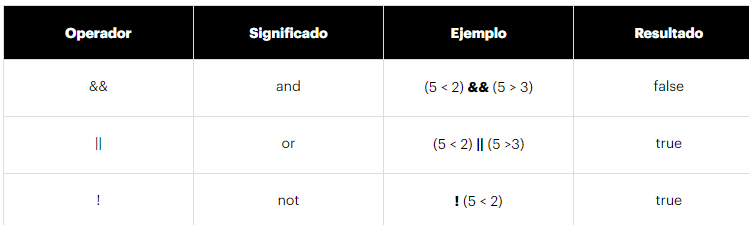
**Operadores lógicos.**

Existen tres tipos de operadores lógicos: la negación o **NOT**, la conjunción o **AND**, y la disyunción u **OR**.



* 1 Las expresiones se evalúan en orden, y la computadora omitirá cualquier expresión redundante. En una declaración **&&**, si la primera expresión es false, la segunda expresión no se evaluará porque AMBAS expresiones deben ser true.
* 2 Lo mismo para la declaración**||**. Si la primera expresión es verdadero, la segunda no se evaluará porque solo debe haber una declaración verdadera para cumplir con los requisitos del operador.
* 3 Utiliza **paréntesis**. Como vimos en el segundo ejemplo de la negación **!**, utilizamos paréntesis para evaluar PRIMERO lo que estaba dentro de ellos, y luego aplicamos la negación. Podemos ajustar cualquier expresión entre paréntesis y se evaluará antes que toda la expresión en conjunto.
* //AND
* function mayorYMenorYPar(num) {
* if( num > 5 && num < 10  && num  % 2 === 0 ) console.log(true);
* else console.log(false);
* };
* mayorYMenorYPar(7);
* mayorYMenorYPar(8);
* // OR
* function operadorOr(str) {
* if (str === 'Henry' || str.length < 2) console.log(true);
* else console.log(false);
* }
* operadorOr('Henry');
* operadorOr('Javascript');
* operadorOr('H');
* // NOT
* function negacion(permiso) {
* if (permiso) console.log('Tiene permiso');
* }
* negacion(true);
* negacion(false);

$node

> Boolean(1)

true

> Boolean(0)

false

> Boolean(true)

true

> Boolean(false)

false

> Boolean("hola")

True

> Boolean("hola")

true

> Boolean(null)

false

> Boolean(undefined)

false

> Boolean([])

True

**Bucles For y While**

**Bucle For**

Por un lado, el Bucle **For** es utilizado cuando sabemos con antelación la cantidad máxima exacta de pasos que queremos ejecutar. Esta cantidad máxima de pasos se sitúa como segundo parámetro entre los paréntesis.

var suma=0;

for (let i = 0; i < 5; i++) {

    suma = suma + i;

    console.log('Variable de iteración: ', i);

 }

 console.log('Variable suma: ', suma);

**Bucle while**

El **Bucle While**es usado cuando no tenemos la certeza de cuántos pasos vamos a necesitar hasta finalizar la ejecución. Por ejemplo, podríamos pensar en una función que siga agregando +1 hasta llegar a un número determinado.

while(suma < 3) {

suma = suma + 1;

console.log(suma);

};

**Expresión Switch**  
La expresión **Switch**evalúa una condición. Compara su valor con una instancia case, y ejecuta declaraciones asociadas a ese case, así como las declaraciones en los case que siguen.

var expr ='Cerezas';

switch (expr) {

    case 'Naranjas':

      console.log('El kilogramo de naranjas cuesta $0.59.');

      break;

    case 'Manzanas':

      console.log('El kilogramo de manzanas cuesta $0.32.');

      break;

    case 'Platanos':

      console.log('El kilogramo de platanos cuesta $0.48.');

      break;

    case 'Cerezas':

      console.log('El kilogramo de cerezas cuesta $3.00.');

      break;

    case 'Mangos':

    case 'Papayas':

      console.log('El kilogramo de mangos y papayas cuesta $2.79.');

      break;

    default:

      console.log('Lo lamentamos, por el momento no disponemos de ' + expr + '.');

  }

  console.log("¿Hay algo más que te quisiera consultar?");

**Bucle Do-While**  
El bucle**Do-While**(hacer mientras) ejecuta una sentencia especificada, hasta que la condición de comprobación se evalúa como falsa. La condición se evalúa después de ejecutar la sentencia, dando como resultado que la sentencia especificada se ejecute al menos una vez.

let result = '';

let i = 0;

do {

  i = i + 1;

  result = result + i;

} while (i < 5);

console.log("result:"+result);

// Expected output: "12345"

**Declaración Continue**  
La declaración **continue** se utiliza dentro de los Bucles For. Nos permite omitir alguna de las iteraciones si se cumple una condición específica.

i = 0;

n = 0;

while (i < 5) {

   i++;

   if (i == 3)

      continue;

   n += i;

   console.log("i:"+i+"    n:"+n);

}

**Break**  
La declaración **break** se utiliza dentro de los Bucles For. Nos permite "romper" o finalizar el bucle con antelación si se cumple una condición específica.

function comprobarBreak(x) {

    var i = 0;

    while (i < 6) {

       if (i == 3)

          break;

       i++;

    }

    return i \* x;

 }

 console.log("break:"+comprobarBreak(5));

break:15

**ver si un nro es par**

const **par** = (**numero**) => {

if (typeof **numero** !== "number") return console. log("Debes ingresar un **numero**");

if (**numero** % 2 === 0) { return console. log(`El **numero** ${**numero**} es **Par**`);

}

Ver si un nro es primo

function esPrimo(num) {

    // Retornar true si "num" es primo.

    // De lo contrario retorna false.

    // [Pista 1]: un número primo sólo es divisible por sí mismo y por 1.

    // [Pista 2]: puedes resolverlo utilizando un `bucle for`.

    // [Nota]: los números negativos, 0 y 1 NO son números primos.

    // Tu código:

    if (num <= 0 || num ===1 || num ===4) return  false;

    for (let x=2 ; x < num/2 ; x++){

       if(num % x ===0) return false;

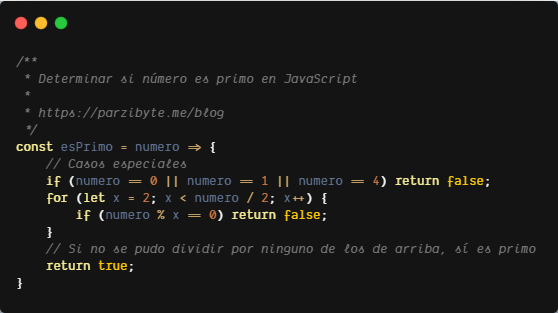
    }

    return true;

 }

Otra forma

Final del formulario

[](https://parzibyte.me/blog/wp-content/uploads/2019/10/N%C3%BAmero-primo-en-JavaScript.png)

Al final tendremos una función que regresará un booleano y nos permitirá saber si un número es primo.

|  |  |  |  |
| --- | --- | --- | --- |
|  | /\*\* | | |
|  | \* Determinar si número es primo en JavaScript | | |
|  | \* | | |
|  | \* https://parzibyte.me/blog | | |
|  | \*/ | | |
|  | const esPrimo = numero => { | | |
|  | // Casos especiales | | |
|  | if (numero == 0 || numero == 1 || numero == 4) return false; | | |
|  | for (let x = 2; x < numero / 2; x++) { | | |
|  | if (numero % x == 0) return false; | | |
|  | } | | |
|  | // Si no se pudo dividir por ninguno de los de arriba, sí es primo | | |
|  | return true; | | |
|  | }  Para probar: | | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
| const numerosParaProbar = [1, 2, 50, 51, 3, 500, 7, 1311]; | | |
|  | | |
| numerosParaProbar.forEach(numero => { | | |
| console.log("¿%d es primo? %s", numero, esPrimo(numero)); | | |
| }); | | |

Homework

function obtenerMayor(x, y) {

   // "x" e "y" son números enteros.

   // Retornar el número más grande.

   // Si son iguales, retornar cualquiera de los dos.

   // Tu código:

   if (x===y) return x;

    else if (x> y) return x;

         else if (y>x) return y;

   }

function mayoriaDeEdad(edad) {

   // Determinar si la persona puede ingresar al evento según su edad.

   // Si tiene 18 años ó más debe retornar el string: "Allowed".

   // Caso contrario: "Not allowed".

   // Tu código:

   if (edad>= 18) return "Allowed";

   return "Not allowed";

}

function conection(status) {

   // El argumento "status" representa el estado de conexión de un usuario.

   // Si el estado es igual a 1, el usuario está "Online".

   // Si el estado es igual a 2, el usuario está "Away".

   // De lo contrario, presumimos que el usuario está "Offline".

   // Retornar el estado de conexión del usuario.

   let estado='';

   switch(status){

      case 1: estado= "Online"; break;

      case 2: estado= "Away"; break;

      default: estado= "Offline";

   }

   return estado;

}

function saludo(idioma) {

   // Retornar un saludo en tres diferentes lenguajes:

   // Si "idioma" es "aleman", devuelve "Guten Tag!".

   // Si "idioma" es "mandarin", devuelve "Ni Hao!".

   // Si "idioma" es "ingles", devuelve "Hello!".

   // Si "idioma" no es ninguno de los anteriores o es `undefined` devuelve "Hola!".

   switch(idioma){

      case 'aleman': saludo='Guten Tag!'; break;

      case 'mandarin': saludo ="Ni Hao!";break;

      case 'ingles': saludo ="Hello!";break;

      default: saludo="Hola!";

   }

   return saludo;

}

function colors(color) {

   let col;

   // La función recibe un color. Retornar el string correspondiente:

   // En caso que el color recibido sea "blue"   --> "This is blue".

   // En caso que el color recibido sea "red"    --> "This is red".

   // En caso que el color recibido sea "green"  --> "This is green".

   // En caso que el color recibido sea "orange" --> "This is orange".

   // Si no es ninguno de esos colores           --> "Color not found".

   // IMPORTANTE: utilizar el statement SWITCH.

   switch (color){

      case 'blue': col="This is blue";break;

      case 'red': col="This is red";break;

      case 'green': col="This is green";break;

      case 'orange': col="This is orange";break;

      default:  col="Color not found";

   }

   return col;

}

function esDiezOCinco(num) {

   // Retornar true si "num" es 10 o 5.

   // De lo contrario, retornar false.

   if (num ===10 || num ===5) return true;

   return false;

}

function estaEnRango(num) {

   // Retornar true si "num" es menor que 50 y mayor que 20.

   // De lo contrario, retornar false.

   if (num>20 && num <50) return true;

   return false;

}

function esEntero(num) {

   // Retornar true si "num" es un entero, ya sea positivo, negativo o cero.

   // Ejemplo: 0.8   ---> false

   // Ejemplo: 1     ---> true

   // Ejemplo: (-10) ---> true

   // De lo contrario, retorna false.

  if ((num %2)===0) return true;

  return false;

}

function fizzBuzz(num) {

   // Si "num" es divisible entre 3, retorna "fizz".

   // Si "num" es divisible entre 5, retorna "buzz".

   // Si "num" es divisible entre 3 y 5 (ambos), retorna "fizzbuzz".

   // De lo contrario, retorna false.

   if ((num%3===0) && (num%5===0)) return "fizzbuzz"

   if (num%3===0) return "fizz"

   if (num%5===0) return "buzz"

   else return false

}

function operadoresLogicos(num1, num2, num3) {

   // La función recibe tres números distintos.

   // Si num1 es mayor a num2 y a num3, y además es positivo, retornar ---> "Numero 1 es mayor y positivo".

   // Si alguno de los tres números es negativo, retornar ---> "Hay negativos".

   // Si num3 es más grande que num1 y num2, aumentar su valor en 1 y retornar el nuevo valor.

   // Si todos los argumentos son cero, retornar ---> "Error".

   // Si no se cumple ninguna de las condiciones anteriores, retornar false.

   if (num1 > num2 && num1 > num3 && num1>0) return "Numero 1 es mayor y positivo";

   if (num1 < 0 || num2 < 0  && num3 < 0 ) return  "Hay negativos";

   if (num3 > num1 && num3 > num2 ){

      num3 +=1;

      return num3;

   }

   if (num1 === 0 && num2 === 0 && num3 === 0) return "Error";

   else return false;

}

function esPrimo(num) {

   // Retornar true si "num" es primo.

   // De lo contrario retorna false.

   // [Pista 1]: un número primo sólo es divisible por sí mismo y por 1.

   // [Pista 2]: puedes resolverlo utilizando un `bucle for`.

   // [Nota]: los números negativos, 0 y 1 NO son números primos.

   // Tu código:

   if (num <= 0 || num ===1 || num ===4) return  false;

   //for (let x=2 ; x < num/2 ; x++){

   for (let x=2 ; x < num -1 ; x++){

      if(num % x ===0) return false;

   }

   return true;

}

function esVerdadero(valor) {

   // Si "valor" es verdadero retornar "Soy verdadero".

   // Caso contrario, retornar "Soy falso".

   // Tu código:

  if(valor) return "Soy verdadero";

  return "Soy falso";

}

function tieneTresDigitos(num) {

   // Si el número recibido tiene tres dígitos retornar true.

   // Caso contrario, retornar false.

   // Tu código:

   let cont=0;

   while (num >=1){

         cont +=1;

         num= num /10;

   }

   if (cont ===3) return true;

   return false;

}

function doWhile(num) {

   // Implementar una función que aumente el valor recibido en 5 hasta un límite de 8 veces.

   // Retornar el valor final.

   // Utilizar el bucle Do-While.

   // Tu código:

   let cont=0;

   do{

      cont+=1;

      num +=5;

   }while(cont<8)

   return num;

}