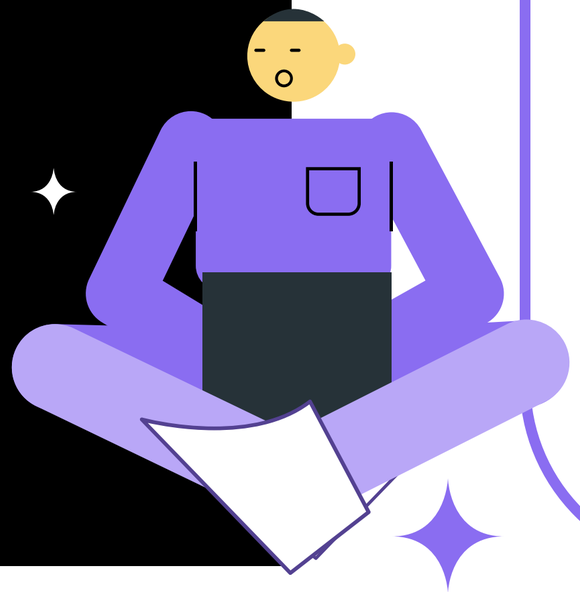
DIGITALHOUSE

Introducción a Javascript

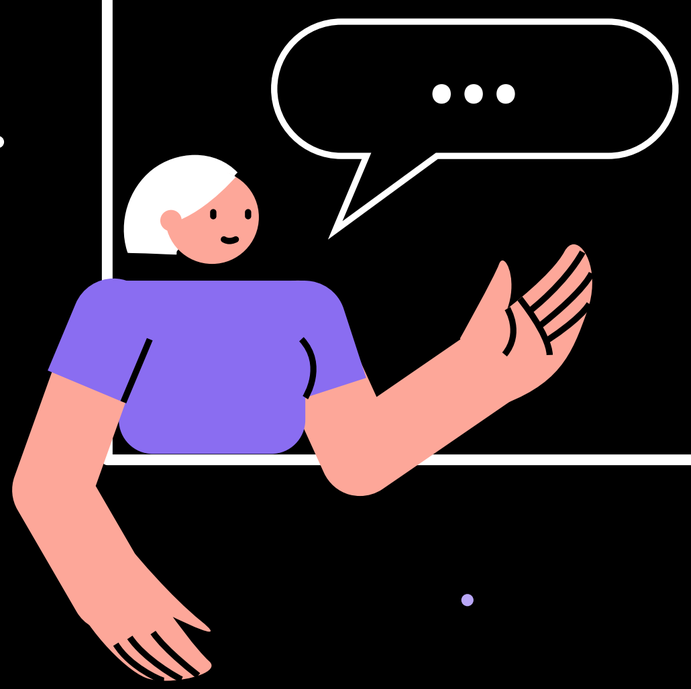
Índice

01 Introducción a Javascript

JavaScript es el lenguaje de programación encargado de dotar de mayor interactividad y dinamismo a las páginas web.



El código de programación de JavaScript se **ejecuta en los navegadores**, ya sean de escritorio o móviles sin importar que sea Android o Iphone.



¿Para qué sirve JS?

01 Permite darle dinamismo a las páginas web.

02 Es capaz de detectar errores en formularios, crear bonitos sliders que se adapten a cualquier pantalla.

03 Hacer cálculos matemáticos de forma eficiente.

04 Modificar elementos de una página web de forma sencilla.

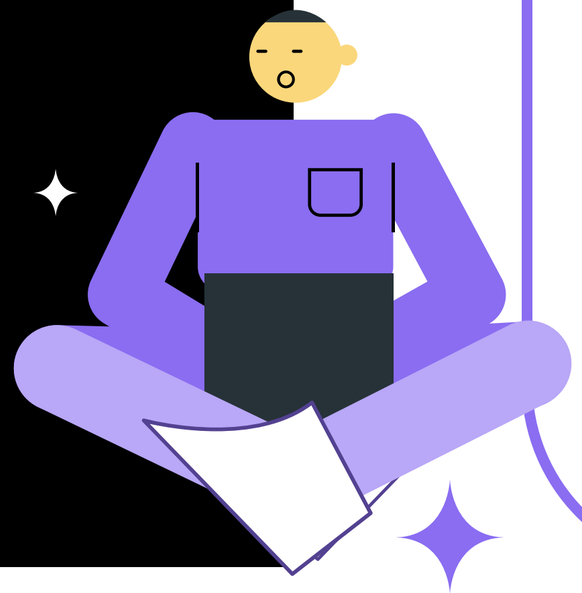
05 Nos permite reutilizar código para optimizar el rendimiento.

06 Intercambiar información sin necesidad de recargar la página

07 Podemos crear efectos y animaciones sin ninguna interacción o respondiendo a eventos

02 Variables

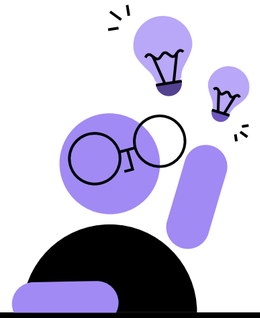
Las **variables** son **espacios de memoria** en la computadora donde podemos **almacenar** distintos tipos de **datos**.



Tipos de variables

En Javascript existen tres tipos de variables:

- **var**
- **let**
- **const**



Const a diferencia de let y var debe inicializarse como veremos a continuación

```
var nombre;
let contador;
const url = “https://www.digitalhouse.com”; //
```

Veamos cada parte en más detalle...

Declaración de una variable

```
var nombreSignificativo;
```

Var

La palabra reservada var le indica a JavaScript que vamos a **declarar una variable de tipo var**.

Nombre

Solo puede estar formado por letras, números y los símbolos $ (pesos) y _ (guión bajo).
No pueden empezar con un número.
No deberían contener ñ o caracteres con acentos.

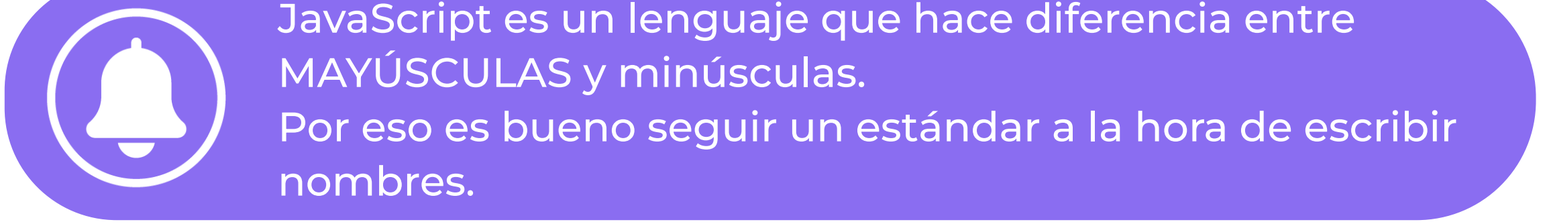
Es una buena práctica que los nombres de las variables usen el formato camelCase, como **variableEjemplo** en vez de variableejemplo o variable_ejemplo

Declaración de una variable

```
var miVariable;
```

No es lo mismo que:

```
var MiVariable;
```



Las buenas prácticas, si bien no son obligatorias para que nuestro código funcione, van a permitir que este sea más fácil de leer y de mantener.

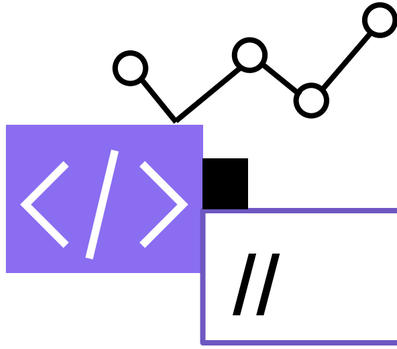
Asignación de un valor

```
var miApodo = “Hackerman”;
```

Nombre
El nombre que nos va a servir para identificar nuestra variable cuando necesitemos usarla.

Asignación
Le indica a JavaScript que queremos guardar el valor de la derecha en la variable de la Izquierda.

Valor
Lo que vamos a guardar en nuestra variable. En este caso, un texto.



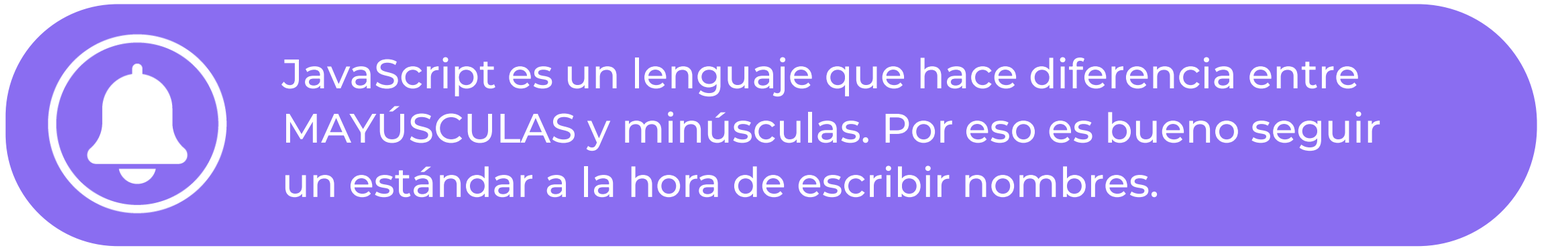
Asignación de un valor

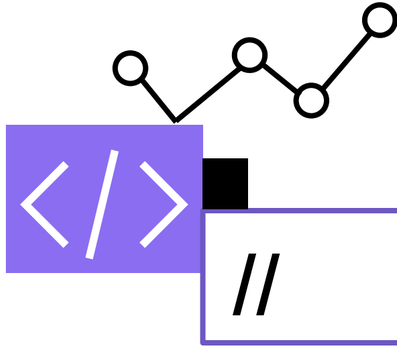
La **primera vez** que declaramos una variable es necesaria la palabra reservada **var**.

```
{}  var miApodo = 'Hackerman';
```

Una vez que la variable ya fue declarada, le asignamos valores sin var.

```
{}  miApodo = 'El Barto';
```





Declaración con let

Estas variables se declaran de una manera similar con la diferencia que utilizamos la palabra reservada let.

```
let contador = 1;
```

La principal diferencia entre var y let es que let solo será accesible en el bloque de código en el que fue declarada.

Los bloques de código son normalmente determinados por las llaves { }.

Veamos un ejemplo:

var

```
if (true) {
  var nombre = "Juan";
}

console.log(nombre);
// Ok, muestra "Juan"
```

Cuando usamos **var**, JavaScript ignora los bloques de código y **convierte nuestra variable en global**.

Eso quiere decir que si hay otra variable nombre en nuestro código, seguramente estemos pisando su valor.

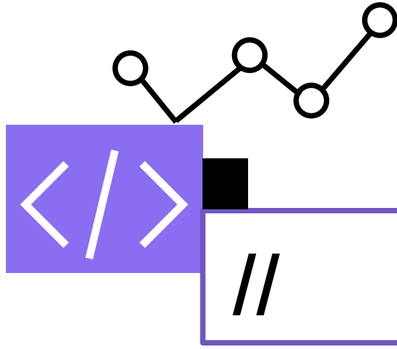
let

```
if (true) {
  let nombre = "Juan";
}

console.log(nombre)
// Error: nombre no existe
```

Cuando usamos **let**, JavaScript **respeta los bloques de código**. Eso quiere decir que nombre no podrá ser accedida fuera del if.

También quiere decir que podemos tener variables con el mismo nombre en diferentes bloques de nuestro código.



Declaración con const

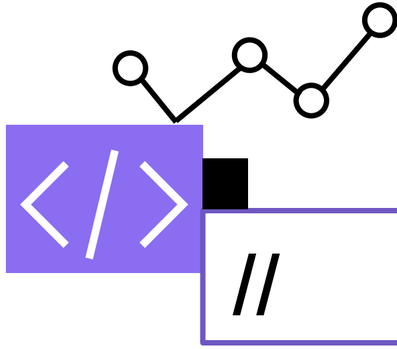
Las variables const se declaran con la palabra reservada const.

```
{}  const email = "mi.email@digitalhouse.com";
```

Las variables declaradas con const funcionan igual que las variables let, estarán disponibles solo en el bloque de código en el que se hayan declarado.

Al contrario de let, una vez que les asignemos un valor, no podremos cambiarlo

```
{}  email = "mi.otro.email@digitalhouse.com";
    // Error de asignación, no se puede cambiar
    // el valor de const
```



Declaración con let o const

Como dijimos antes, tanto **let** como **const** son accesibles dentro del bloque donde son declaradas.
Por esta razón solo podemos declararlas una vez. Si volvemos a declararlas, JavaScript nos devolverá un error.

{}

```
let contador = 0;
let contador = 1;
// Error de re-declaración de la variable

const email = “mi.email@hotmail.com”;
const email = “mi.nuevo.email.com@hotmail.com”;
// Error de re-declaración de la variable
```

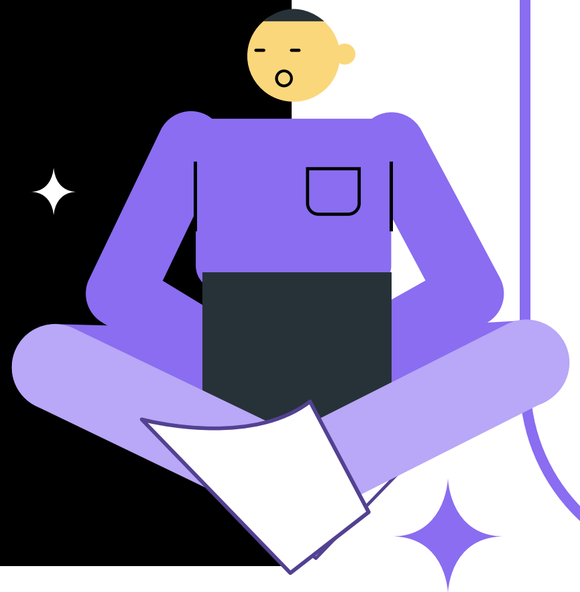
Las palabras reservadas como var, let y const solo pueden utilizarse para el propósito que fueron creadas.

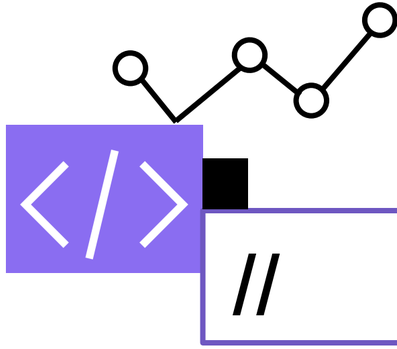
No pueden ser utilizadas como:

nombre de variables, funciones, métodos o identificadores de objetos.

03 Tipos de datos

Los **tipos de datos** le **permiten** a JavaScript conocer las **características** y **funcionalidades** que estarán disponibles para ese dato.





Numéricos (number)

```
let edad = 35;  // número entero
let precio = 150.35;  // decimales
```

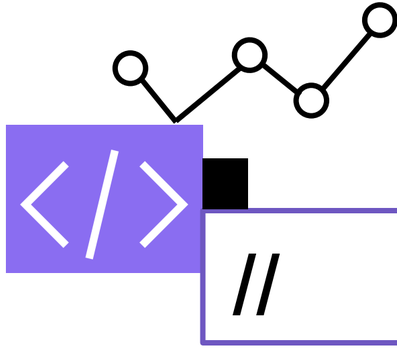
Como JavaScript está escrito en inglés usaremos un punto para separar los decimales.

Cadenas de caracteres (strings)

```
let nombre = 'Mamá luchetti';  // comillas simples
let ocupacion = "Master of the sopas";  // comillas dobles
```

Lógicos o booleanos

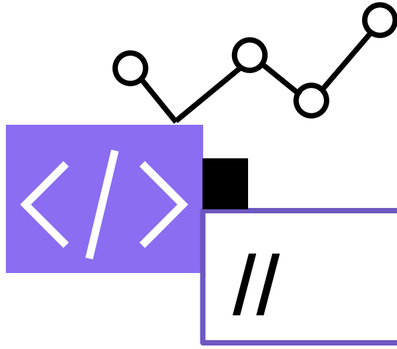
```
let laCharlaEstaCopada= true;
let hoyBajaElDolar = false;
```



Objetos (object)

A diferencia de otros tipos de datos que pueden contener un solo dato, los objetos son colecciones de datos y en su interior pueden existir todos los anteriores.
Los podemos reconocer porque se declaran con llaves { }.

```
{}
let persona = {
  nombre: 'Javier', // string
  edad: 34, // number
  soltero: true // boolean
}
```

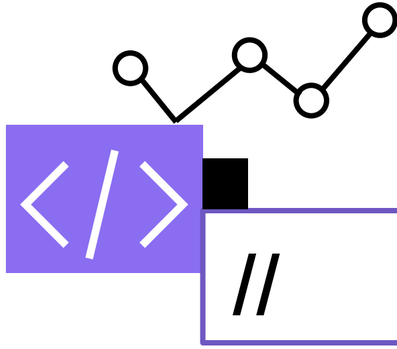


Array

Al igual que los objetos, los arrays son colecciones de datos. Los podemos reconocer porque se declaran con corchetes **[ ]**.
Los arrays son un tipo especial de objetos, por eso **no los consideramos como un tipo de dato más**.
Los mencionamos de manera especial porque son muy comunes en todo tipo de código.

```
{}
let comidasFavoritas = ['Milanesa napolitana', 'Ravioles con bolognesa', 'Pizza calabresa'];

let numerosSorteados = [12, 45, 56, 324, 452];
```



Tipos de **datos especiales**

NaN (Not a Number)

Indica que el valor no puede ser parseado como un número.

```
let malaDivision = "35" / 2; // NaN no es un número
```

Null (valor nulo)

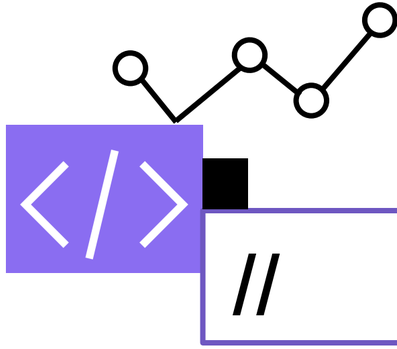
Lo asignamos nosotros para indicar un valor vacío o desconocido.

```
let temperatura = null; // No llegó un dato, algo falló
```

Undefined (valor sin definir)

Las variables tienen un valor indefinido hasta que les asignamos uno.

```
let saludo; // undefined, no tiene valor
saludo = "¡Hola!"; // Ahora si tiene un valor
```



Los **comentarios** son partes de nuestro código **que no se ejecutan**.

Pueden comenzar con dos barras inclinadas **//** o **/* */**

Los usamos para explicar lo que estamos haciendo y **dejar información útil** para nuestro equipo o para nuestro yo del futuro.

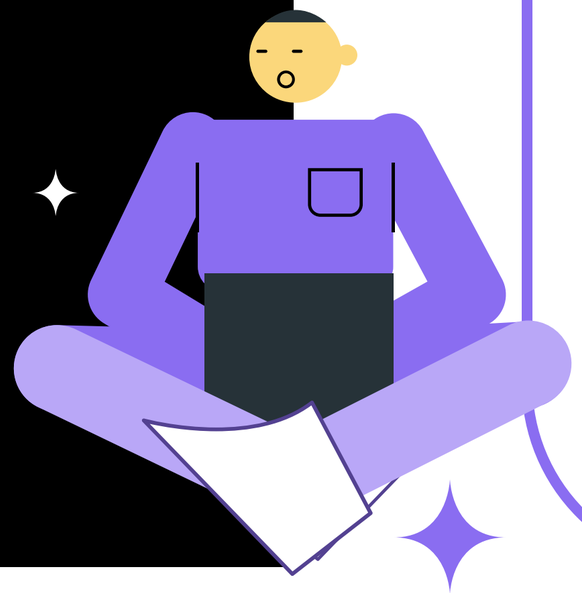
{}

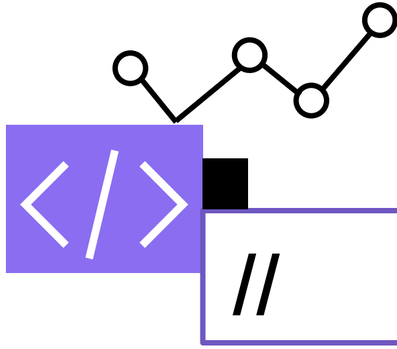
```
// Este es un comentario en línea.

/* Este es un comentario multilínea */
```

04 Operadores

Los **operadores** nos permiten **manipular el valor** de las variables, realizar **operaciones y comparar** sus valores





De asignación

Asignan el valor de la derecha en la variable de la izquierda.

```
let edad = 35; // Asigna el número 35 a edad
```

Aritméticos

Nos permiten hacer operaciones matemáticas, devuelven el resultado de la operación.

```
10 + 15 // Suma → 25
10 - 15 // Resta → -5
10 * 15 // Multiplicación → 150
15 / 10 // División → 1.5
```

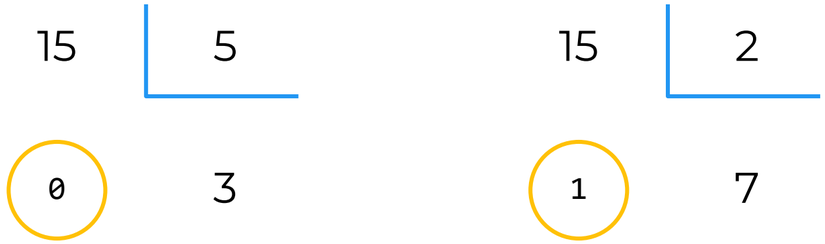
Aritméticos (continuación)

Nos permiten hacer operaciones matemáticas, devuelven el resultado de la operación.

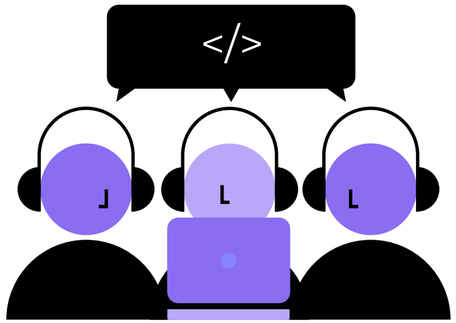
```
15++ // Incremento, es igual a 15 + 1 → 16
15-- // Decremento, es igual a 15 - 1 → 14
```

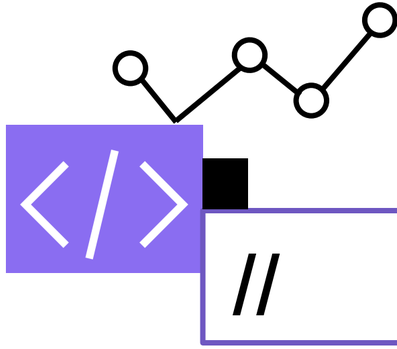
```
15 % 5 // Módulo, el resto de dividir 15 entre 5 → 0
15 % 2 // Módulo, el resto de dividir 15 entre 2 → 1
```

El operador de módulo % nos devuelve el resto de una división.



Los **operadores aritméticos** siempre devolverán el resultado **numérico** de la operación que se esté realizando.





De concatenación

Sirve para unir cadenas de texto. Devuelve otra cadena de texto.

```
let nombre = 'Teodoro';
let apellido = 'García';
let nombreCompleto = nombre + ' ' + apellido;
```

Si mezclamos otros tipos de datos, estos se convierten a cadenas de texto.

```
let fila = 'M';
let asiento = 7;
let ubicacion = fila + asiento; // 'M7' como string
```

DIGITALHOUSE

¡Muchas gracias!