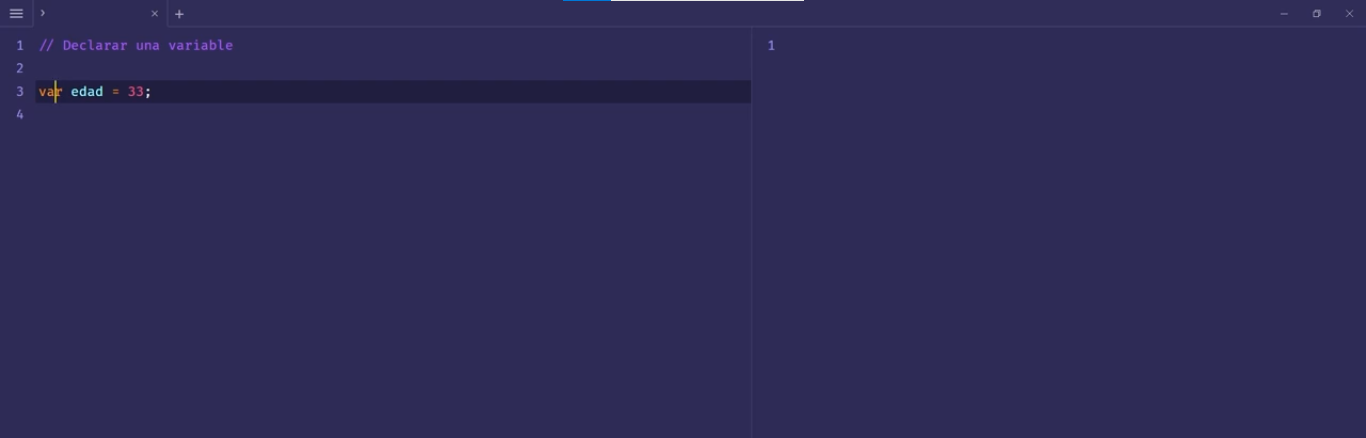
**Variables y datos conceptos**:

**Variable concepto**: espacio de memoria que sirve para guardar un dato –sería una caja donde se almacena ese dato-

**Datos concepto:** Con JS trabajamos con **distintos tipos de información** –numeros, fechas, etc- ese tipo de información **se llama datos**!

**Variable:** cajón dentro de lo que guardas ropas, cada cajón tiene etiquetas con nombres ej pantalón medias, etc.-

**Declarar un variable**: poner una etiqueta a la caja para saber que tipo de ropas vamos a poner

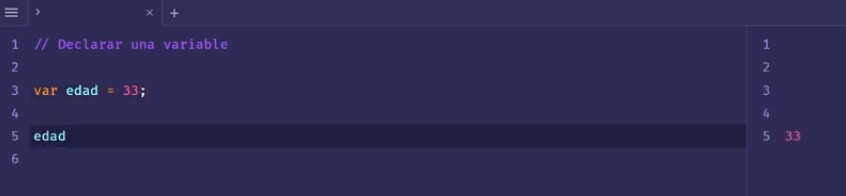


En este ejemplo **“crea una variable var -espacio de memoria- llamado edad y guarda el dato 33 adentro!”**

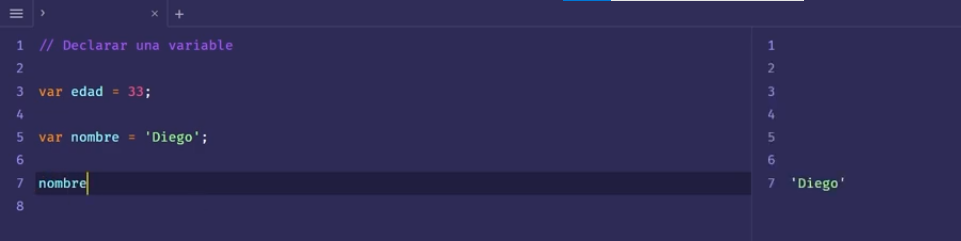
**var** –tipo de variable-

**edad** –nombre de la variable-

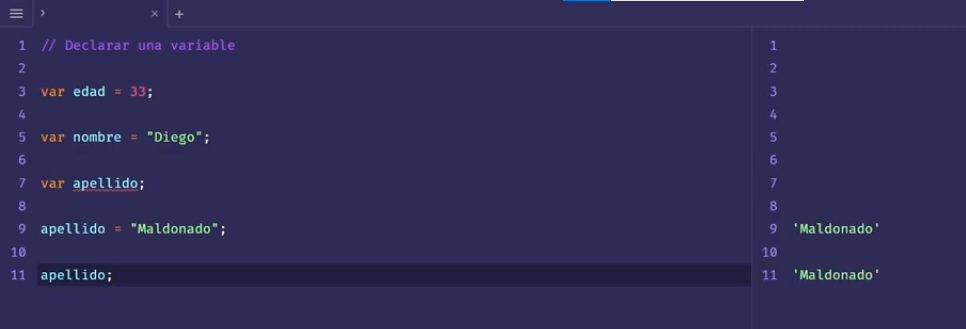
**33** –tipo de dato-



Si escribimos **edad**  a la derecha pone el dato **33**

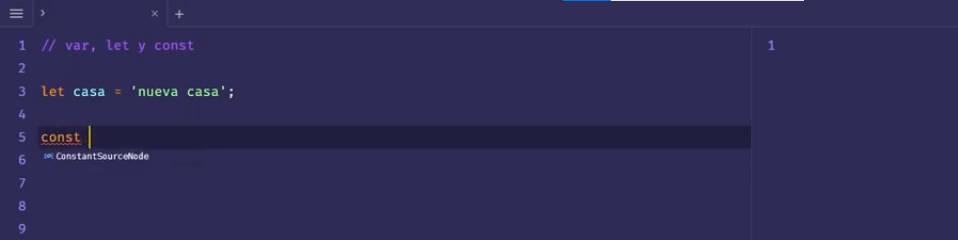


Si ponemos una variable que se llame nombre con el dato **Diego.** Al escribir la etiqueta **nombre** va a salir a la derecha el dato **Diego**

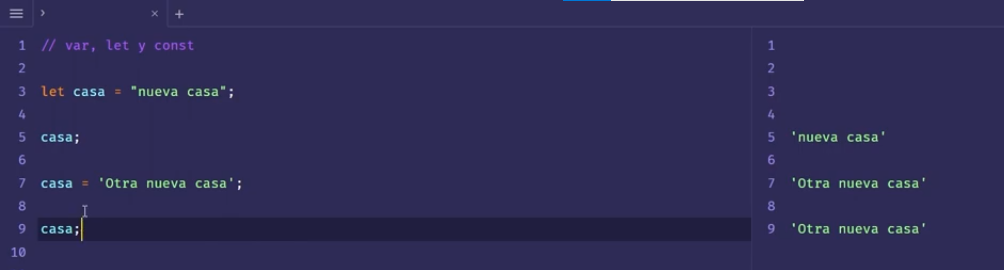
****

También podemos declarar una variable – en este caso **var apellido**- sin ningún valor. Y luego asignar el valor mas abajo **apellido= “Maldonado”;**

**Diferencias entre variables var, let y const**:



La variable **var no se utiliza más**. En vez de dicha variable ahora utilizamos la variable **let!**

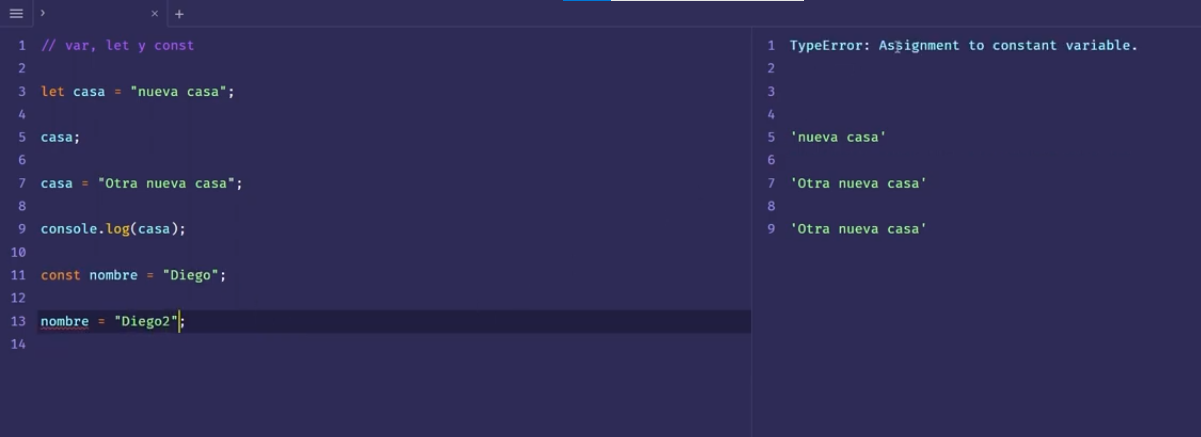
****

La variable **let** como es **no es constante** en este caso le asignamos valor **=”nueva casa”;** colocamos más abajo **casa;** y nos da como resultado a la derecha **“nueva casa”**

Si luego le asignamos a la variable **let** otro valor **=”otra nueva casa”;** mas abajo ponemos el valor **casa;** nos va a dar como resultado **“Otra nueva casa”**

**Esto lo permite hacer la variable let porque NO ES CONSTANTE**: es decir asignar nuevos valores como en nuestro ejemplo **“Otra nueva casa”**

**Diferencia entre variable let y const**:

****

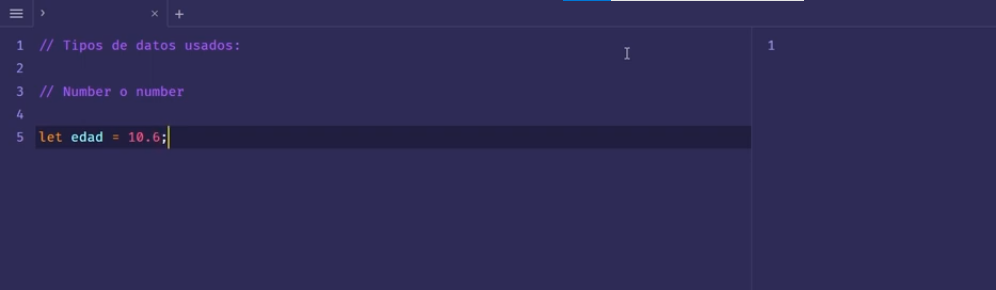
Diferente es con la variable **const** que como su nombre lo dice es constante. No podemos asignar un valor y luego asignar otro distinto en dicho ejemplo: **const nombre = “Diego”**

y luego **nombre = “Diego2”**  a la derecha nos pondrá **TypeError: Asigment to constant variable.**

**Tipos de datos más usados**:

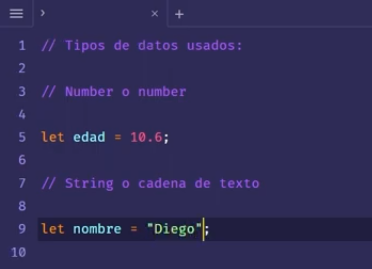
1. **Numeros –pueden ser enteros o con coma-**:

va el número sin comillas

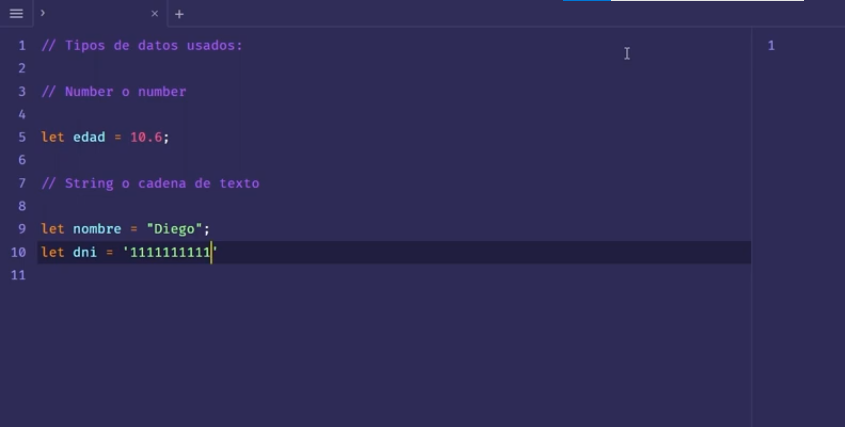


1. **String o cadena de texto:**

**siempre tiene que ir entre comillas dobles o simples!**

****

**Puedo poner números en string o cadena de texto poniendo comillas simples o dobles pero va a pasar por un valor de texto no de número como en el ejemplo anterior**

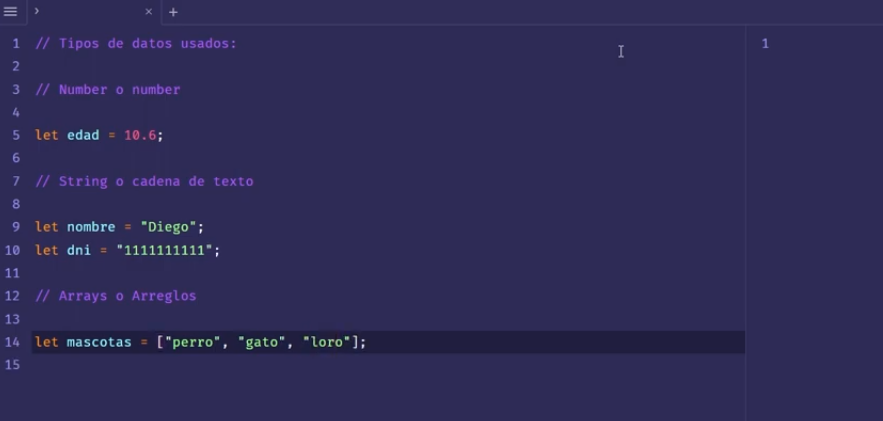
****

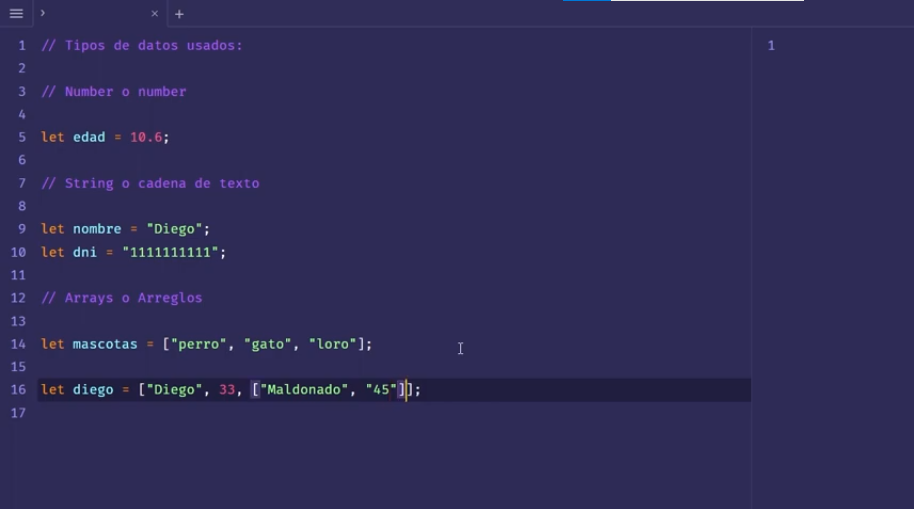
1. **Arrais o arreglos**:

**Colecciona distintos valores, podes guardar distintos tipos de datos en una especie de lista.**

**Es como una caja que contiene muchas cajas adentro**

**EL ARRAIS SIEMPRE VA ENTRE CORCHETES Y PUEDO GUARDAR CUALQUIER TIPO DE DATOS PERO DEBEN SER SEPARADOS POR UNA COMA**

****

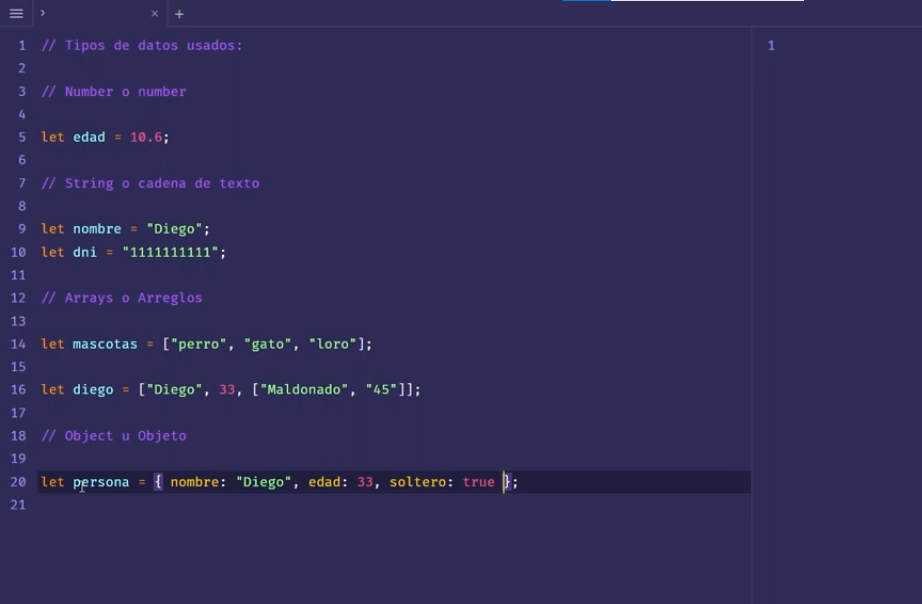
****

En la línea 16 se puede observar dentro del arrais que está entre corchetes los valores: string –“Diego”-, number -33- y otro arrais con más valores dentro –[‘Maldonado’, ‘45’]

1. **Objet u Objeto**:

Es una colección de pares. Cada valor que se guarde dentro va a estar asociado a otro valor y

**Se escribe entre llaves**  { } por un lado la propiedad o llave y por el otro el valor de esa llave



En este caso la propiedad es nombre: y el valor es ‘Diego’

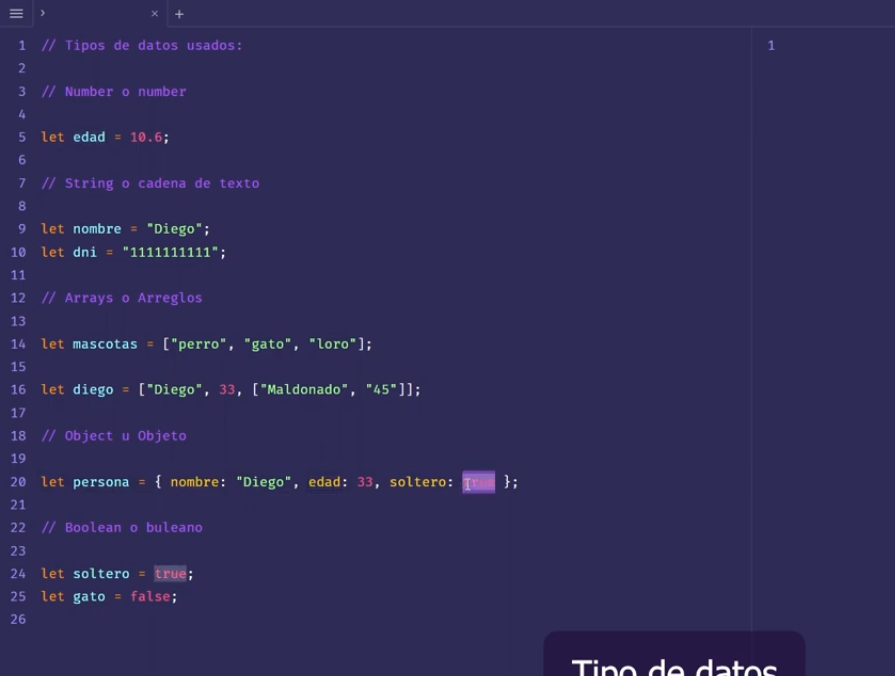
Propiedad es edad: y valor 33

Propiedad es soltero y el valor es true

Y se separa por comas

1. **Boolean o buleano**:

Define para saber si algo es verdadero o falso, sirve para tomar decisiones



Es verdadero o falso

1. **Null o nulo**:

****

Es para dar valor nulo

1. **Undefined o indefinido**:

Es una variable que fue declarada pero no tiene un valor asignado



Da como resultado undefined

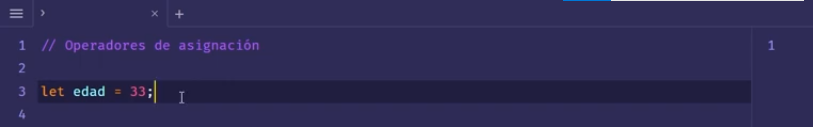
**Operadores y definiciones de expresión**:

Los operadores son símbolos que representan acciones específicas, es decir asignar, comparar, hacer cuentas, etc

**Operadores de asignación**:

Let edad = 33

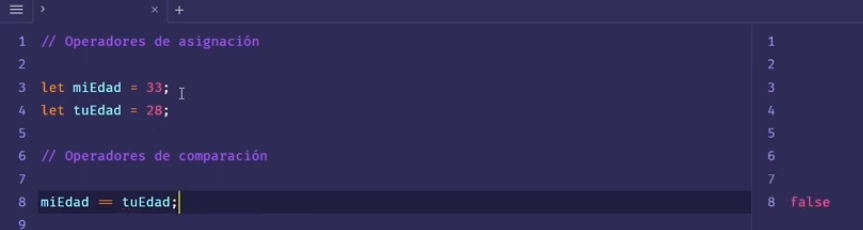
El **=** es el operador de asignación



**Operadores de comparación**:

1. **Operación de comparación simple**:

Sirven para comparar datos



Se utiliza el doble igual en los operadores de comparación ==

En este caso da false –es decir no- porque miEdad y tuEdad no tienen los mismos valores

También tenemos operadores de comparación mayor y menor

En este caso línea 8 da true porque 33 es mayor a 28



Comparadores de desigualdad. En el mismo ejemplo preguntamos si miEdad es diferente a tuEdad va a dar true



Desigualdad estricta igual que igualdad estricta pero con desigualdad no solo compara los valores si no que también compara que sean string o de numero

1. **Operación de comparación estricta**:

Compara el valor y tipo de dato y se utilizan 3 iguales ===



Aca da como resultado el buleano false porque en la línea 3 el valor es numero y en la línea 4 el valor es string



1. **Operadores aritméticos**:

+-\*/ %sirven para numero sumar restar multiplicar y dividir

**Resta**



**División**

****

**Multiplicación**

****

**Suma**

****

**Módulo o porcentaje**

****

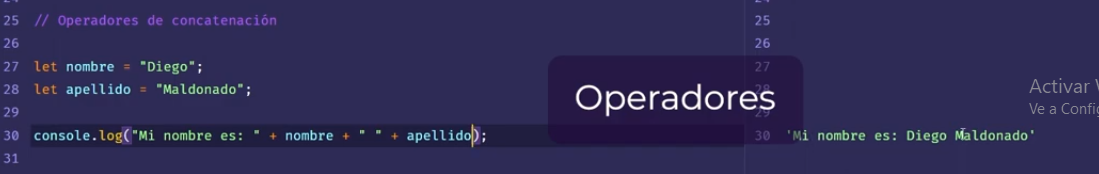
1. **Operadores lógicos**:

**&&** y o AND

**||** o o OR

Sirve para comparar dos cosas

1. **Operadores de concatenación**:



Concatena los dos valores diferentes

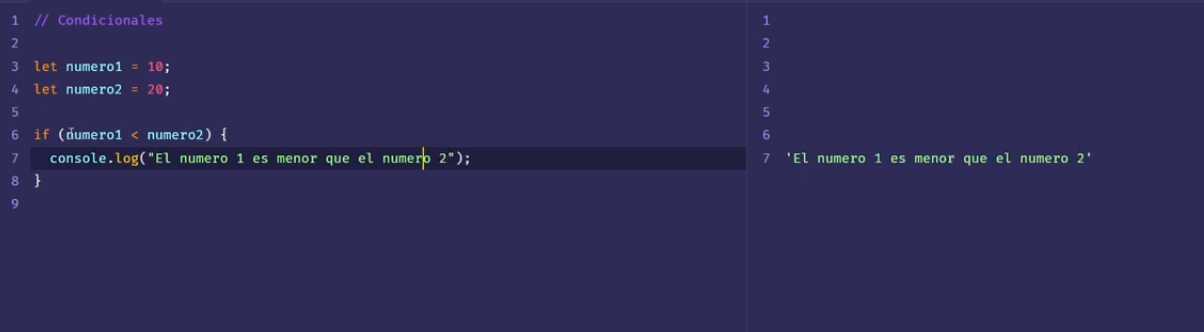
Notese que suma un espacio vacío entre los dos +:  **+” “+**

**Condicionales if**:

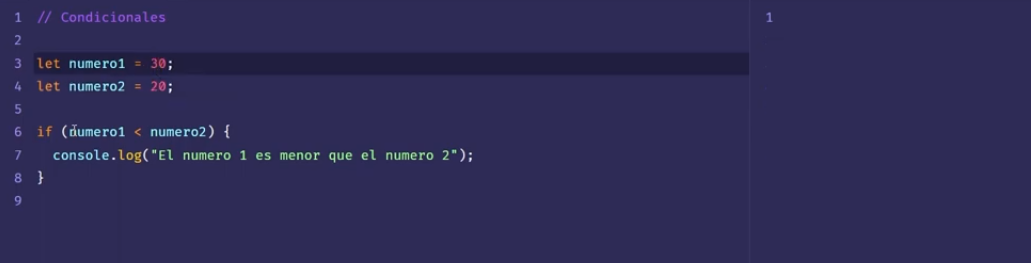
Conjunto de órdenes que se van a ejecutar si una condición se cumple.

Como se escribe;

**If( ) – siempre la condición va entre paréntesis**

****

**En el ejemplo si el numero1 es menor que numero2 consoleá el texto “El numero 1 es menor que el numero 2”**

****

**En el ejemplo dos si cambiamos el numero 10 por el 30 ya no consolea nada porque no se cumple la condición**

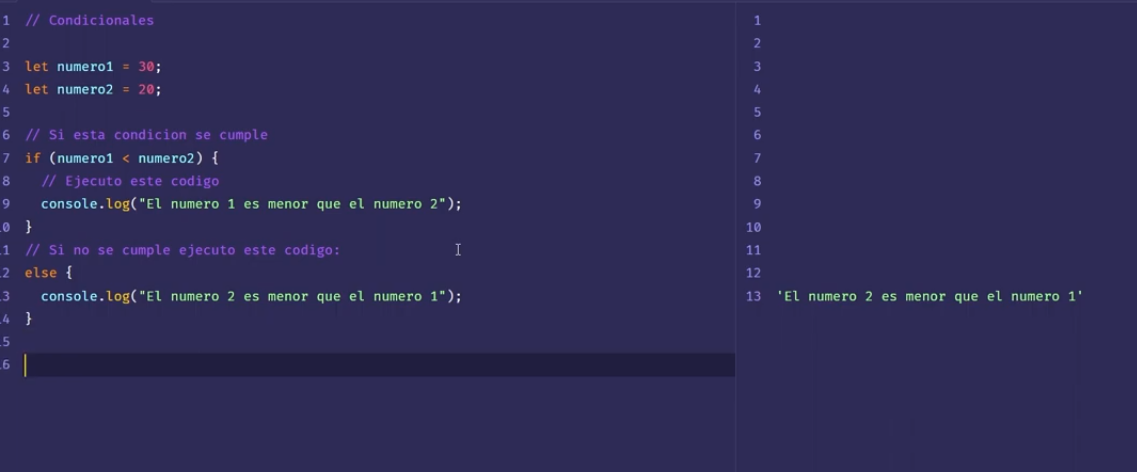
Se escribe de la siguiente forma:

**If** (numero1 < numero2) {

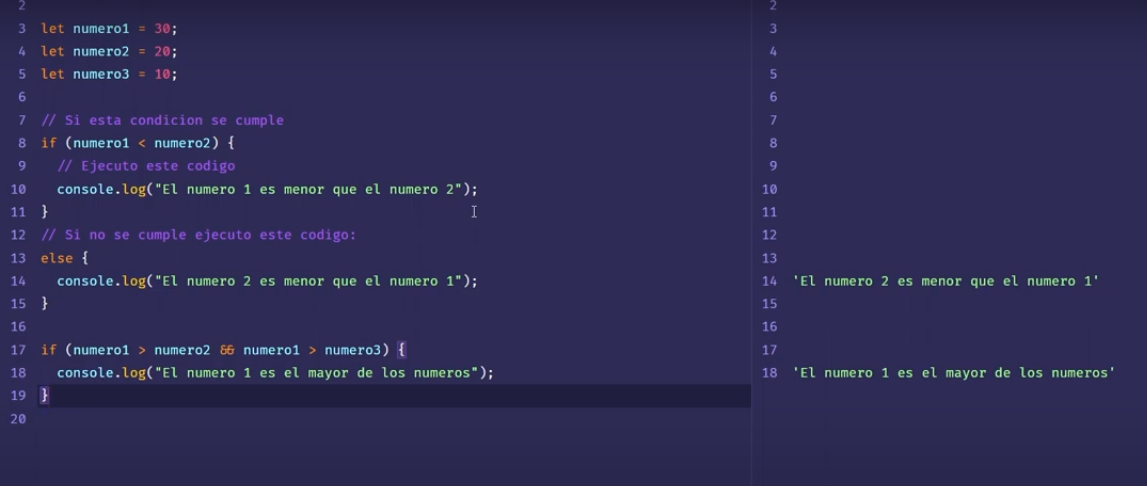
**console.log**(“El numero 1 es menor que el numero 2”)

**Condicional else**

Si la condición **no se cumple se ejecuta el código**

****

Se puede poner una tercera condición

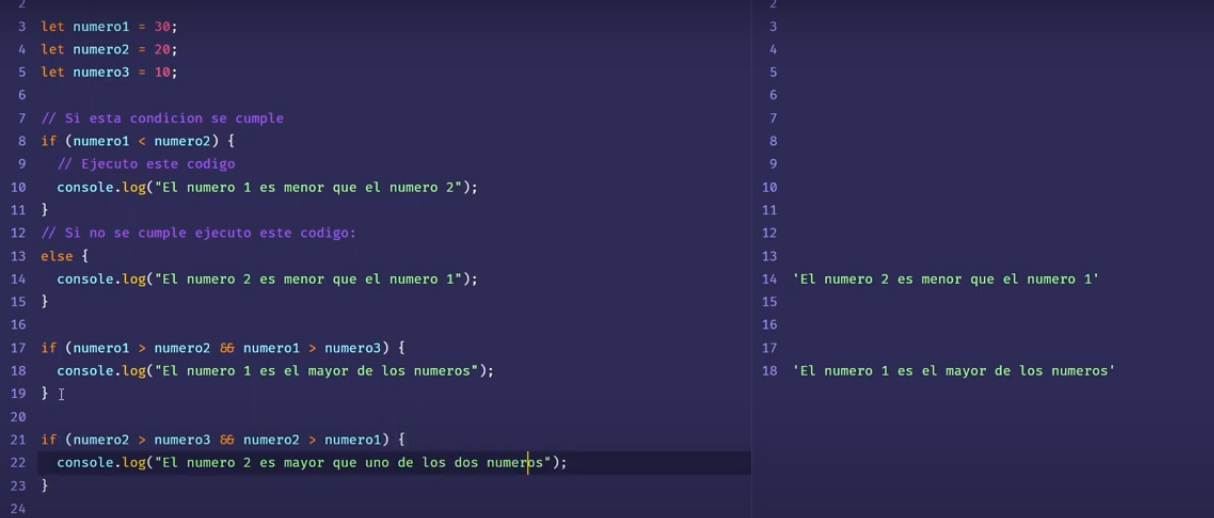


**Si el numero1 es mayor a numero2 y numero 1 es mayor a numero3 ejecuta la orden y de hecho se ejecuta la orden:**

**“El numero 1 es el mayor de los números”**

**Porque el numero 1 es mayor que el numero 2 y el numero 3**

**Condicional or**:



En este caso si colocamos el condicional

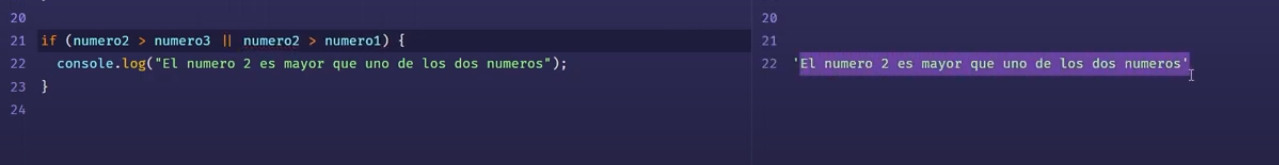
if (numero2 > numero3 && numero2 > numero1) {

Console.log(“El numero 1 es el mayor de los números”);

}

La orden no se ejecuta porque no se cumple porque la segunda condición no se cumple

Para ello utilizamos el condicional **or**

****

if (numero2 > numero3 || numero2 > numero1) {

Console.log(“El numero 1 es el mayor de los números”);

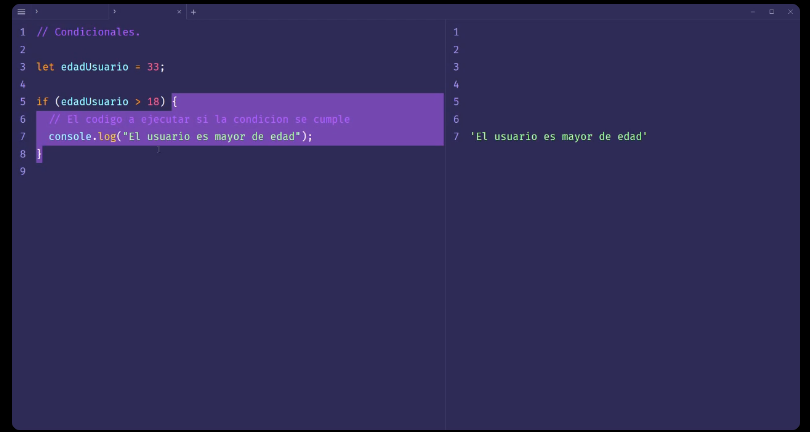
}

La condiconal or se escribe asi: **||**

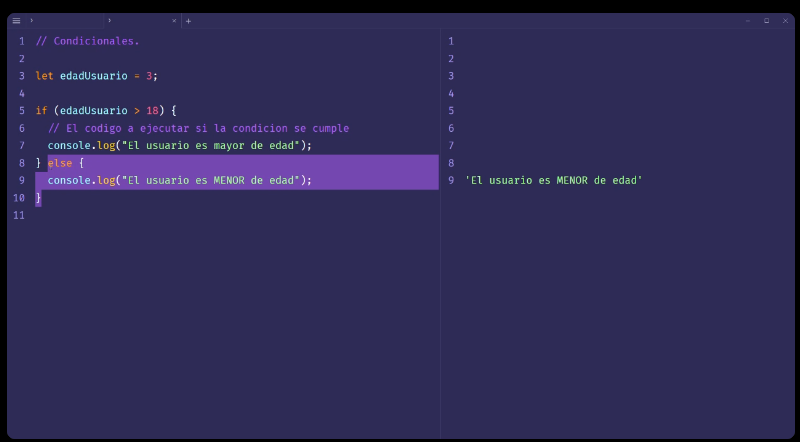
Aca si se ejecuta la orden porque se cumple una u otra condición

**Condicional switch**:

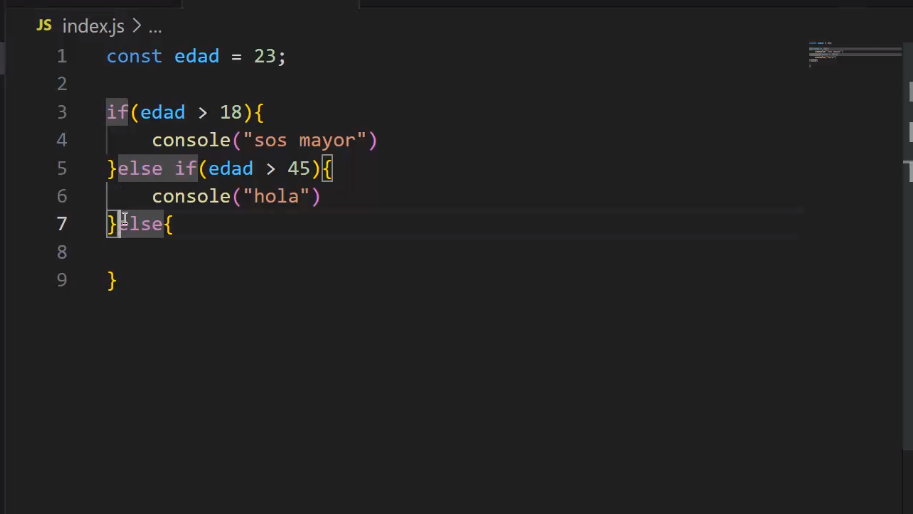
**Clases 03/09/23 condición –if-:**

****

**Se hace con if y con else**

****

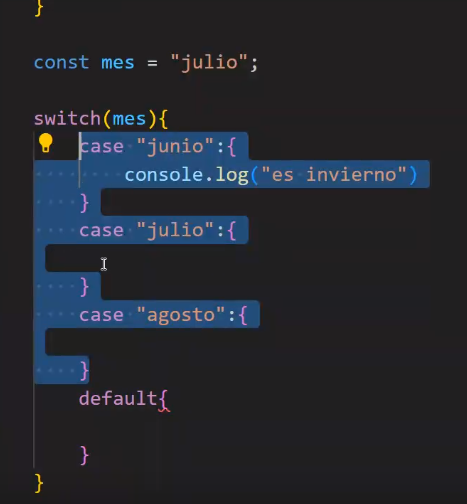
**Podes poner todos los else if que quieras**:

****



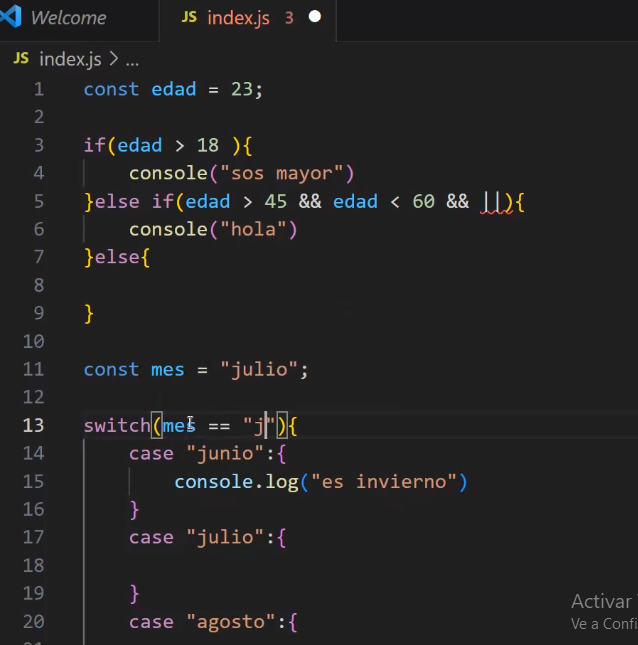
Se cumple y ejecuta lo que esta en corchete

El default sirve si no cumpliste los requisitos



El switch se usa más para string –cadena de texto-

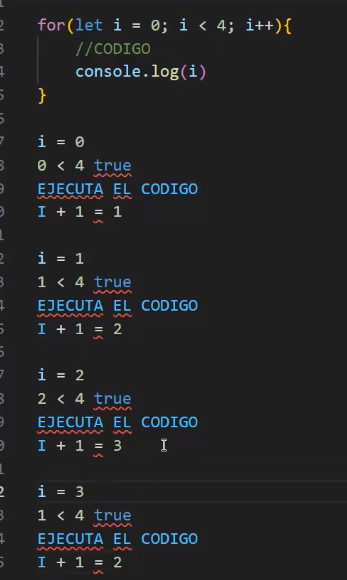
Para condiciones complejas se usa mas el if

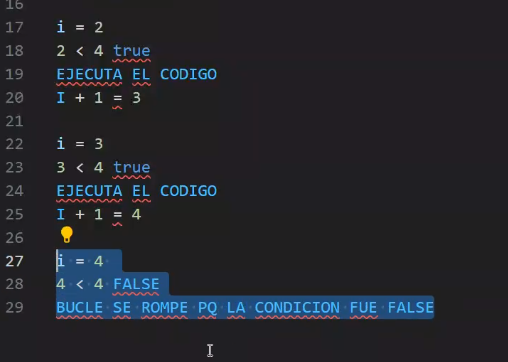


Se usa mas el if el switch se está usando de dejar

**Bucles**:

Sirven para repetir un bloque de código cierta cantidad de veces





Mientras la condición se fue compliendo se ejecuta y se rompe cuando 4 no es menor que 4

**While**: ambos sirven para recorrer lo que hay en un arral –caja con varios datos-

**For**:

