UNIDAD 2

//comentario de una sola línea

/* */ comentario de varias líneas

var,let,const son formas de declarar variables aunque let ya no se usa

```
 */
```

```
let uno= 1;
//declarar string
/* se pueden declarar con comillas simples o dobles*/
let nombre = "Joan";
let apellido = 'Melsion';
let nombreCompleto = "profe";
let comillas = "Texto 'entrecomillado'";
//para usar comillas dobles:
let comillas = "Texto /"entrecomillado/"";

//declarar numeros: no hacen falta comillas
```

DECLARAR NÚMEROS: no hacen falta comillas

```
let edad = 26; // esto es un número y puede ser positivo o negativo
let anyos = "26"; // esto es un String
console.log(typeof edad);
console.log(typeof anyos);

let masEdad = -66;
let precio = 21.99;
console.log(typeof precio);//nos dice el tipo de variable que es

let aprobados = Infinity;
let suspendidos = -Infinity;

console.log(typeof aprobados); //también tipo número
console.log(masEdad + aprobados);//sigue siendo infinity
```

DECLARAR NULL:

Indica algo que está definido pero que su valor está vacío o se le ha dado un valor nulo

```
let valor = null;
console.log(typeof valor);//da tipo object
```

UNDEFINED:

Algo que ni siquiera ha sido definido

```
let algo;
console.log(algo); // da tipo indefinido
```

NaN O NOR A NUMBER:

Cuando intentamos usar algo como un número sin serlo

```
let algo = "algo";
console.log(algo/2);
```

TYPEOF:

Para saber el tipo que está representando una variable

```
console.log(typeof edad);
// delete funciona en los objetos

let persona={
    nombre : "ester",
    edad: 22
}
console.log(persona.edad);
delete persona.edad;// borra el objeto
console.log(persona.edad);
```

BOOLEANO:

Se utilizan para representar valores true o false

```
let messi = true;
let penaldo = false;//valores falsy
```

Pueden ser considerados false: null, string vacío “”, undefined, 0 y Nan

OPERADORES ARITMÉTICOS:

Se utilizan para operaciones matemáticas:

// +

// -

// *

// /

// % (módulo de una división)

// ++

```
let a = 4;
a++;//aquí suma 1
console.log(++a); //suma 1 aquí
```

// -- decremento

// += suma y asigna (a +=b // a = a + b)

// -= resta y asigna (a -= b // a = a – b)

// *= multiplica y asigna (a *= b // a = a * b)

// /= divide y asigna (a /= b // a = a / b)

// %= modula y asigna (a %= b // a = a % b)

// cambio de signo (a = 5, b = -a, b = -5)

CONVERSIÓN DE TIPOS:

```
let a = "5";
let b = "6";
console.log(typeof a, typeof b); // dice que es String

let A = parseInt(a);//convertimos a en entero desde String
let B = parseInt(b);
console.log(A + b);//concatena los valores
console.log(A + B);// suma los valores /// tambien se puede poner como cgl (parseInt(a)+parseInt(b))
// al hacer formularios da los tipos siempre de String, por lo que JS sumará los valores

let c = "4.22"; // al hacer un cgl lo convierte en entero por lo que se pierde el decimal
console.log(parseFloat (c)); // lo convierte en número decimal

console.log(typeof a, typeof b);

let A = +a; // poner el signo más delante lo convierte en entero desde String
let B = +b;// igual con este
let C = +c;// aunque sea decimal también lo convierte en número
console.log(+a + +b);//también funciona, pero si no ponemos los ewspacios no le gusta
```

OPERADORES DE COMPARACIÓN:

Comparar 2 valores y devolver un resultado booleano:

// < mayor que

```
console.log(5 > 4); // da true
console.log(5 < 4); // da false
```

// >= mayor o igual que

```
console.log(5 >= 4); //da true
console.log(3 >= 3); // da false
```

// menor o igual que

```
console.log(5 <= 4); // da false
console.log(5 <= 5); // da true
```

// == igual que (= solo asigna valor, para comparación poner ==)

```
console.log(5 == 4); // da false
let a = 4;
console.log(a == 4);// da igual y si ponemos en un String compara valor así que daría igual
```

// != distinto que

```
let a = 5;
let b = "4";
console.log(a != b); //true
```

// === igual en valor y tipo que

```
let a = 5;
let b = "5";
console.log(a === b);//da false porque son iguales en valor pero no en tipo
```

// !== distinto en valor o tipo que (basta que uno sea distinto)

```
console.log(a !== b); // da true porque son distintos en tipo
```

OPERADORES BOOLEANOS

Permiten comparar expresiones booleanas con las que se construyen condiciones que se pueden aplicar car en funciones bucles...

// && AND solo devuelve true cuando ambos operadores lo son

True && true -> true

True && false -> false

False && true -> false

False && false -> false

// || OR devuelve true si cualquiera de ambos operadores lo es

True || true -> true

True || false -> true

False || true -> true

False || false -> false

```
let a = true;
let b = false;
console.log(a || b); // dan todas true a menos que a y b sean false
// si quitamos una barra la comparación es 0 o 1

let cadena = "xsd"; // para trabajar bien con una cadena, compararla siempre con null primero y si tiene longitud:
if(cadena != null && cadena.length > 0) {
    console.log(cadena);
}else{
    console.log("La cadena está vacía"); // si solo ponemos un signo & o | solo miraría la primera condición
}
```

// ! NOT devuelve lo contrario al operando

```
let a = true;
let b = false;
```

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