ACADEMY by NUMEN

FULL STACK

Comenzamos en unos minutos


ACADEMY
by NUMEN

Javascript: Clase 1
Tipos de datos primitivos , variables, y operadores aritméticos y lógicos.

ACADEMY by NUMEN

Preparación

Para poder iniciar con Javascript debe ya tener instalados los siguientes programas:

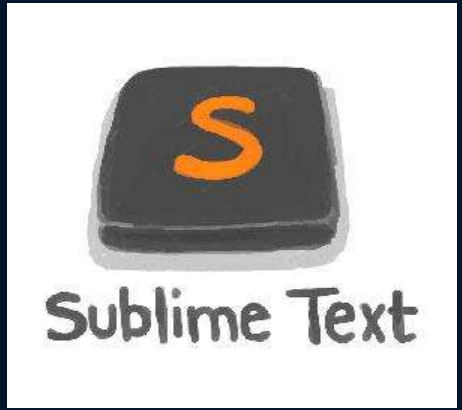
- Editor de Texto
- Github (Cuenta)
- Git (Gitbash)
- Node Js

En el campus de la academia podrán encontrar video tutoriales para cada uno de estos programas

Editores de texto

Son programas que te permiten realizar o escribir código fuente de tus proyectos. Al ser dinámicos, son idóneos para cuando desarrollas uno con varios lenguajes de programación. Algunos de ellos son:

- **Sublime Text**
- **Atom**
- **Visual Studio Code**



GitHub

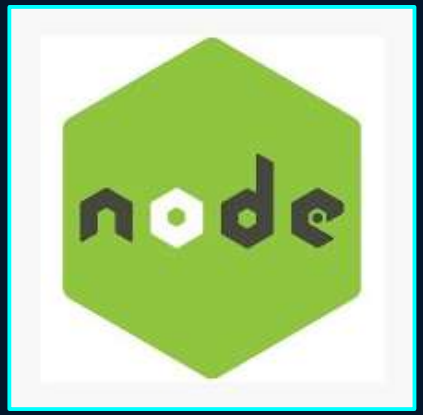
Es una red para almacenar tus repositorios, sería un repositorio de repositorios. Es uno de los tantos disponibles en internet, y el más popular. GitHub NO es lo mismo que Git, aunque funcionen muy bien juntos. Github es un lugar donde podés compartir tu código o encontrar otros proyectos. También actúa como portfolio para cualquier código en el que hayas trabajado.

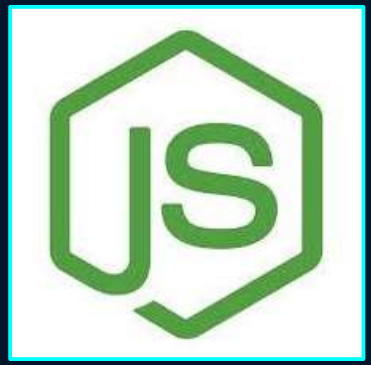
Git / GitBash

Git Bash es una aplicación para entornos de Microsoft Windows que ofrece una capa de emulación para una experiencia de líneas de comandos de **Git**. Bash es el acrónimo en inglés de Bourne Again Shell. Una shell es una aplicación de terminal que se utiliza como interfaz con un sistema operativo mediante comandos escritos.

Node

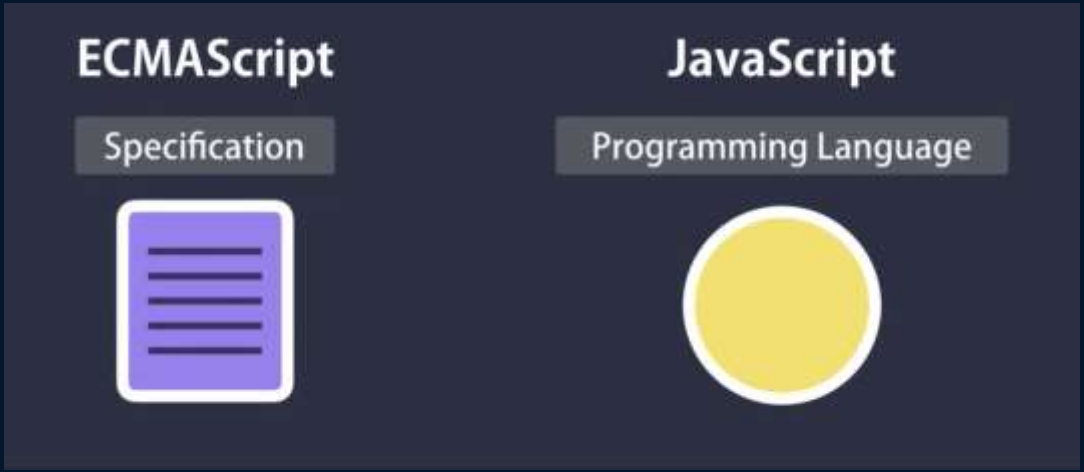
Node. js es un entorno JavaScript que nos permite ejecutar en el servidor, de manera asíncrona, con una arquitectura orientada a eventos y basado en el motor V8 de Google. Es una plataforma que avanza muy rápidamente y cada vez está más presente en el mercado.



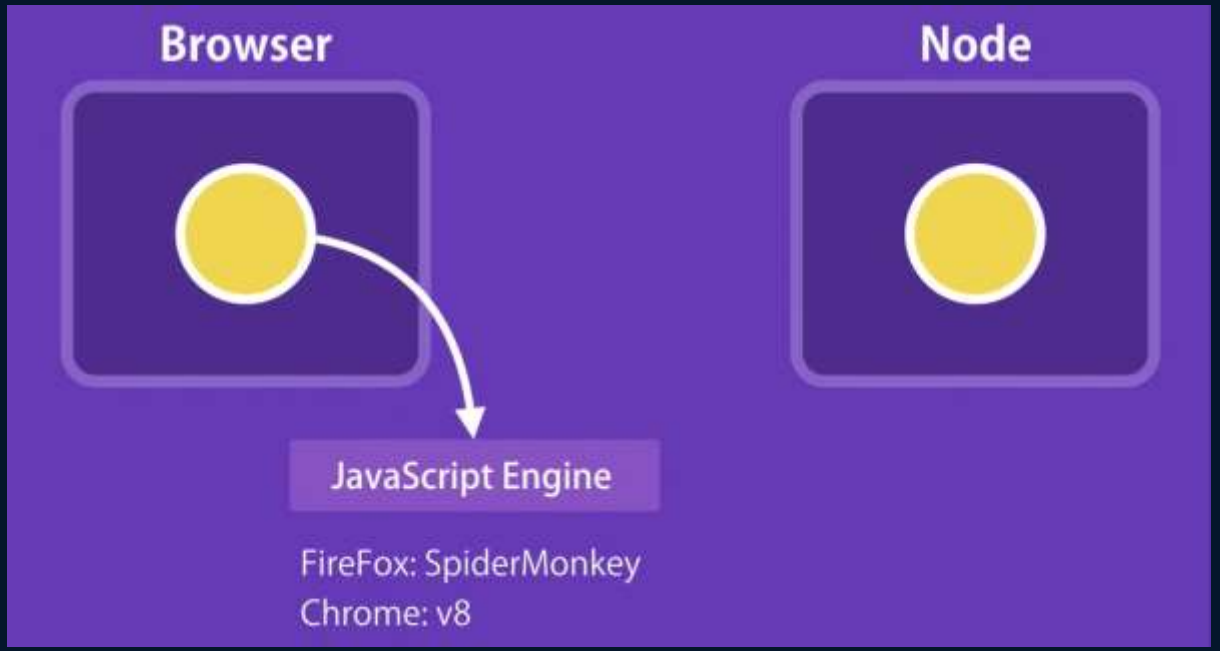


¿Qué es Javascript?

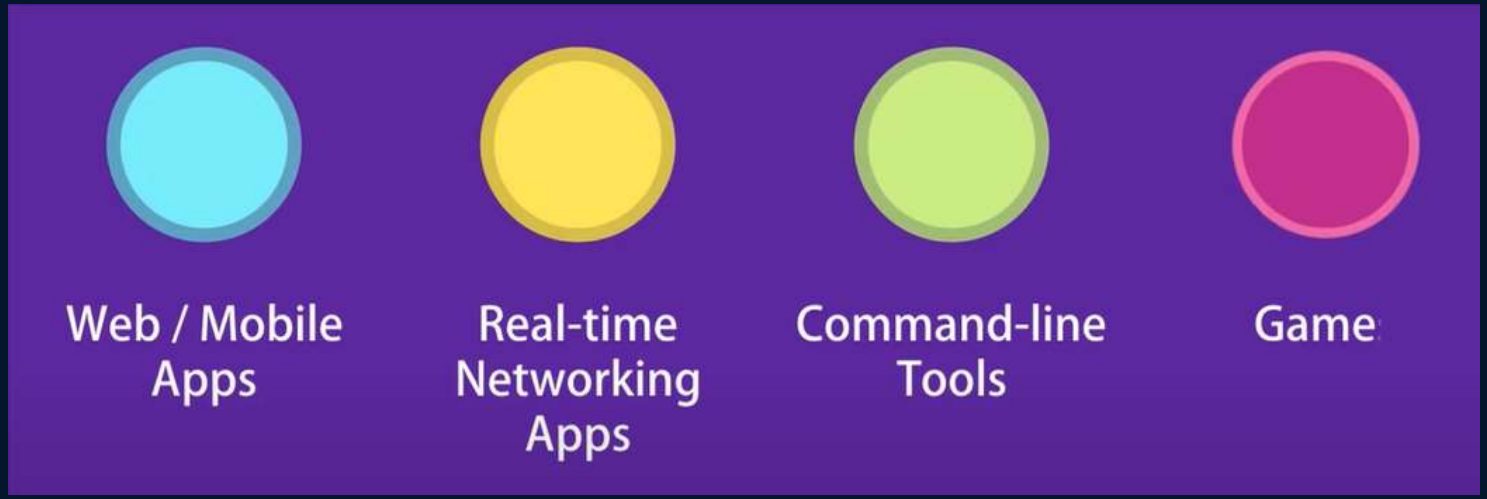
JavaScript es un lenguaje de programación que se ajusta al estándar ECMAScript. La especificación ECMAScript es un estándar destinado a garantizar la interoperabilidad de las páginas web en diferentes navegadores web.



Javascript es el único lenguaje que puede hacer ***isomorfismo***. ¿Qué significa esto? Significa que puede trabajar tanto en Frontend como en Backend. Es decir, es la única tecnología con la que puedes hacer toda una aplicación de principio a fin.



¿Qué puedes hacer con Javascript?



Características de Javascript

- Lenguaje de Alto Nivel
    - Interpretado
    - Dinámico
    - Débilmente tipado
    - Sensible a MAYÚSCULAS y minúsculas
    - No necesitas los puntos y comas al final de cada línea.

Tipos de datos

Primitivos: Se accede **directamente** al valor.

- Undefined
- Boolean (true, false)
- Number
- String (cadena de textos)
- BigInt
- Symbol
- Null

No es valor:

- NaN

Compuestos: Se accede a la **referencia** del valor.

- Object = {}
- Function() {}
- Array = []
- Class {}

Palabras Reservadas

A: abstract

B: boolean, break, byte

C: case, catch, char, class, const, continue

D: debugger, default, delete, do, double

E: else, enum, export, extends

F: false, final, finally, float, for, function

G: goto

I: if, implements, import, in, instanceof, int, interface

L: let, long

N: native, new, null

P: package, private, protected, public

R: return

S: short, static, super, switch, synchronized

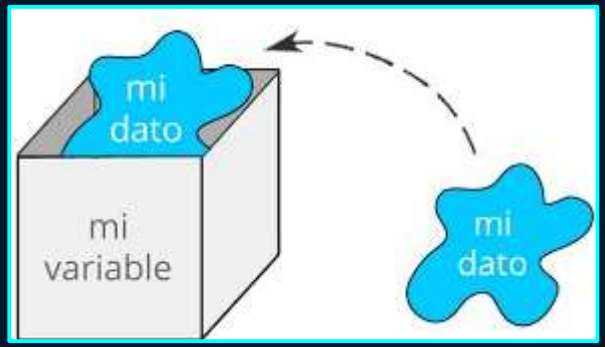
T: this, throw, throws, transient, true, try, typeof

V: var, volatile, void

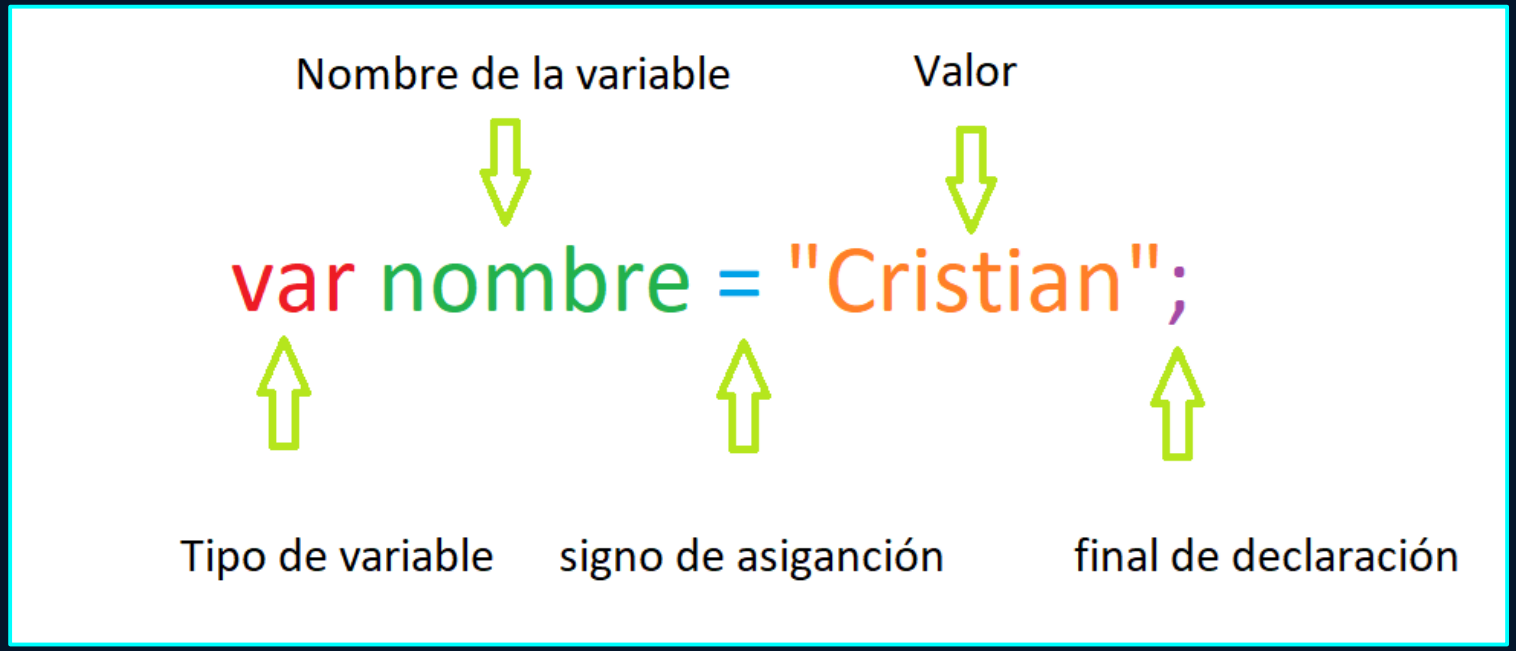
W: while, with

Variables

- Una variable es una forma de almacenar el valor de algo para usar más tarde.
- Una variable es un recurso de memoria del ordenador reservado para alojar una información.
- Las variables tienen un continente que puede o no almacenar un contenido. En caso de no tenerlo javascript guarda un espacio de memoria con el valor *undefined*



Estructura de una variable



Tipos de datos primitivos: Cadena de texto

El **String** se utiliza para representar y manipular una secuencia de caracteres.

Las cadenas son útiles para almacenar datos que se pueden representar en forma de texto.

Por detrás, un string es una colección de caracteres con una posición determinada. Esto nos permitirá interactuar con esas posiciones en futuras clases.

Las cadenas de texto pueden recibir propiedades. Una de ellas es la propiedad **length** que nos permite contar la cantidad de caracteres que componen a una cadena.

```
var cadena = "Academia Numen"

console.log(cadena)

var cadena2 = new String("Numen")

console.log(cadena2)

console.log(cadena.length)
```

Concatenación

Concatenar es una elegante palabra de la programación que significa: "unir". Para unir cadenas en JavaScript el símbolo de más (+), el mismo operador que usamos para sumar números, pero en este contexto hace algo diferente.

Una concatenación se vería de esta manera:

```
var nombre = "Sherlock",
  apellido = "Holmes";

var saludo = "Hola mi nombre es " + nombre + " " + apellido + "."
```

Tipos de datos primitivos: Números

`Number` es un objeto primitivo envolvente que permite representar y manipular valores numéricos cómo 37 o -9.25.

Al igual que con las cadenas, los números son también en el fondo, colecciones de números individuales con posición determinada.

En este caso, podemos notar que además interactuar con propiedades que nos brindan información (como el length en el caso de las cadenas), también es posible interactuar con métodos son, a diferencia de las propiedades, son acciones que modifican los datos.

```
var numero = 2021,
  numeroNegativo = -32,
  numeroConDecimales = 3.1416;

console.log(numero)
console.log(numeroNegativo)
console.log(numeroConDecimales)

var numero2 = new Number("Numen")

console.log(numero2)

console.log(numero.toString())
```

Datos Primitivos: Booleanos

Un **boolean** es un dato lógico que solo puede tener los valores true o false.

Este tipo de dato tiene muchas utilidades. Entre ellas nos permite cerciorarnos del resultado de una comparación, como por ejemplo si un número es igual a otro, o si es mayor o menor a otro, etc.

También nos permite programar en falso cuando algo está apagado de modo cambie a verdadero cuando está encendido y muchas cosas más.

```
var verdadero = true;
var falso = false;

var v = new Boolean(true)
var f = new Boolean(false)

console.log(verdadero)
console.log(falso)
console.log(v)
console.log(f)

console.log(new Boolean(""))
console.log(new Boolean("Hola mundo"))
console.log(new Boolean(0))
console.log(new Boolean(1))
```

Tipos de Datos Primitivos: Not a Number

La propiedad global NaN es un valor que representa Not-A-Number (no es un número). Es un tipo de dato que se arrojará cada vez que se realice una operación que involucre operadores numéricos pero que como resultado no pueda dar un número. También puede utilizarse intencionalmente para corroborar que el valor que se ingrese sea un número, pero esto lo último lo veremos más adelante.

```
var noEsUnNumero = "Hola" * 3.7;

console.log(noEsUnNumero);
```

Tipos de datos primitivos: Nulo

El valor `null` es un literal de Javascript que representa intencionalmente un valor nulo o "vacío".

```
var noSeAsignoUnValor = null;

console.log(noSeAsignoUnValor);
```

Tipos de datos primitivos: Indefinido

El valor undefined representa un valor que aún no se ha definido.
Una variable a la que no se le ha asignado valor, o no se ha declarado en absoluto (no se declara, no existe) son de tipo undefined. Un método o sentencia también devuelve undefined si la variable que se está evaluando no tiene asignado un valor. Una función devuelve undefined si no se ha devuelto un valor.

```
var noSeAsignoValor;

console.log(noSeAsignoValor);

console.log(nombre);
```

Undefined y null

Non-zero value

null

0

undefined

Operadores aritméticos

Operador	Descripción	Valor de y	Operación	Valor de x
+	Suma	5	x = y + 2	7
-	Resta	5	x = y - 2	3
*	Multiplicación	5	x = y * 2	10
/	División	5	x = y / 2	2.5
%	Resto división	5	x = y % 2	1
--	Decrementar	Dado y=5		
		4	y--	
		4	x = --y	4
		4	x = y--	5
++	Incrementar	Dado y=5		
		6	y++	
		6	x = ++y	6
		6	x = y++	5

Operadores de comparación

Los operadores de comparación son los siguientes:

```
>    >=    <    <=    ===    !==
```

```
var mayorQue = 1 > 2;
var menorQue = 2 < 3;

var mayorOIgualQue = 10 == 10;
var menorOIgualQue = 10 <= 10;

console.log(mayorQue);
console.log(menorQue);
console.log(mayorOIgualQue);
console.log(menorOIgualQue);
```

Estos operadores funcionan como lo harían en una clase de matemáticas, mayor que, menor que, etc. Utilizamos estos operadores para evaluar dos expresiones. A medida que la computadora ejecuta el código, el operador devolverá un verdadero (si la declaración es verdadera) o un falso.

Operadores de comparación

El "triple igual" (===) no debe confundirse con un solo signo igual (que indica asignar un valor a una variable). El triple igual comparará todo sobre los dos elementos, incluido el tipo, y devolverá si son exactamente iguales o no.

El “doble igual” (==) solo verifica la igualdad de valores. Hace coerción de tipos inherentemente. Esto significa que antes de verificar los valores, convierte los tipos de las variables para que coincidan.

```
// igual que
var dobleIgual = 1 == 1;
var dobleIgual = 1 == '1';
var tripleIgual = 1 === 1;
var tripleIgual = 1 === '1';

console.log(dobleIgual);
console.log(tripleIgual);
```

```
// Distinto que
var distinto = 1 != 1;
var distinto = 1 != '1';
var distinto2 = 1 !== 1;
var distinto2 = 1 !== '1';
```

¿Cuales creen que serán los resultados de estas operaciones?

Operadores lógicos

Los operadores lógicos se utilizan normalmente con valores booleanos (lógicos); cuando ese el caso, devuelven un valor booleano. Sin embargo, los operadores && y || en realidad devuelven el valor de uno de los operandos especificados, por lo que si estos operadores se utilizan con valores no booleanos, pueden devolver un valor no booleano. Los operadores lógicos se describen en la siguiente tabla.

Operador	Descripción	Ejemplo
&&	AND: Este nos retorna verdadero si los dos valores a comparar son verdaderos de lo contrario retorna false	var1&&var2
||	OR: Nos retorna verdadero si alguno de los dos valores a comparar es verdadero sino false	var1||var2
!	NOT: Niega el valor, si la variable es true la devuelve falso y a la inversa	!var1

Coerción de datos

Como mencionamos anteriormente, JavaScript es un lenguaje de programación débilmente tipado. Esto significa que al momento de crear variables no hace falta que indiques de qué tipo de dato será el valor contenido, y que puede llegar a cambiar durante la ejecución de tu código. JavaScript también es un lenguaje de tipado dinámico. Esto hace referencia a que la verificación de tipos de datos se realiza en tiempo de ejecución..

```
let variable = "texto";
console.log(typeof variable); // "string"
variable = 42;
console.log(typeof variable); // "number"
```

Estas características que definen a JavaScript pueden ser ventajosas, como también perjudiciales. La flexibilidad provista al no definir el tipo de datos de las variables puede acarrear errores en tus proyectos, si el código no se implementa de la manera adecuada. Puedes crear una variable y asignarle un valor de cierto tipo de dato, y en el transcurso de la ejecución del código, cambiar el valor de esa variable a uno de otro tipo de dato, y JavaScript seguirá ejecutando tu aplicación normalmente. Si en alguna parte de tu código asumes que la variable tiene un valor de cierto tipo, y en realidad es de otro, puede que se genere un error en la ejecución, o que tu código devuelva un resultado incorrecto.

```
let variable1 = "1";

let variable2 = variable1 + 1;

console.log(variable2); // 11
```

Se llama coerción de datos, o coerción de tipos, al proceso mediante el cual JavaScript convierte el valor de una variable de un tipo a otro. Este proceso se realiza al momento de ejecutar una operación donde uno de los operandos contiene un valor de un cierto tipo de datos, pero el otro operando posee un valor de otro tipo.

null

El intérprete de JavaScript hace la conversión de tipos para adaptarse a las operaciones que defines en tu código. De esta manera, se convierte el valor asignado a una de las variables involucradas en la operación a un valor “equivalente” del tipo de dato del otro operando.

Por medio de la coerción de datos, JavaScript permite operar con valores sin importar si ellos son de distintos tipos de datos. Puedes sumar el valor de una variable de tipo number con un valor string. También puedes comparar un valor de tipo boolean con otro de tipo number. Puedes concatenar un number con un string y formar una nueva cadena de texto.

Tipos de coerción de datos

Existen dos formas de coerción de datos: Implícita y Explícita.

null

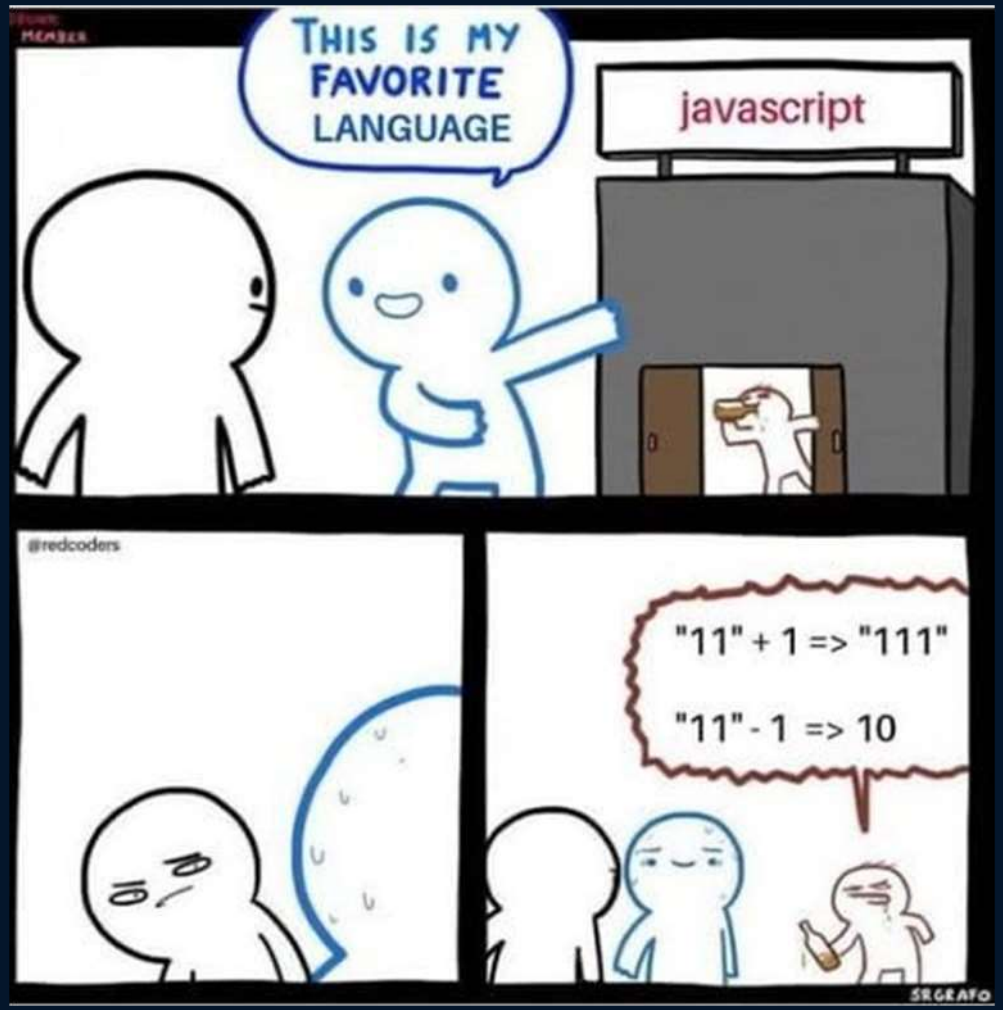
La coerción implícita es la que se aplica automáticamente cuando intentas ejecutar una operación con dos valores de distintos tipos. En este caso, JavaScript intenta interpretar los valores y convertir uno de ellos al tipo de dato del otro valor, para que la operación se pueda llevar a cabo.

```
let variable1 = "1";

let variable2 = variable1 + 1;

console.log(variable2); // 11
```

En cambio, la coerción explícita es el proceso mediante el cual el programador indica explícitamente, usando ciertas funciones provistas por JavaScript, a qué tipo de dato se desea convertir un valor.

```
Boolean(0) // false

Boolean(1) // true

Number("") // 0

Number("1") // 1

String(10) // "10"
```

null


THIS IS MY FAVORITE LANGUAGE
javascript
@redcoders
"11" + 1 => "111"
"11" - 1 => 10
SRGRAFO

Tarea sobre coerción de datos

¿Cuál crees que será el resultado de la ejecución de estas operaciones?

```
6 / "3" =
"2" * "3" =
4 + 5 + "px" =
"$" + 4 + 5 =
"4" - 2 =
"4px" - 2 =
7 / 0 =
parseInt("09") =
```

```
3>2>1 =
12 + "" =
"15" * 2 =
"15" - "11" =
undefined + 6 =
"Hello" + null =
null + 25 =
```

Piensa primero cuál será el resultado y después prueba ver la respuesta usando la siguiente pagina: https://jsconsole.com/

ACADEMY
by NUMEN