



# JS | Variables, Operadores y Tipos

## Después de esta lección podrás:

1. Comprender y aplicar la sintaxis básica de JavaScript.
2. Aprender qué es una variable de tipo dinámico.
3. Declarar variables en JavaScript.
4. Comprender qué es una cadena en JavaScript y algunos consejos para manipularlos.

## ¿Qué es una variable?

Las variables se utilizan para **almacenar información** para ser manipulada en un programa. Deben llevar un nombre descriptivo, para que nuestros programas puedan ser entendidos por nosotros mismos.

Piensa en las variables como contenedores que tienen información. Su único propósito es **etiquetar** y almacenar **datos** en la memoria para que podamos usarlos en todo nuestro programa.

## Declarando una variable en Javascript

Las variables se declaran con `var` como veremos a continuación, que vamos a crear la variable **name**:

```
var name;
```

Ahora que la variable ya existe, podemos darle un valor para que lo guarde.

```
name = "Pedro";
```

El almacenamiento de un valor en una variable se llama **inicialización** de variable. Puedes hacer la inicialización de la variable en el momento de la creación de la variable o en un momento posterior cuando necesites esa variable.

```
var name = "Pedro"; // En una sola línea hemos creado la variable, y la hemos inicializado
```

Veamos un ejemplo en el que creamos variables y las imprimimos por pantalla, que podremos copiar a un `fichero.js`, para probarlo en nuestro ordenador:

```
var name = "Luis Garcia"; // Creamos una variable y la inicializamos
console.log(name);
var edad; // Creamos una variable sin inicializarla con un valor
console.log(edad); // Imprime por pantalla: undefined --> Porque la variable existe, pero no tiene valor dentro
edad = 33; // Ahora le damos valor
console.log(edad); // Imprime por pantalla: 33
```

Que no nos asuste la palabra `undefined`, simplemente es lo que nos muestra JavaScript cuando una variable todavía no tiene valor dentro, no está definido.

```
1 var name = "Luis Garcia"; // Creamos una variable y la inicializamos
2 console.log(name);
3 var edad; // Creamos una variable sin inicializarla con un valor
4 console.log(edad); // Imprime por pantalla: undefined -> Porque la variable
5 edad = 33; // Ahora le damos valor
6 console.log(edad); // Imprime por pantalla: 33
```

```
node codigo.js
Luis Garcia
undefined
33
```

## Reglas para nombrar variables

A la hora de **nombrar** una variable, debemos ponerle un nombre descriptivo, para que cuando leamos el código, entendamos que valor guarda. Es decir, para crear una variable que guarda el DNI de alguien, un ejemplo sería:

```
var dni = "00000000A";

// Veamos ejemplos de mal nombrado
var manzanas = "00000000B"; // este NO sería un buen nombre de variable
var dddddnnnniiii = "00000000C"; // este NO sería un buen nombre de variable
var cosas_Magicas = "00000000D"; // este NO sería un buen nombre de variable
```

Pero... ¿qué ocurre si mi variable no la puedo nombrar con una palabra? Todos tranquilos, para esto utilizaremos lo que se llama notación **camelCase**. Simplemente bastaría con eliminar los espacios entre palabras, y poner en mayúscula la primera letra: **asíDeFácil** 😊

```
var nombreCompleto = "Jose Francisco Perez Lopez";
```

## Valores "variables"

Las variables se llaman así por algo, porque sus valores se pueden cambiar en cualquier parte del código, veamos un ejemplo:

```
var side = "light side";
console.log(side); // <== "light side"; - valor inicial
side = "dark side";
console.log(side); // <== "dark side" - el valor a cambiado
```

A screenshot of a code editor window titled 'codigo.js'. The editor shows four lines of JavaScript code: 1. 'var side = "light side";', 2. 'console.log(side); // <== "light side"; - valor inicial', 3. 'side = "dark side";', and 4. 'console.log(side); // <== "dark side" - el valor a cambiado'. Below the code editor, there is a terminal panel with tabs for 'COMMENTS', 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is active, showing the command 'node codigo.js' and its output: 'light side' followed by 'dark side' on the next line. The terminal also shows the file path '~/Desktop/js' and system information like '22:18' and '100%' battery.

En el bootcamp explicaremos los distintos **tipos** de datos que pueden almacenar las variables, pero por ahora centrémonos solo en dos: números y textos. Pues bien, una variable puede almacenar diferentes tipos de datos 🤪 (por eso es importante ponerle un nombre descriptivo):

```
var myValue = "cadena";
console.log(myValue); // es un texto: "cadena"
myValue = 5;
console.log(myValue); // es un numero: 5
```



```
JS codigo.js x
Users > marioantoniopolodaza > Desktop > js > JS codigo.js > ...
1 var myValue = "cadena";
2 console.log(myValue); // es un texto: "cadena"
3 myValue = 5;
4 console.log(myValue); // es un numero: 5

COMMENTS PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: zsh
~/Desktop/js
node codigo.js
cadena
5
```

## Operadores

Si estás familiarizado con las matemáticas, el término **operador** te sonará. Cuando estamos haciendo una suma básica, en el ejemplo `5 + 2`, `+` es el operador, y la **suma** es la operación.

En matemáticas, hay una gran cantidad de operadores más allá de la simple suma. En el código (específicamente JavaScript por ahora), también. Algunos simples que te pueden venir a la mente son:

- `+` Suma
- `-` Resta
- `*` Multiplicación
- `/` División

Como en otras ocasiones vamos a comprobar en nuestro `fichero.js` el resultado de operar con ellos. Añadimos dos variables, asignamos un valor y realizamos tantas operaciones como queramos. Siempre dentro del `console.log` que nos imprimirá en pantalla la respuesta.

```
var num1 = 5;
var num2 = 2
console.log(num1 + num2); // Imprime 7
console.log(num1 - num2); // Imprime 3
console.log(num1 * num2); // Imprime 10
console.log(num1 / num2); // Imprime 2.5
```



## Operador de asignación

Anteriormente vimos como asignar variables. Cuando hacemos `variable = expresión`, `=` es el operador de asignación porque está **asignando** un valor a una variable.

```
var jediLevel = 9; // Asignacion de valor 9
jediLevel = jediLevel + 1; // Asignacion de la suma de 1 al nivel (9 + 1 --> 10)
console.log(jediLevel);
```

Como hemos visto, podemos asignar valores directamente, o asignar el resultado de una operación (una suma). A continuación detallaremos en una tabla la manera de **operar y asignar**, a la vez:

**Tabla de asignaciones**

 Name	 Operator	 Equivalent
<u>Asignación</u>	X=Y	N/A
<u>Suma + Asignación</u>	X+=Y	X=X+Y
<u>Restar + Asignación</u>	X-=Y	X=X-Y
<u>Multiplicación + Asignación</u>	X*=Y	X=X*Y
<u>División + Asignación</u>	X/=Y	X=X/Y

Y para ponerlo en práctica, veamos un bloque de código con diferentes ejemplos (probar a copiarlo y ejecutarlo en vuestro `fichero.js`):

```
var num1 = 5;
var num2 = 10;

// asignación
num1 = num2;
console.log(num1); // num1 pasa a valer 10

// asignación y suma (num1 ya no vale el 5 inicial, recordar que ahora vale 10)
num1 += num2;
console.log(num1); // num1 pasa a ser 20

// asignación y resta
num1 -= num2;
console.log(num1); // num1 pasa a ser 10
```

```
// asignación y multiplicación
num1 *= num2;
console.log(num1); // num1 pasa a ser 100

// asignación y división
num1 /= num2;
console.log(num1); // num1 pasa a ser 10
```










También podemos **mezclar** varios operadores, es importante que tengamos en cuenta en qué orden se van a ejecutar, por ejemplo si hacemos lo siguiente:

```
var jediLevel = 10 - 50 * 100;
```

¿Qué se ejecutará antes? ¿La resta o la multiplicación? En general siempre se comportarán como en matemáticas, es decir que en este caso primero se ejecutará la multiplicación. Cuando tengas dudas puedes consultar la siguiente tabla donde se muestran los operadores por su orden de precedencia.

Como puedes observar los paréntesis y su contenido siempre serán lo primero que se resuelva y las restas lo último, **igual que en matemáticas**.

### Precedencia del operador

 Name 1	 Operator	 Name
1		Parentesis
2		Exponente
3		Multiplicación
4		División
5		Suma
6		Resta

## Tipos de datos

Existen diferentes **tipos** de datos en JavaScript, para representar números, textos, etc. Por ahora solo nos centraremos en tres (números, textos y valores lógicos [verdadero/falso]):

## Tipo Number

El tipo numérico nos permite definir números, fácil ¿no? Con los número podremos hacer las operaciones matemáticas que antes mencionamos:

```
var numeroEntero = 8;
var numeroDecimal = 3.55;
var numeroNegativo = -11;
var resultado = 6.22 + 7.11;
```

## Tipo String

Con el tipo string, podremos definir cadenas de texto, que son simplemente una secuencia de caracteres. Las declaramos entre comillas siempre, ya sean dobles `""` o simples `' '`:

```
var frase = "En un lugar de la Mancha, de cuyo nombre no quiero acordarme...";
var otraFrase = 'no ha mucho tiempo que vivía un hidalgo de los de lanza...';
```

Cada carácter puede ser una letra, un número, puntuación. Incluso podremos unir cadenas de texto unas con otras, mediante `+`, no confundir con el operador de suma de números:

```
var value = "Hola me llamo Jose y tengo 2 hijas";
console.log(value);
// Imprimirá: Hola me llamo Jose y tengo 2 hijas

var nombreHijas = ': Maria y Ana';
console.log(value + nombreHijas);
// Imprimirá: Hola me llamo Jose y tengo 2 hijas: Maria y Ana
```

Podemos acceder a los caracteres dentro de las cadenas con su posición numérica, teniendo en cuenta que la primera casilla es 0. Para ello usaremos los corchetes `[]`:

```
var jediName = "Luke";
console.log(jediName[0]); // => L
console.log(jediName[3]); // => e
console.log(jediName[8]); // => undefined
console.log(jediName[-1]); // => undefined
```



## Tipo Boolean

Este es el tipo de valor lógico verdadero o falso, en la próxima sección lo veremos en profundidad, por lo que aquí no haremos spoiler 🚧

## Resumen

En esta lección aprendimos que es Javascript y parte de su sintaxis básica.

Ahora ya sabes cómo se comportan las variables en Javascript, como crearlas y asignarles valores. Además, aprendimos operadores para manipularlas y realizar operaciones básicas.

Finalmente vimos algunos tipos primitivos, como números y cadenas de texto, así como algunos trucos adicionales que seguramente serán útiles en el futuro.

## Recursos Extra

- [MDN](#) - This will be your main resource for any Javascript search. It has tutorials, guides and tools. Is the formal Javascript Documentation.
- [Variables Cheat Sheet](#)
- [Understanding Javascript Variables](#)

## Seguimos aprendiendo en:

🌟 [JS | Funciones](#)