

JavaScript desde cero

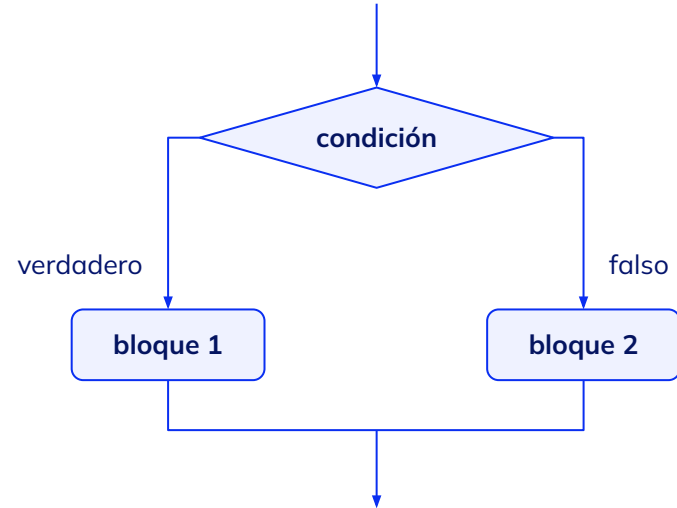
Módulo 2

Estructuras de control de flujo

Estructuras de control de flujo

Si solamente se trabaja con operadores y variables, se genera una sucesión de pasos lineales. No se puede **decidir si algo se mostrará o no** según determinadas circunstancias o, en todo caso, **repetir una instrucción varias veces**.

Para realizar estas secuencias más complejas y con múltiples posibilidades se debe trabajar con **estructuras de control de flujo**.



Tipo *booleano*

Los tipos de variables de datos **booleanos** son aquellos conformados únicamente por dos posibles valores: **true/false**.

Un *booleano* puede ser **definido en forma explícita**, por ejemplo, **al asignar el valor true a una variable**.

Recordemos que, por ejemplo, si utilizamos el cuadro de diálogo **confirm()**, el resultado, que retornará al pulsar alguno de los botones, será **verdadero** o **falso**.

```
let respuesta = confirm("¿Estás segura/o de continuar?")
```

El resultado será el siguiente:

¿Estás segura/o de continuar?

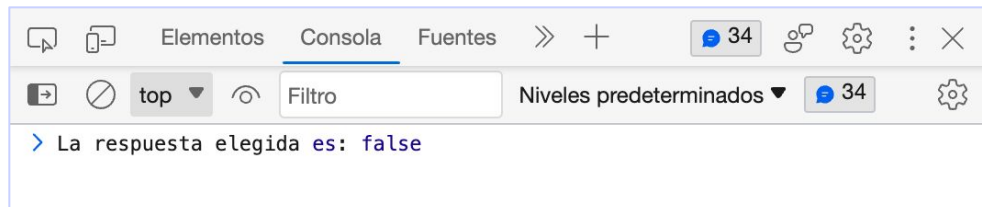
Cancelar

Aceptar

Vamos a mostrar el resultado en la consola JS:

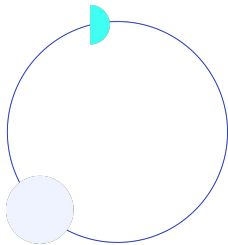
```
let respuesta = confirm("¿Estás segura/o de continuar?")  
  
console.log("La respuesta elegida es:", respuesta)
```

Si el usuario **cancela**, el resultado será el siguiente:



Operadores de comparación

De igual forma, un **valor booleano** puede ser el **resultante de la comparación de valores** que parten de variables, constantes, y otros tantos elementos que conforman una aplicación.



```
let nombre = "EducaciónIT"
let empresa = "EDUCACIONIT"

let nroA = 2005
let nroB = 1996

nombre === empresa      //retornará false
nombre !== empresa      //retornará true
nroA > nroB              //retornará true
nroA >= nroB             //retornará true
nroA < nroB              //retornará false
nroA <= nroB             //retornará false
```

Operadores lógicos

También podemos utilizar operadores lógicos para generar procesos más complejos donde decimos, por ejemplo, que un **número es igual o mayor a otro** pero también que **debe cumplir o no con otra condición**.

```
let nombre = "EducaciónIT"
let empresa = "EDUCACIONIT"

let nroA = 2005
let nroB = 1996

nombre !== empresa && nroA > nroB //retornará true

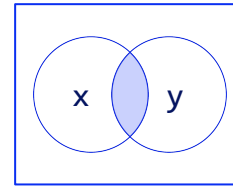
nombre === empresa && nroA > nroB //retornará false

nombre === empresa || nroA < nroB //retornará false

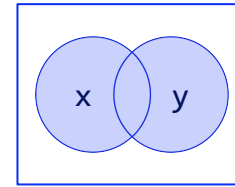
empresa !== nombre || nroA > nroB //retornará true
```

Siempre es interesante **graficar estas situaciones** para entender el proceso de descarte, donde algo termina dando **true** o **false**.

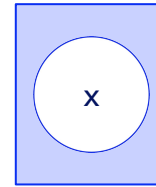
Veamos los gráficos a la derecha. Como podemos apreciar en estos, encontraremos mucha similitud en el tipo de integración de los operadores lógicos respecto al mundo de las matemáticas.



x and y



x or y



not x



Estructuras condicionales

Condicionales

Permiten decisiones acerca de **qué sentencias (órdenes) debe ejecutar el programa**, en base a una **condición / expresión booleana**; es decir, si algo retorna **true/false**.

Se puede decir que a diferencia de los programas que veníamos generando, en este caso, **se seguirán una serie de pasos u otra, según el dato de entrada**.

Por lo tanto, **el resultado o dato de salida dependerá del dato de entrada**.

Los condicionales que explicaremos en las siguientes pantallas son:

- **if.**
- **if, else.**
- **if, else, else if.**



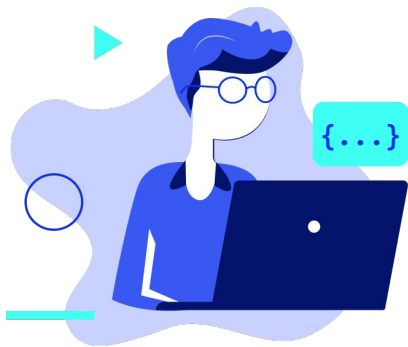
Ejemplo:

```
modoHomeOffice = Sí  
horaActual = 9:00  
  
sí horaActual es igual o mayor a las 9:00  
  "Hora del ☕. También unas 🍪🍪."
```



if

El condicional **if** es la estructura más utilizada al momento de necesitar planteos condicionales para tomar una u otra decisión.



```
// si - condicion
if(condicion) {
    // bloque de codigo a ejecutar
    ...
}
```

Si la condición se cumple

(es decir, si su valor es **true**)

Se ejecutarán las instrucciones que se encuentran dentro de las llaves de bloque `{...}`.

```
let mostrarMensaje = true;

if(mostrarMensaje) {
  alert("Hola Mundo");
}
```

Si la condición no se cumple

(es decir, si su valor es **false**)

No se ejecuta ninguna instrucción contenida en las llaves de bloque `{...}` y el **programa continúa con la ejecución de cualquier otra instrucción o instrucciones que estén por fuera del condicional.**

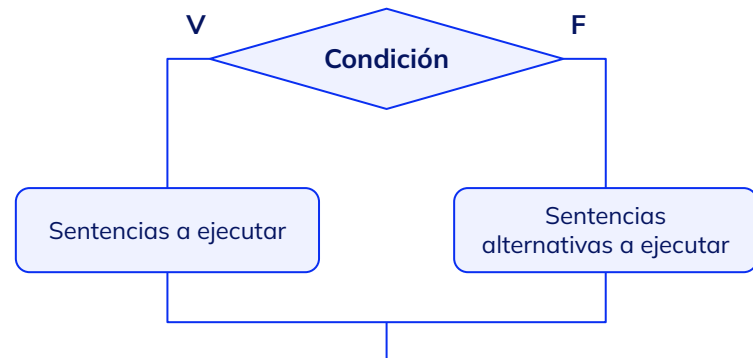
```
let mostrarMensaje = false;

if(mostrarMensaje) {
  alert("No vas a ver este mensaje");
}
```

if, else

En este caso, **se fija una alternativa**. Algoritmo y pseudocódigo que representan esta situación:

```
modoHomeOffice = Sí  
horaActual = 9:00  
  
sí horaActual es igual o mayor a las 9:00  
    "Hora del 🍳. También de unas 🍪🍪"  
    "Prender la compu y loguearme al server"  
sino  
    "Seguir 🤔🤔🤔"
```



La cláusula **else** nos permite definir una condición alternativa, a ejecutar siempre que la condición principal **if()** retorne un valor *booleano false*.

```
let mayorEdad = 18;

if (mayorEdad >= 18) {
  console.log("Es mayor de edad");
} else {
  console.log("Es menor de edad");
}
```



if, else, else if

La estructura **else if** brinda la posibilidad de que **se ejecuten una serie de pasos si una opción (no la primera) es verdadera.**

Es posible encadenar tantos **else if** alternativos, como consideremos necesario hacerlo.



```
let edad = 26;

if(edad < 12) {
  alert("Todavía eres muy pequeño");
}
else if(edad < 19) {
  alert("Eres un adolescente");
}
else if(edad < 35) {
  alert("Aun sigues siendo joven");
}
else {
  alert("Piensa en cuidarte un poco más");
}
```


Condicionales compuestos

La estructura **if, y else...if** permiten analizar condiciones simples, y también compuestas. En este último caso, **podemos sumar diferentes condiciones, validando si una de las dos, si las dos, o si ninguna se cumple.**

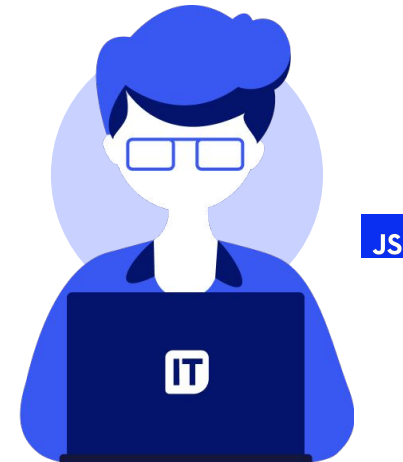
Aquí entran en acción **los condicionales compuestos**, quienes se combinan con los operadores lógicos **&&** (and), **||** (or) y eventualmente **!** (not).

```
let nombreIngresado = prompt("Ingrese su nombre");

if ( (nombreIngresado != "") && ( nombreIngresado == "juan" || nombreIngresado == "JUAN"))
{
    console.log("El nombre ingresado es Juan");
} else {
    console.log("Error: Ingresar nombre valido");
}
```

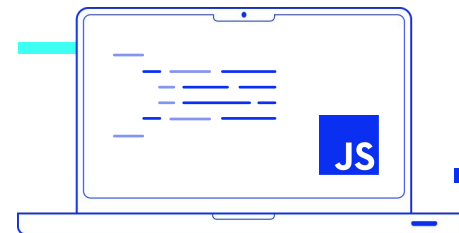
Los operadores lógicos son imprescindibles para realizar aplicaciones complejas, ya que se utilizan para **tomar decisiones sobre las instrucciones** que debería ejecutar el programa en función de ciertas condiciones.

El **resultado** de cualquier operación que utilice operadores lógicos siempre es un **valor lógico**, o **booleano**.



Revisión

- Repasar el concepto de **estructura de control de flujo**.
- Generar diferentes opciones de **pseudocódigo** para trabajar con **condicionales**.
- Implementar **if, else, else if**.
- Combinar los **diferentes tipos de output vistos**.
- Implementar los conocimientos en el **Proyecto Integrador**.



**¡Sigamos
trabajando!**