es requerido como parte de la sintaxis, pero se ve bien!
*/
```

Tabla 6. Comentario en JavaScript.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Las literales son valores que aparecen directamente en un programa. Algunos ejemplos son los siguientes:

```
12 // Número doce
1.2 // Numero uno punto dos
"Hola Mundo" // Una cadena de texto
'Hi' // Otra cadena
true // Un valor booleano
false // Otro valor booleano
null // Ausencia de un objeto
```

Tabla 7. Literales en JavaScript.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

# Explicación

Los identificadores son simples nombres. En JavaScript, los identificadores son usados para nombrar constantes, variables, propiedades, funciones y clases. Un identificador puede comenzar con una letra, un guion bajo o un signo de pesos. Los caracteres subsecuentes pueden ser letras, dígitos, guiones bajos o signos de pesos. Los siguientes son identificadores válidos:

```
i
mi_nombre_de_variable
v15
_muestra
$cadena
```

Tabla 8. Identificadores en JavaScript.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Como lo explica Arbeláez (2021), muchos lenguajes de programación cuentan con un sublenguaje de tipos para declarar variables, es decir, este sublenguaje instruye al compilador acerca de qué tipo de valores puede tener cada variable.

JavaScrip, al ser débilmente tipado, no controla los tipos de las variables que se declaran y utilizan, de este modo, es posible usar **variables** de cualquier tipo en un mismo escenario.

Las variables, al igual que en todos los lenguajes de programación, son espacios de memoria donde guardamos temporalmente datos a los que podemos acceder en cualquier momento de la ejecución del programa. Es importante, en este momento, mencionar las constantes, que también son espacios de memoria donde se guardan datos, pero la diferencia es que este valor no podrá cambiar durante todo el programa. Para definir una constante, se utiliza la palabra reservada **const**. Por ejemplo:

```
const web = "www.google.com";
const pi = 3.1416;
```

Tabla 9a. Definición de constantes en JavaScript.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Pueden tener varios tipos y clases al igual que las variables. Para definir una variable se utiliza la palabra reservada **var** o **let**. Por ejemplo:

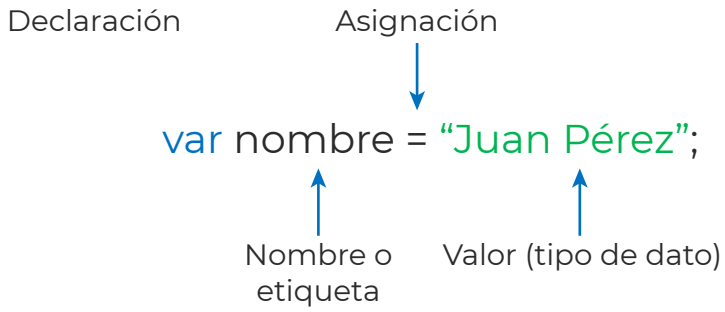
```
var miDato;
let otroDato;
```

Tabla 9b. Definición de variables en JavaScript.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

# Explicación

Aunque la palabra reservada **let** se utiliza más en el lenguaje TypeScript, cada día aumenta más su uso en JavaScript para indicar que la **variable es local**, es decir, solo está disponible en una parte del código. Por otro lado, el uso de la palabra reservada **var** es para declarar **variables globales** que pueden ser referidas en cualquier parte del código.

De aquí se puede observar que las variables se dividen en cuatro partes:




Existen al menos dos maneras para declarar una variable: se puede asignar un valor inicial en la misma línea que la declaramos. Por ejemplo:

```
var dato = 0;
```

Tabla 10. Inicialización de variables en JavaScript.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

O podemos declarar la variable y posteriormente, en alguna otra línea, asignarle un valor:

```
var dato;
dato = 0;
```

Tabla 11. Asignación de valor a variable en JavaScript.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Las variables deberán apegarse a una de las más conocidas convenciones de nominación:

| Nombre | Descripción | Ejemplo |
|---|---|---|
| CamelCase | La primera letra es minúscula y cada palabra añadida comienza con una letra mayúscula. | var miValorEjemplo; |
| Notación Pascal | La primera letra es mayúscula y cada palabra añadida comienza con letra mayúscula. | var miValorPrueba = 0 |
| Notación Húngara | Se agrega una letra minúscula al inicio de una variable con notación Pascal, esta letra inicial indica el tipo de variable: i para entero, s para string, etcétera. | var iMiValorPrueba = 0<br>var sOtroValor = 'Hola'; |

Tabla 12. Uso de notaciones en JavaScript.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

# Explicación

Hasta ahora hemos hablado de tipo de datos sin definir cuáles son. JavaScript tiene cuatro **tipos de datos** primitivos para almacenar en variables:

**1. Numérico:** este tipo de dato primitivo permite almacenar valores numéricos. Con estos valores se pueden realizar operaciones (aritméticas y lógicas), tales como contar, hacer cálculos o comparaciones, entre otros. Estas variables pueden representar números que van desde −9,007,199,254,740,992 (−253) hasta 9,007,199,254,740,992 (253). Ejemplos:

```
var numeroEntero = 1;
var numeroDecimal = 1.66;
```

Tabla 13. Definición de variable de tipo numérico.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

**2. Booleano:** este tipo de dato almacena un valor que indica falso o verdadero, prendido o apagado, 0 o 1, sí o no. Los valores booleanos se utilizan para indicar estados. Ejemplos:

```
var si = true;
var no = false;
```

Tabla 14. Definición de variable de tipo booleano.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

**3. Cadena (*string*):** este tipo de dato primario almacena caracteres o palabras. Se determina entre comillas dobles o simples (comillas o apóstrofos). Con este tipo de dato se pueden hacer concatenaciones, subcadenas, obtener longitud de cadena, remplazar, poner en mayúsculas o minúsculas, etc. Ejemplo:

```
var cadena = “Esto es una variable que de tipo string”;
var otraCadena = ‘Esto es otra cadena de texto’;
<button onclick=”alert(‘Gracias’)”>Oprime aquí</button>
```

Tabla 15. Definición de variable de tipo cadena (string).
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

**4. Indefinido:** este tipo de dato se utiliza para indicar la ausencia de valor o que no ha sido definido aún. Ejemplo:

```
var dato; // el valor de la variable dato es indefinido
var dato = undefined;
```

Tabla 16. Definición de variables de tipo indefinido.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

# Explicación

En el pasado, algunos autores han afirmado que otro tipo de almacenamiento de datos que tiene JavaScript son los objetos. Como en JS todo es un objeto, inclusive las funciones, todo hereda de la clase **Object**. Entonces, los objetos se pueden definir como una estructura donde se agregan valores.

Por otro lado, Flanagan (2020) indica que el tipo objeto tiene una diferencia muy grande con los tipos de datos primitivos: los tipos de datos primitivos son inmutables, mientras que los objetos pueden mutar. Los primitivos son comparados por valor, es decir, dos valores son iguales solamente si tienen el mismo valor. Esto pareciera obvio, pero no sucede precisamente así para los tipos string. Por ejemplo:

```
var s = "hello"; // Inicia como un texto en minúsculas
s.toUpperCase(); // Regresa "HELLO", pero no se altera s
s // => "hello": la cadena original no ha cambiado
```

Tabla 17. Valores de tipos de datos primitivos.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

En este ejemplo se muestra qué sucede con los objetos:

```
var o = { x: 1 }; // inicia con un objeto
o.x = 2; // El objeto muta cambiando el valor de la propiedad
o.y = 3; // Vuelve a mutar agregando una nueva propiedad
var a = [1,2,3]; // los arreglos son también mutables
a[0] = 0; // Se puede cambiar el valor de cualquiera de los elementos del arreglo
a[3] = 4; // y se pueden agregar valores al arreglo
```

Tabla 18. Valores en los objetos.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Los objetos no son comparados por valor, dos objetos no son iguales aun si tienen los mismos elementos y valores. De igual forma, dos arreglos distintos no son iguales si ellos tienen el mismo número de elementos en el mismo orden. Ejemplo:

```
let o = {x: 1}, p = {x: 1}; // Dos objetos con las mismas propiedades
o === p // => false: distintos objetos nunca son iguales
let a = [], b = []; // dos arreglos distintos y vacíos
a === b // => false: distintos arreglos/objetos nunca son iguales
```

Tabla 19. Comparación de valores en objetos.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

# Explicación

Los objetos, en ocasiones, son llamados **tipos de referencia** para distinguirlos de los tipos primitivos. Como Haverbeke (2018) explica, JavaScript hace todo lo posible por aceptar casi cualquier código que se le ingrese, incluso los programas que hacen cosas "raras", es decir, al ser muy flexible sobre el tipo de dato, de hecho, puede convertir o coercionar el tipo de dato según lo necesite. Algunos ejemplos son los siguientes:

```
10 + " objects" // => "10 objects": Convierte el número 10 a cadena
"7" * "4" // => 28: se realiza la coerción de las cadenas a números
let n = 1 - "x"; // n == NaN; la cadena "x" no se puede coersionar a número
n + " objects" // => "NaN objects": NaN lo convierte en la cadena "NaN"
```

Tabla 20. Conversión del tipo de dato.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Si bien es cierto que JavaScript por sí mismo puede hacer este trabajo, en algunos casos, es necesario convertir algún tipo de dato para utilizarlo de la manera más conveniente para el desarrollador. Para esto se deben utilizar algunas funciones dependiendo del tipo de dato, pero solo se puede realizar con tipos de datos primitivos.

Existen operadores que permiten formar expresiones para poder realizar estas operaciones con los tipos y objetos. Dichos operadores son los siguientes:

- Operadores aritméticos: permiten formar operaciones aritméticas como la suma, resta, operaciones con paréntesis, divisiones y multiplicaciones.

| Operador | Descripción | Ejemplo |
|---|---|---|
| + | Adición | var resultado = 9 + 3; // 12 |
| - | Substracción | var resultado = 9 - 3; // 6 |
| * | Multiplicación | var resultado = 9 * 3; // 27 |
| / | División | var resultado = 9 / 3; // 3 |
| % | Módulo | var resultado = 9 % 2; // 1 (Residuo de la división) |
| ++ | Incremento | var resultado = 9 ++ ; // 10 |
| -- | Decremento | var resultado = 9 -- ; // 9 (Residuo de la división) |
| ** | Exponenciación (Potencia) | var resultado = 2**3 ; // 8 |

# Explicación

Ejemplos:

```
var r = 1; // r=1
var s = ++r; // r=2, s=2
var t = s++ +r; //r=2, s=3, t=5
```

Tabla 21. Operaciones aritméticas.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

- Operadores booleanos: estos solo pueden tener dos tipos posibles: verdadero o falso. Sin embargo, con este tipo de operadores se pueden realizar negaciones, igualdades o comparaciones. Ejemplos:

```
!true=false // negación
5 > 3 // Comparación
true === true // identidad o igualdad
```

Tabla 22. Operaciones booleanas.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

- Operadores lógicos: son operadores que pueden devolver falso o verdadero dependiendo de las tablas de verdad. Estos tipos de operadores son muy utilizados para realizar comprobaciones. Se utilizan los operadores AND (&&) y OR (||). Por ejemplo:

```
0 && true //devuelve 0, porque el valor 0 es considerado un valor falso
var port = process.env.PORT || 5000;
```

Tabla 23. Operaciones lógicas.
Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos.

Algo importante a considerar en los diferentes tipos de operadores es su precedencia, la cual determina el orden en el que los operadores son utilizados para evaluar una expresión. Esta precedencia puede ser modificada por medio del uso de paréntesis. Por ejemplo:

```
console.log(3 + 4 * 5); // 3 + 20
// salida esperada: 23

console.log((3 + 4) * 5); // 7 * 5
// salida esperada: 35

console.log(4 * 3 ** 2); // 4 * 9
// salida esperada: 36, realiza primero la potencia 3**2= 9 y después multiplica 4

var a;
var b;

console.log(a = b = 5);
// salida esperada: 5
```

# Cierre

A lo largo de este tema pudiste aprender acerca del software que vas a necesitar para poder comenzar a desarrollar en JavaScript. Es de suma importancia que instales alguna de las opciones de IDE para que realices las prácticas. Recuerda que el mejor IDE para el desarrollo en JavaScript es en el que te sientas más cómodo para echar a andar tu imaginación.

Además, se analizó la importancia de comenzar a trabajar en entornos colaborativos. Comienza a practicar creando ambientes remotos para que te familiarices con los ambientes que se tienen actualmente en la mayoría de las empresas que aplican las mejores prácticas para el desarrollo de software.

Las características de JavaScript que se revisaron en este tema te permitirán conocer el lenguaje y familiarizarte con la sintaxis básica para poder comenzar a desarrollar tus propios proyectos y marcar el inicio de una prominente carrera como desarrollador de software.

Finalmente, vale la pena reflexionar en el dominio tan abrumador que tiene JavaScript frente a otros lenguajes de programación:

- ¿Cuál crees que sea la mayor fortaleza de JavaScript?
- ¿Cuál es la razón principal por la que JavaScript sigue dominando el mercado de los lenguajes de programación?
- ¿Cuáles serían algunos factores para que cambie la tendencia actual del desarrollo de aplicaciones basadas en la web?

# Referencias bibliográficas


- Arbeláez, R. (2021). *Javascript Nivel Junior*. Edición Kindle.
- Flanagan, D. (2020). *JavaScript: the Definitive Guide* (7a ed.). Estados Unidos: O'Reilly Media, Inc.
- Haverbeke, M. (2018). *ELOQUENT JAVASCRIPT* (3ª ed.). Estados Unidos: No Starch Press.
- LOS TIEMPOS CAMBIAN. (2013). *Cuatro factores para el éxito de JavaScript*. Recuperado de http://www.lostiemposcambian.com/blog/javascript/cuatro-factores-para-el-exito-de-javascript/
- Merino, M. (2022). *JavaScript es el lenguaje más usado (por 10° año consecutivo) y Clojure el mejor pagado en la encuesta de Stack Overflow de 2022*. Recuperado de https://www.genbeta.com/desarrollo/javascript-lenguaje-usado-10o-ano-consecutivo-clojure-mejor-pagado-encuesta-stack-overflow-2022


# Para saber más

**Los siguientes enlaces son externos a la Universidad Tecmilenio, al acceder a ellos considera que debes apegarte a sus términos y condiciones.**

**Lecturas**

- MDN. (s.f.). *Un primer acercamiento a JavaScript*. Recuperado de https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/A_first_splash

**Videos**

- Fazt. (2018, 31 de agosto). *Curso Javascript para Principiantes* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=RqQ1d1qEWlE
- Fazt. (2020, 28 de diciembre). *¿Qué se puede hacer con Javascript? (Juegos, Apps, Desktop, CIIs, etc)* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=qY2JD78kShQ

## Checkpoints

**Asegúrate de:**

- Utilizar correctamente los elementos básicos del lenguaje JavaScript para que tus conocimientos sean sólidos y se fortalezcan con el avance del curso.
- Usar el entorno de trabajo que más se te acomode para poder desarrollar programas.
- Crear una cuenta en GitHub para el trabajo colaborativo.

## Requerimientos Técnicos

- Computadora con acceso a Internet.
- Editor de texto.
- Permisos de administrador y/o Git instalado previamente.

## Prework

**Los siguientes enlaces son externos a la Universidad Tecmilenio, al acceder a ellos considera que debes apegarte a sus términos y condiciones:**

- Descarga la herramienta Git del sitio oficial dependiendo de tu sistema operativo y crea una cuenta en GitHub. Si no sabes cómo hacerlo, ve a la página: https://docs.github.com/es/desktop o https://git-scm.com/book/es/v2
- Crea un directorio principal donde coloques todos tus repositorios y archivos.
- Crea una carpeta específica para tu primer proyecto en Git.
- Asegúrate de tener instalado un entorno de desarrollo como Visual Studio Code o un editor de código como Sublime Text. Incluso puedes utilizar uno en línea, por ejemplo, https://playcode.io/
- Se requiere que tengas un navegador web instalado, de preferencia Google Chrome en su última versión.

Tecmilenio no guarda relación alguna con las marcas mencionadas como ejemplo. Las marcas son propiedad de sus titulares conforme a la legislación aplicable, se utilizan con fines académicos y didácticos, por lo que no existen fines de lucro, relación publicitaria o de patrocinio.

---