Variables

¿Qué es una variable?

Una **variable** es un contenedor que almacena un valor, similar a las variables en matemáticas y nos permiten reutilizar y manejar datos de forma eficiente.

Declaración de variables

Para declarar variables en JavaScript, se utilizan las palabras clave `var`, `let` y `const`, cada una con características específicas:

1. `var`:

- Se utiliza para declarar variables que pueden ser reasignadas.
- Tienen un alcance **funcional**, lo que significa que son accesibles dentro de la función en la que se declaran, o globalmente si se declaran fuera de cualquier función.
- Ejemplo:

```javascript
var nombre = "Juan";
nombre = "Pedro"; // Reasignación válida
```

2. `let`:
   - Permite declarar variables que pueden ser reasignadas, pero tienen un alcance de **bloque** (es decir, son accesibles solo dentro del bloque donde se declaran).
   - Es preferible usar `let` para la mayoría de las declaraciones de variables debido a su alcance más predecible.
   - Ejemplo:

```javascript
let edad = 25;
if (true) {
    let edad = 30; // Alcance limitado al bloque
    console.log(edad); // Imprime 30
}
console.log(edad); // Imprime 25
```

3. `const`:
   - Se utiliza para declarar constantes, es decir, variables cuyo valor no puede ser reasignado después de su declaración.
   - También tiene un alcance de **bloque**.
   - Ejemplo:

```javascript
const PI = 3.14;
PI = 3.14159; // Esto generaría un error porque PI no puede ser reasignado
```

> ⚠ **Warning**
>
> `var` Se considera obsoleto y está en desuso debido a su alcance menos predecible. Es recomentable usar siempre `let` o `const`

Reglas para nombres de variables:

1. Deben empezar con una letra (preferiblemente minúscula), `$` o `_`
2. No pueden contener espacios.

> 🖉 **Note**
> Es una buena práctica utilizar la notación **camelCase** para separar palabras. Ejemplo: `miVariable`.

Tipos de datos

Tipos nativos:

JavaScript tiene varios tipos de datos fundamentales:

Números

Representan tanto enteros como decimales.

- **Enteros (`int`)**:

```
let numero = 10;
```

- **Decimales (`float`)**:

```
let decimal = 10.5;
```

> ⚠ **Attention**
>
> En JavaScript los decimales deben indicarse con el punto `.` y no con la coma `,`

Cadenas (Strings)

Secuencias de caracteres encerradas entre comillas simples (`' '`), dobles (`" "`), o backticks (`` ` ``).

- **Ejemplo**:

```
let saludo = 'Hola';
let nombre = "Mundo";
let mensaje = `Hola ${nombre}`;
```

Booleanos

Pueden tomar los valores `true` o `false`.

- **Ejemplo**:

```
let isActive = true;
let isComplete = false;
```

Nulos (`null`)

Representan la ausencia intencionada de un valor.

- **Ejemplo**:

```
let resultado = null;
```

Indefinidos (`undefined`)

Se asigna automáticamente a variables que han sido declaradas pero no inicializadas.

- **Ejemplo**:

```
let sinValor;
console.log(sinValor); // Imprime "undefined"
```

Tipos complejos:

Objetos

Colecciones de propiedades y métodos. Los objetos se crean utilizando llaves `{}`.

- **Ejemplo**:

```
let persona = {
   nombre: "Juan",
   edad: 30,
   saludo: function() {
      return `Hola, soy ${this.nombre}`;
   }
};
```

Arrays

Listas ordenadas de valores, que pueden ser de cualquier tipo de dato.

- **Ejemplo**:

```
let frutas = ["manzana", "naranja", "plátano"];
console.log(frutas[0]); // Imprime "manzana"
```

Conversión de tipos

JavaScript permite la conversión entre tipos de datos:

- **Conversión implícita**: Ocurre automáticamente cuando se utilizan diferentes tipos en una expresión.
  - Ejemplo:

    ```
    let resultado = '5' + 2; // '52' (cadena)
    let suma = '5' - 2; // 3 (número)
    ```

- **Conversión explícita**: Se puede realizar mediante funciones como `Number()`, `String()`, y `Boolean()`.
  - Ejemplo:

    ```
    let numeroComoCadena = "123";
    let numeroConvertido = Number(numeroComoCadena); // Convierte a número
    ```

Operadores en JavaScript

Operadores básicos

Los operadores en JavaScript son símbolos que realizan operaciones sobre variables y valores. A continuación, se detallan los tipos de operadores más comunes:

Operadores Aritméticos

Estos operadores se utilizan para realizar cálculos matemáticos.

- **Suma (`+`)**:

  ```
  let suma = 5 + 3; // Resultado: 8
  ```

- **Resta (`-`)**:

  ```
  let resta = 5 - 3; // Resultado: 2
  ```

- **Multiplicación (`*`)**:

  ```
  let multiplicacion = 5 * 3; // Resultado: 15
  ```

- **División (`/`)**:

```javascript
let division = 15 / 3; // Resultado: 5
```

- **Módulo (resto de la división, `%`)**:

```javascript
let modulo = 10 % 3; // Resultado: 1
```

Operadores de Asignación

Se utilizan para asignar valores a las variables.

- **Asignación simple (`=`)**:

```javascript
let x = 10;
```

- **Resta y asignación (`-=`)**:

```javascript
let y = 20;
y -= 5; // Equivalente a y = y - 5; ahora y es 15
```

- **Multiplicación y asignación** (`*=`):

```javascript
let z = 10;
z *= 2; // Equivalente a z = z * 2; ahora z es 20
```

- **División y asignación (`/=`)**:

```javascript
let a = 40;
a /= 4; // Equivalente a a = a / 4; ahora a es 10
```

- **Módulo y asignación (`%=`)**:

```javascript
let b = 10;
b %= 3; // Equivalente a b = b % 3; ahora b es 1 (residuo de la división)
```

Operadores de Comparación

Estos operadores se utilizan para comparar dos valores y devuelven un valor booleano (`true` o `false`).

- **Igualdad (`==`)**: Compara los valores, sin tener en cuenta el tipo (realiza conversión de tipos)

```javascript
console.log(5 == '5'); // true (igualdad débil)
```

- **Igualdad estricta (`===`)**: Compara tanto el valor como el tipo.

```javascript
console.log(5 === '5'); // false (igualdad estricta)
```

- **Desigualdad (`!=`)**: Comprueba si son diferentes, sin tener en cuenta el tipo.
- **Desigualdad estricta (`!==`)**: Compara tanto el valor como el tipo, pero devuelve `true` si son diferentes.

Operadores Lógicos

Los operadores lógicos se utilizan para combinar expresiones booleanas y evaluar condiciones. Los más comunes son:

- **Y lógico (`&&`)**: Devuelve `true` si ambas expresiones son verdaderas.

```
let a = true;
let b = false;
console.log(a && b); // false
```

- **O lógico (`||`)**: Devuelve `true` si al menos una de las expresiones es verdadera.

```
console.log(a || b); // true
```

- **No lógico (`!`)**: Invierte el valor de la expresión booleana.

```
console.log(!a); // false
```