**Logotipo

Descripción generada automáticamenteCurso Javascript**

Contenido

[Variables 3](#_Toc81184530)

[String 4](#_Toc81184531)

[Métodos 5](#_Toc81184532)

[Length 5](#_Toc81184533)

[IndexOf 5](#_Toc81184534)

[Includes 5](#_Toc81184535)

[Concatenar cadenas 5](#_Toc81184536)

[Template strings 5](#_Toc81184537)

[Trim 5](#_Toc81184538)

[Replace 6](#_Toc81184539)

[Slice 6](#_Toc81184540)

[Substring 6](#_Toc81184541)

[CharAt 6](#_Toc81184542)

[Repeat 6](#_Toc81184543)

[Split 6](#_Toc81184544)

[toUpperCase 7](#_Toc81184545)

[toLowerCase 7](#_Toc81184546)

[toString 7](#_Toc81184547)

[Number 7](#_Toc81184548)

[Operadores matemáticos 7](#_Toc81184549)

[Operadores de incremento y decremento 8](#_Toc81184550)

[Operadores de Asignación 8](#_Toc81184551)

[Operadores de Comparación 9](#_Toc81184552)

# Variables

Hay 3 formas de declarar una variable:

* var
* let
* const

Reglas que tiene JavaScript para nombrar variables:

* Los nombres de las variables **no pueden contener espacios**.
* Los nombres de las variables deben comenzar con una letra, un guión bajo ( \_ ) o un signo de dólar ( $ ).
* Los nombres de las variables solo pueden contener letras, números, guiones bajos o signos de dólar.
* Los nombres de las variables **distinguen entre mayúsculas y minúsculas**.
* Es posible que algunas palabras no se utilicen como nombres de variables porque tienen otros significados dentro de JavaScript.Abajo aparece la tabla de palabras reservadas.
* **No use nombres demasiado cortos**. Los nombres simples de una letra o los nombres que no tienen sentido no son una buena opción al nombrar variables.
* **Use más de una palabra para nombrar su variable**. Esto asegurará que su nombre de variable sea preciso.
* **Cuando use más de una palabra en los nombres de sus variables, coloque siempre el adjetivo a la izquierda**. Por ejemplo, esto es correcto: var greenGrass.
* **Elija un estilo para los nombres con más de una palabra y cúmplalo**. Las dos formas más comunes de unir palabras para crear un nombre son camelCase y el uso de un guión bajo ( \_ ). JavaScript es flexible, cualquiera de los dos métodos funciona.

Tabla

Descripción generada automáticamente

## String

Los strings o cadenas de texto bueno representan un texto que se puede asignar a una variable. Cosas como el nombre de un cliente, el nombre de un producto, la categoría. Las cadenas en JavaScript están contenidas dentro de un par de comillas simples '' o comillas dobles "".

'This is a string. 👏';  
"This is the 2nd string. 💁";

Existen 3 formas de declarar una variable String:

* const producto = 'Monitor 20 pulgadas"';
* const producto2 = String("Monitor 24 pulgadas");
* const producto3 = new String("Monitor 27 pulgadas");

## Métodos

Las cadenas tienen sus propias variables y funciones integradas, también conocidas como propiedades y métodos.

### Length

Conocer la cantidad de letras de la cadena de texto

console.log(producto.length); //19

### IndexOf

Devuelve el primer índice en el que se puede encontrar un elemento dado en la matriz

console.log(producto.indexOf('pulgadas')); //11

### Includes

Determina si una matriz incluye un cierto valor entre sus entradas devolviendo verdadero o falso según corresponda.

console.log(producto.includes('pulgadas')); //true

### Concatenar cadenas

const producto = "Monitor 20 pulgadas";

const precio = "30 USD";

console.log(producto.concat(precio)); //Monitor 20 pulgadas 30 USD

  console.log(producto + "Con un precio de: " + precio);

//Monitor 20 pulgadasCon un precio de: 30 USD

### Template strings

Crea una plantilla literal envolviendo su texto con comillas invertidas ( `) de la siguiente manera:

console.log(`El Producto ${producto} tiene un precio de ${precio}`);

//El Producto Monitor 20 pulgadas tiene un precio de 30 USD

y obtienes las siguientes características

* + **Una cadena de varias líneas**: una cadena que puede abarcar varias líneas.
  + **Formato de cadena**: la capacidad de sustituir parte de la cadena por los valores de variables o expresiones. Esta característica también se llama interpolación de cadenas.
  + **Escape de HTML**: la capacidad de transformar una cadena para que sea seguro incluirla en HTML.

### Trim

Se utiliza para eliminar los espacios en blanco del principio y el final de una cadena

console.log(producto.trim());

### Replace

Busca una cadena para un valor especificado, o una expresión regular, y devuelve una nueva cadena donde se reemplazan los valores especificados.

console.log(producto.replace('Monitor','Monitor Curvo')); //Monitor Curvo 20 pulgadas

### Slice

Extrae una sección de una cadena y la devuelve como una nueva cadena, sin modificar la cadena original.

console.log(producto.slice(0,10)); //Monitor 20

### Substring

Método extrae caracteres, entre índices (posiciones), de una cadena y devuelve la subcadena.

console.log(producto.substring(0,10)); //Monitor 20

Diferencia entre slice y substring

Si le pasas un número mayor de inicio y la extensión slice no hace nada, pero substring lo modifica y trata de hacer algo.

console.log(producto.slice(2,1));//No funciona

console.log(producto.substring(1,0)); //Lo modifica a (0,1) //Resultado M

### CharAt

El método devuelve el carácter en un índice especificado en una cadena.

console.log(usuario.charAt(0)); //Resultado M

### Repeat

Construye y devuelve una nueva cadena que contiene el número especificado de copias de la cadena en la que se llamó, concatenadas juntas.

const texto= ' en Promoción'.repeat(3);

console.log(texto);  //en Promoción en Promoción en Promoción

### Split

Divide una cadena en una lista ordenada de subcadenas, coloca estas subcadenas en una matriz y devuelve la matriz. La división se realiza buscando un patrón; donde el patrón se proporciona como primer parámetro en la llamada al método.

const actividad="Estoy aprendiendo Javascript Moderno";

console.log(actividad.split(" "));

[ 'Estoy', 'aprendiendo', 'Javascript', 'Moderno' ]

### toUpperCase

El método devuelve el valor de la cadena de llamada convertido a mayúsculas (el valor se convertirá en una cadena si no lo es).

console.log(producto.toUpperCase());//MONITOR 20 PULGADAS

### toLowerCase

Devuelve el valor de la cadena de llamada convertido a minúsculas.

console.log(producto.toLowerCase());//monitor 20 pulgadas

### toString

El método devuelve una cadena que representa el objeto.

const precio=300;

console.log(typeof precio); //number

console.log(typeof precio.toString()); //string

## Number

Son valores que se pueden utilizar en operaciones matemáticas. No necesita ninguna sintaxis especial para los números.

Existen 3 formas de declarar una variable Number:

* const numero1 = 30;
* const numero2 = Number(45);
* const numero3 = new Number(80);

### Operadores matemáticos

const numero1 = 30;

const numero2 = 20;

//Suma

resultado = numero1 + numero2;

console.log(resultado); //50

//Resta

resultado = numero1 - numero2;

console.log(resultado); //10

//Multiplicacion

resultado = numero1 \* numero2;

console.log(resultado); //600

//Division

resultado = numero1 / numero2;

console.log(resultado); //1.5

//Modulo

resultado = numero1 % numero2;

console.log(resultado); //10

### Operadores de incremento y decremento

let a = 3,

    b = 2;

let z = a + b;

//Incremento

//Pre-incremento (el operador ++ antes de la variable)

z = ++a;

console.log(a);

console.log(z);

//Post-incremento  (el operador ++ despues de la variable)

z = b++;

console.log(b);

console.log(z);

//Decremento

//Predecremento

z = --a;

console.log(a);

console.log(z);

//Postdecremento

z=b--;

console.log(b);

console.log(z);

### Operadores de Asignación

let a = 1;

//Suma

a += 3; //a =a+3

console.log(a); //4

//Resta

a -= 2; //a=a-2

console.log(a); //-1

//Multiplicacion

a \*= 3; //a=a\*3

console.log(a); //3

//Division

a /= 1; //a=a/1

console.log(a); //1

//Modulo

a %= 5; //a=a%5

console.log(a); //0

//Potencia

a \*\*= 5; //a=a\*\*5

console.log(a); //1

### Operadores de Comparación

let a = 3,

  b = 2,

  c = "3";

let z = a == b; //revisa el valor sin importar el tipo

console.log(z); false

z = a != c; //revisa el valor sin importar el tipo

console.log(z); false

//Comparador estricto

z = a === c; //revisa los valores pero también los tipos

console.log(z); false

z = a !== c; //revisa los valores pero también los tipos

console.log(z); true

**spread operator and rest parameter**