CARACTERISTICAS JAVASCRIPT

DIEGO ALEJANDRO MESA VASQUEZ

CENTRO DE TECNOLOGIAS
AGROINDUSTRIALES

MEDELLÍN

2023

Introducción

Javascript es uno de los lenguajes de programación mas utilizados en el mundo porque fue la herramienta que mejor se adaptó a las necesidades crecientes de los navegadores web.

Este lenguaje se introdujo en 1995 como una forma de agregar programas a páginas web en el navegador Netscape Navigator. En su momento fue una idea novedosa. En los primeros días de la World Wide Web, **HTML era bastante simple**, y bastante fácil de aprender casi todo lo que se necesitaba saber para agrupar páginas web. Cualquiera podía hacer una Web juntando tablas, texto y añadiendo alguna imagen.

En esa época, empezaban a desarrollarse **las primeras aplicaciones web** y por tanto, las páginas web comenzaban a incluir formularios complejos. Con unas aplicaciones web cada vez más complejas y una velocidad de navegación tan lenta, surgió la necesidad de un lenguaje de programación que s**e ejecutara en el navegador del usuario**. De esta forma, si el usuario no rellenaba correctamente un formulario, no se le hacía esperar mucho tiempo hasta que el servidor volviera a mostrar el formulario indicando los errores existentes.

https://openwebinars.net/blog/que-es-javascript/

Principales diferencias entre los lenguajes compilados e interpretados.

Lenguajes Compilados:

Proceso de Compilación: En los lenguajes compilados, el código fuente se traduce por completo a código máquina (o un equivalente de bajo nivel) en una sola etapa llamada compilación. El resultado de esta compilación es un archivo ejecutable que puede ser ejecutado directamente por la computadora

Rendimiento: Debido a que el código fuente se traduce completamente antes de la ejecución, los programas compilados tienden a ser más eficientes en términos de rendimiento, ya que el código máquina generado está altamente optimizado.

Portabilidad: Los programas compilados generalmente son específicos de la plataforma para la que se compiló el código. Si se desea ejecutar el mismo programa en diferentes plataformas, es necesario compilarlo para cada una de ellas.

Errores de Compilación: En los lenguajes compilados, los errores en el código se detectan durante la fase de compilación. Esto significa que es necesario corregir los errores antes de poder generar un archivo ejecutable.

Lenguajes Interpretados:

Proceso de Interpretación: En los lenguajes interpretados, el código fuente se traduce línea por línea (o instrucción por instrucción) durante la ejecución por un programa llamado intérprete. No se genera un archivo ejecutable separado.

Rendimiento: Los programas interpretados generalmente son más lentos que los compilados, ya que cada instrucción debe ser traducida y ejecutada en tiempo real.

Portabilidad: Los programas interpretados suelen ser más portátiles, ya que solo se necesita un intérprete compatible en la plataforma de destino para ejecutar el programa. No es necesario recompilar el código para cada plataforma.

Errores de Ejecución: En los lenguajes interpretados, los errores pueden detectarse durante la ejecución, lo que permite una corrección más flexible de los problemas sin necesidad de recompilar.

En resumen, las diferencias clave entre los lenguajes compilados e interpretados radican en cómo se traduce y ejecuta el código. Los lenguajes compilados se traducen por completo antes de la ejecución y tienden a ser más eficientes en términos de rendimiento, mientras que los lenguajes interpretados se traducen y ejecutan línea por línea, lo que ofrece una mayor portabilidad y flexibilidad en la detección de errores

Características principales de JavaScript

- Lenguaje **de Programación del Lado del Cliente:** JavaScript es principalmente un lenguaje de programación del lado del cliente, lo que significa que se ejecuta en el navegador web del usuario. Permite la interacción dinámica y la modificación del contenido de la página sin tener que recargarla.

- **Lenguaje Interpretado:** JavaScript es un lenguaje interpretado, lo que significa que no requiere un proceso de compilación. El código fuente se ejecuta directamente en el navegador o en un entorno de servidor que lo admite.

- **Sintaxis Sencilla:** JavaScript tiene una sintaxis relativamente sencilla y fácil de aprender, lo que lo hace accesible para programadores principiantes y experimentados.

- **Orientado a Objetos:** Aunque JavaScript es conocido como un lenguaje de programación orientado a objetos, su modelo de objetos es basado en prototipos. Esto significa que los objetos se crean a partir de otros objetos existentes en lugar de usar clases tradicionales.
- **Dinamismo:** JavaScript es un lenguaje dinámico, lo que significa que las variables no están vinculadas a un tipo de dato específico y pueden cambiar de tipo durante la ejecución.

- **Interacción con el DOM:** JavaScript se utiliza ampliamente para manipular el Document Object Model (DOM) de una página web. Esto permite la modificación de contenido, estilos y comportamientos de la página en respuesta a eventos o acciones del usuario.

- **Gestión de Eventos:** JavaScript permite la captura y el manejo de eventos, como hacer clic en un botón o mover el mouse, lo que permite crear interacciones y experiencias más dinámicas en las aplicaciones web.

- **Librerías y Frameworks:** Existen numerosas librerías y frameworks de JavaScript, como jQuery, React, Angular y Vue.js, que facilitan el desarrollo de aplicaciones web complejas y modernas.

- **Asincronismo:** JavaScript es inherentemente asincrónico, lo que significa que puede manejar operaciones que no bloquean la ejecución del programa, como solicitudes de red, de manera eficiente utilizando callbacks, promesas o async/await.

- **Compatibilidad con Plataformas Múltiples:** JavaScript se ejecuta en la mayoría de los navegadores web modernos, lo que lo convierte en un lenguaje multiplataforma y ampliamente compatible.

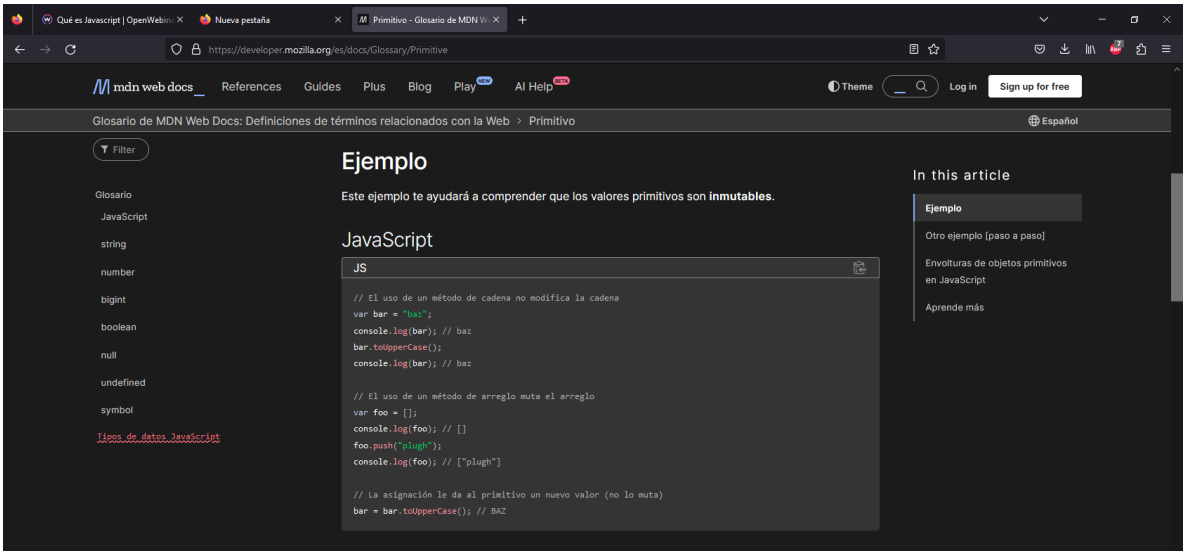
- **Comunidad Activa:** La comunidad de desarrolladores de JavaScript es grande y activa, lo que ha llevado al crecimiento constante del ecosistema de herramientas y recursos

Tipos de datos primitivos y uso en JavaScript

En **JavaScript**, un **primitive** (valor primitivo, tipo de dato primitivo) son datos que no son un **objeto** y no tienen **métodos**. Hay 6 tipos de datos primitivos: **string, number, bigint, boolean, undefined** y **symbo**l. También hay **null**, que aparentemente es primitivo, pero de hecho es un caso especial para cada **Object**: y cualquier tipo estructurado se deriva de null por la **Cadena de prototipos**.

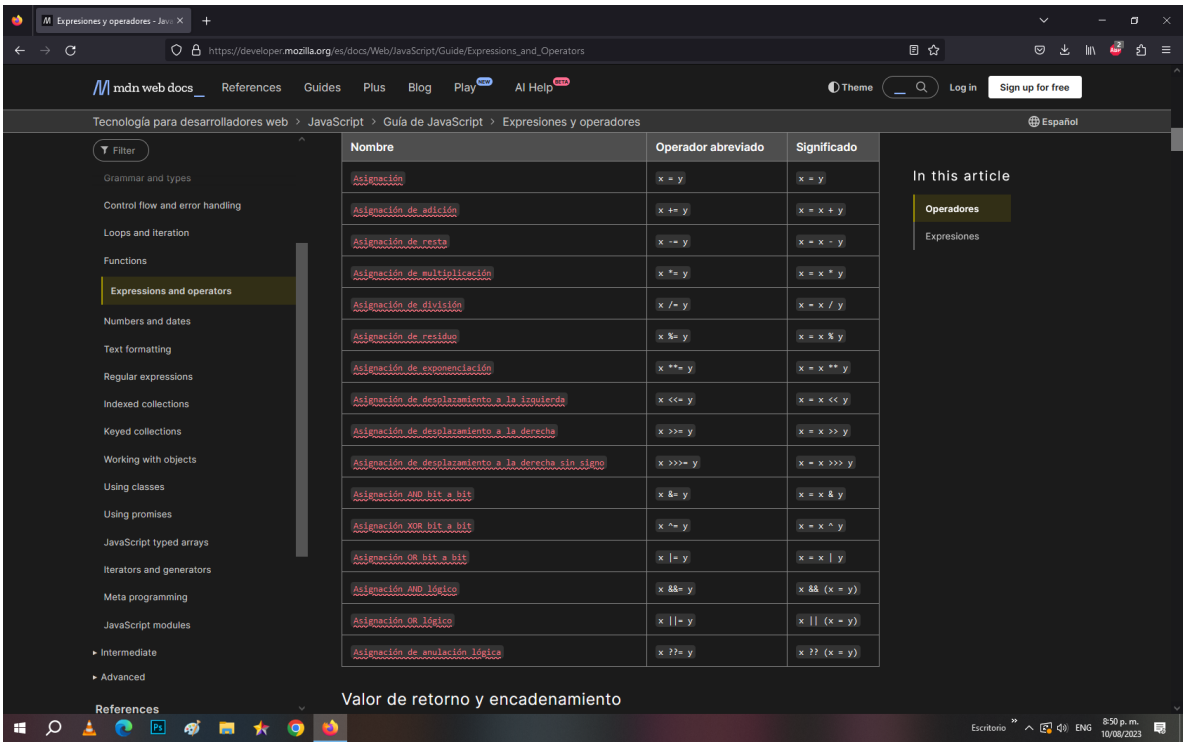
La mayoría de las veces, un valor primitivo se representa directamente en el nivel más bajo de la implementación del lenguaje.

Todos los primitivos son **inmutables**, es decir, no se pueden modificar. Es importante no confundir un primitivo en sí mismo con un valor primitivo asignado a una variable. Se puede reasignar un nuevo valor a la variable, pero el valor existente no se puede cambiar de la misma forma en que se pueden modificar los objetos, los arreglos y las funciones.

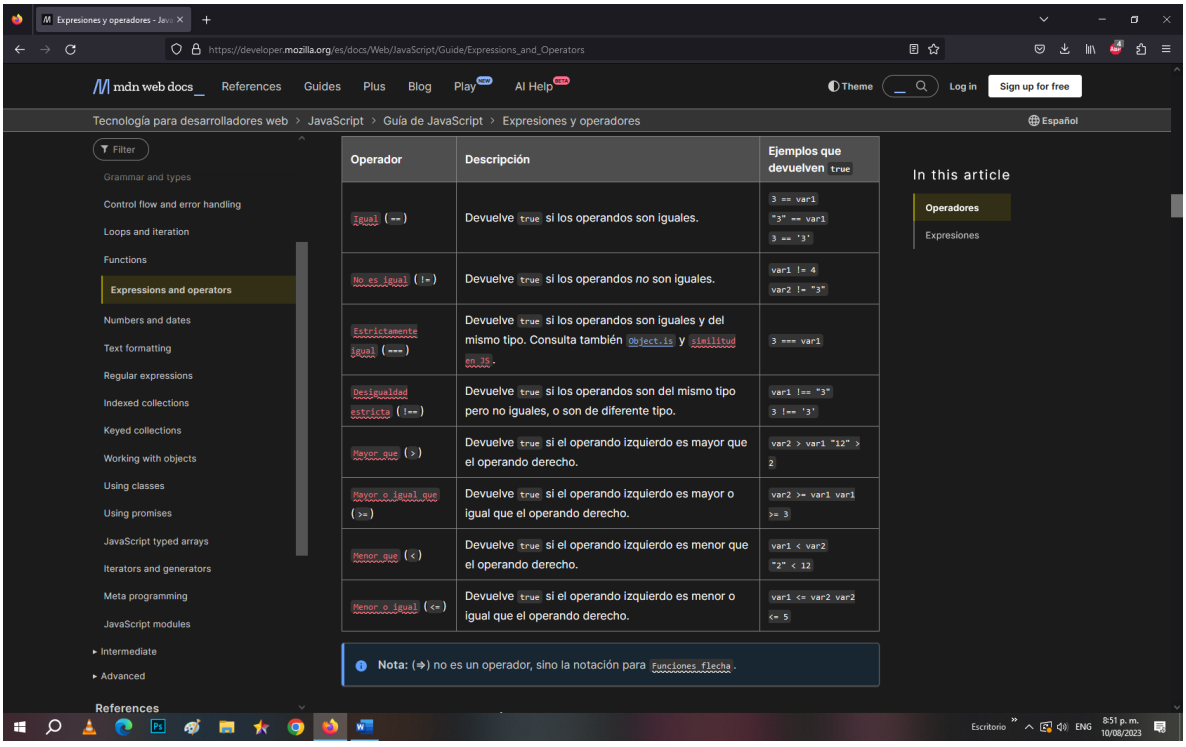


https://developer.mozilla.org/es/docs/Glossary/Primitive

Operadores en JavaScript

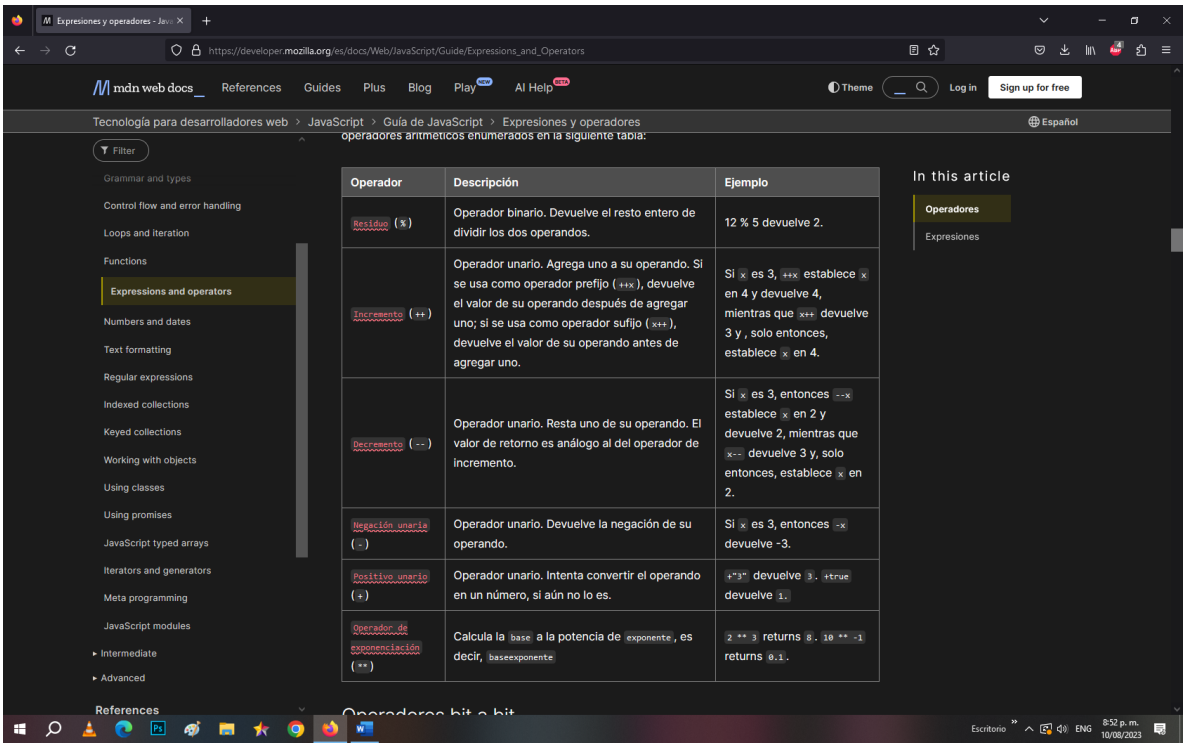


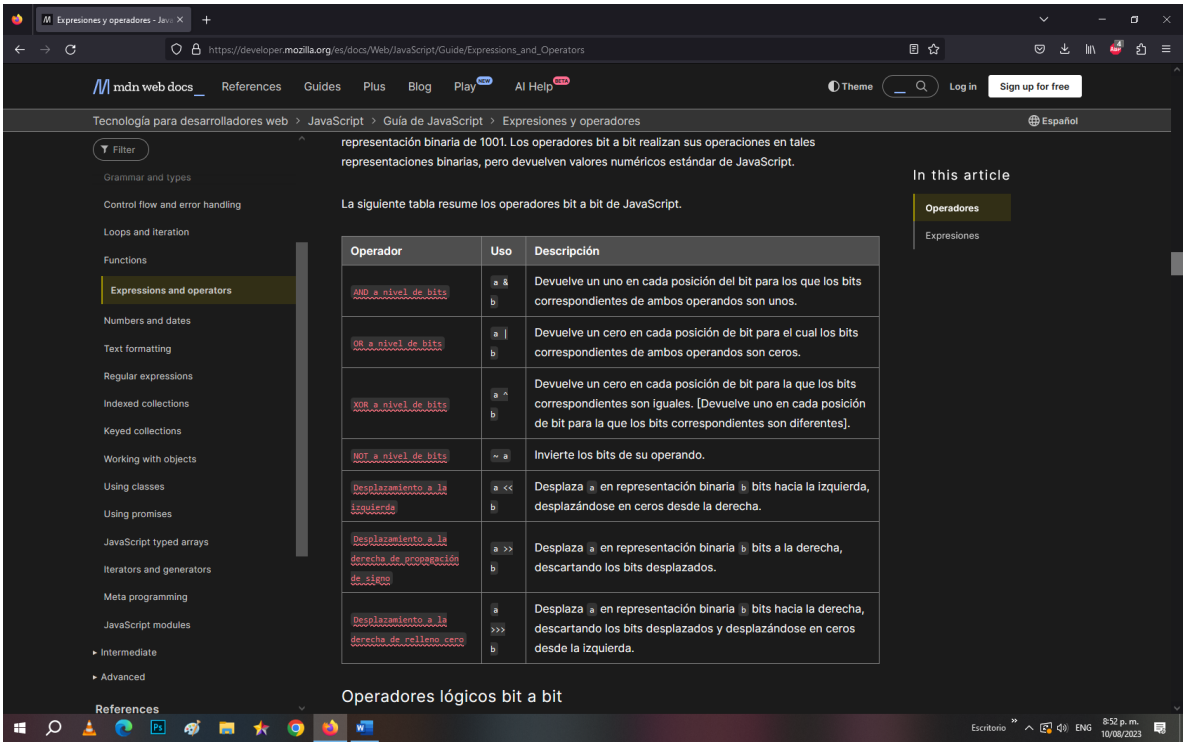
Nombre	Operador abreviado	Significado
Asignación	x = y	x = y
Asignación de adición	x += y	x = x + y
Asignación de resta	x -= y	x = x - y
Asignación de multiplicación	x *= y	x = x * y
Asignación de división	x /= y	x = x / y
Asignación de residuo	x %= y	x = x % y
Asignación de exponenciación	x **= y	x = x ** y
Asignación de desplazamiento a la izquierda	x <<= y	x = x << y
Asignación de desplazamiento a la derecha	x >>= y	x = x >> y
Asignación de desplazamiento a la derecha sin signo	x >>>= y	x = x >>> y
Asignación AND bit a bit	x &= y	x = x & y
Asignación XOR bit a bit	x ^= y	x = x ^ y
Asignación OR bit a bit	x |= y	x = x | y
Asignación AND lógico	x &&= y	x && (x = y)
Asignación OR lógico	x ||= y	x || (x = y)
Asignación de anulación lógica	x ??= y	x ?? (x = y)



Operador	Descripción	Ejemplos que devuelven `true`
Igual (==)	Devuelve `true` si los operandos son iguales.	`3 == var1` `"3" == var1` `3 == '3'`
No es igual (!=)	Devuelve `true` si los operandos *no* son iguales.	`var1 != 4` `var2 != "3"`
Estrictamente igual (===)	Devuelve `true` si los operandos son iguales y del mismo tipo. Consulta también `Object.is` y similitud en JS.	`3 === var1`
Desigualdad estricta (!==)	Devuelve `true` si los operandos son del mismo tipo pero no iguales, o son de diferente tipo.	`var1 !== "3"` `3 !== '3'`
Mayor que (>)	Devuelve `true` si el operando izquierdo es mayor que el operando derecho.	`var2 > var1` `"12" > 2`
Mayor o igual que (>=)	Devuelve `true` si el operando izquierdo es mayor o igual que el operando derecho.	`var2 >= var1` `var1 >= 3`
Menor que (<)	Devuelve `true` si el operando izquierdo es menor que el operando derecho.	`var1 < var2` `"2" < 12`
Menor o igual (<=)	Devuelve `true` si el operando izquierdo es menor o igual que el operando derecho.	`var1 <= var2` `var2 <= 5`

Nota: (⇒) no es un operador, sino la notación para `Funciones flecha`.





https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Expressions_and_Operators