DIGITALHOUSE

Los Operadores

Índice

01 Lógicos

Los operadores lógicos siempre devolverán un booleano, es decir, true o false, como resultado.

Lógicos

Permiten combinar valores booleanos, el resultado también devuelve un booleano.

Existen tres operadores y (**and**), o (**or**), negación (**not**).

AND (&&) → **todos los valores deben evaluar como true** para que el resultado sea **true**.

```
{ }   (10 > 15) && (10 != 20)     // false
```


```
{ }   (12 % 4 == 0) && (12 != 24)  // true
```

OR (||) → **al menos un valor debe evaluar como true** para que el resultado sea **true**.

```
{ }   (10 > 15) || (10 != 20)      // true
```

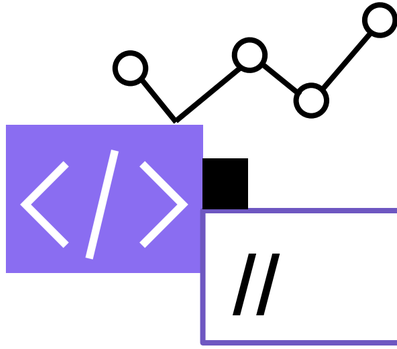

```
{ }   (12 % 5 == 0) && (12 != 12)  // false
```

NOT (!) → niega la condición. Si era true, será false y viceversa.

```
{ }   !false // true
      !(20 > 15) // false
```

02 Condicionales

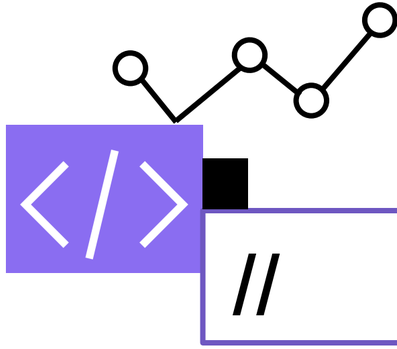
Nos permiten **evaluar condiciones** y realizar diferentes acciones según el resultado de esas evaluaciones.



Condicional simple

Versión más básica del **if**. Establece una condición y un bloque de código a ejecutar en caso de que sea verdadera.

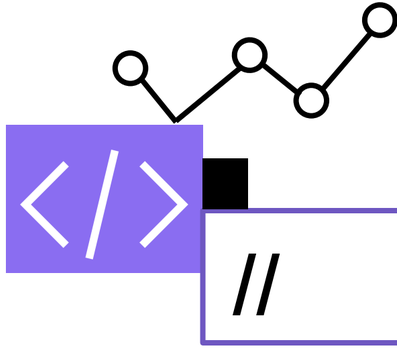
```
{}
if (condición) {
  // código a ejecutar si la condición es verdadera
}
```



Condicional con bloque else

Igual al ejemplo anterior, pero agrega un bloque de código a ejecutar en caso de que la condición sea falsa.
Es importante tener en cuenta que el bloque **else** es opcional.

```
{}
if (condición) {
  // código a ejecutar si la condición es verdadera
} else {
  // código a ejecutar si la condición es falsa
}
```



Condicional con bloque else if

Igual que el ejemplo anterior, pero agrega un if adicional. Es decir, otra condición que puede evaluarse en caso de que la primera sea falsa.

Podemos agregar todos los bloques else if que queramos, **solo uno podrá ser verdadero.** De lo contrario entrará en acción el bloque else, si existe.

{}

```
if (condición) {
  // código a ejecutar si la condición es verdadera
} else if (otra condición) {
  // código a ejecutar si la otra condición es verdadera
} else {
  // código a ejecutar si todas las condiciones son falsas
}
```

Funcionamiento de un if

{código}

```
let edad = 19;
let acceso = '';

if (edad < 16) {
  acceso = 'prohibido';
} else if (edad >= 16 && edad <= 18) {
  acceso = 'permitido solo acompañado de un mayor';
} else {
  acceso = 'permitido';
}
```

Declaramos la variable **edad** y le asignamos el número 19.

{código}

```
let edad = 19;
let acceso = '';

if (edad < 16) {
  acceso = 'prohibido';
} else if (edad >= 16 && edad <= 18) {
  acceso = 'permitido solo acompañado de un mayor';
} else {
  acceso = 'permitido';
}
```

Declaramos la variable **acceso** y le asignamos un string vacío, con la intención de asignarle un nuevo valor según el resultado que arrojen los condicionales declarados a continuación.

{código}

```
let edad = 19;
let acceso = '';
```

```
if (edad < 16) {
  acceso = 'prohibido';
} else if (edad >= 16 && edad <= 18) {
  acceso = 'permitido solo acompañado de un mayor';
} else {
  acceso = 'permitido';
}
```

Iniciamos el condicional. Nuestra primera condición evalúa si **edad** es menor a 16. En caso de ser **verdadera**, le asignamos el string *'prohibido'* a la variable **acceso**.
En este caso, l**a condición es falsa**, por lo tanto **JavaScript pasa a evaluar la siguiente condición**.

{código}

```
let edad = 19;
let acceso = '';

if (edad < 16) {
  acceso = 'prohibido';
} else if (edad >= 16 && edad <= 18) {
  acceso = 'permitido solo acompañado de un mayor';
} else {
  acceso = 'permitido';
}
```

Declaramos un bloque **else if** para contemplar una **segunda condición**:
Esta condición va a ser compuesta y va a requerir:
- que edad sea mayor o igual a 16
- que edad sea menor o igual a 18

La condición nuevamente es **falsa**, por lo tanto JavaScript continúa leyendo el condicional.

{código}

```
let edad = 19;
let acceso = '';

if (edad < 16) {
  acceso = 'prohibido';
} else if (edad >= 16 && edad <= 18) {
  acceso = 'permitido solo acompañado de un mayor';
} else {
  acceso = 'permitido';
}
```

Como **ninguna** de las condiciones anteriores era **verdadera**, se ejecuta el código dentro del else.

Por lo tanto, ahora la variable **acceso** es igual al string *'permitido*'.

#Tip DH

Es una **buena práctica** inicializar las variables con el **tipo de dato** que van a almacenar.

De esa manera queda más claro para qué se van a utilizar.

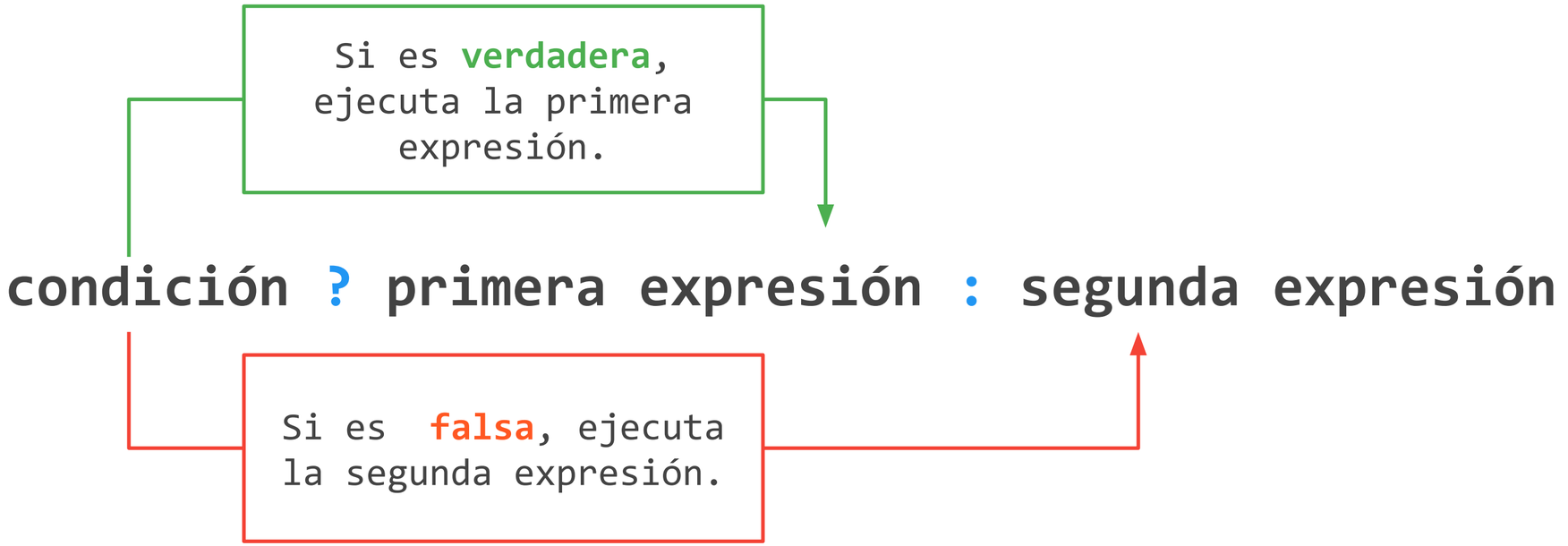
```
let texto = ''; // un texto vacío
let numero = 0; // un número vacío
let listado = []; // un array vacío
let objeto = {}; // un objeto vacío
```

03 Condicional Ternario

Si algo se usa mucho en programación, los lenguajes suelen darnos una versión abreviada.

Estructura básica

A diferencia de un if tradicional, el **if ternario** se escribe de forma **horizontal.** Al igual que el if tradicional, tiene el mismo flujo (si esta condición es verdadera hacer esto, si no, hacer esto otro), pero en este caso **no hace falta** escribir la palabra **if ni la palabra else.**



Encontrá más info sobre if ternario: aquí

Estructura básica

Para el if ternario **es obligatorio** poner código en la **segunda expresión**. Si no queremos que pase nada, podemos usar un string vacío `''`.

```
{}  4 > 10  ? 'El 4 es más grande' : 'El 10 es más grande';
```

Condición

Declaramos una expresión que se evalúa como true o false.

Primera expresión

Si la condición es verdadera, se ejecuta el código que está después del signo de interrogación.

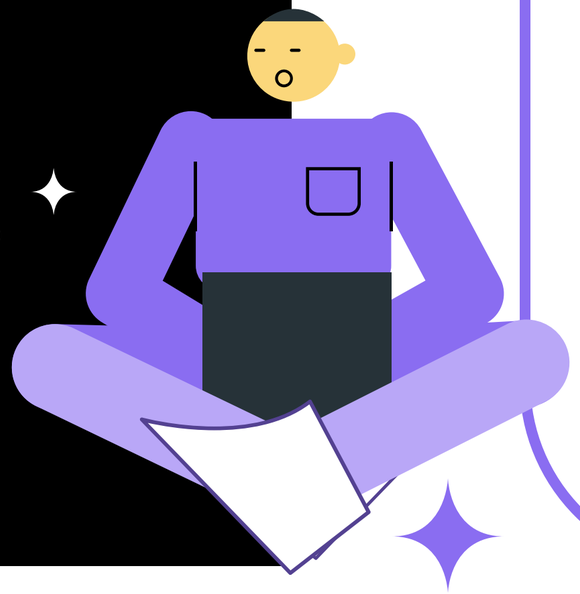
Segunda expresión

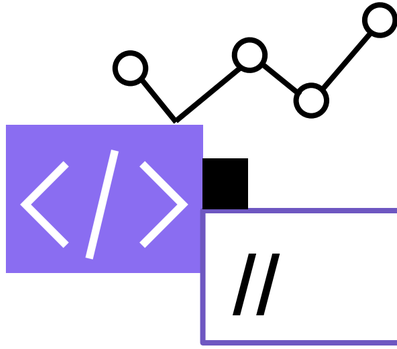
Si la condición es falsa, se ejecuta el código que está después de los dos puntos.

Es obligatorio escribirla.

04 Switch

El switch nos propone una sintaxis más legible para los casos en los que queremos evaluar muchas posibilidades de un solo valor.





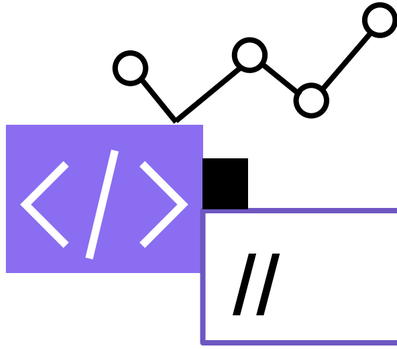
Estructura básica

El switch está compuesto por una expresión a evaluar, seguida de diferentes casos, tantos como queramos, cada uno contemplando un escenario diferente.

Los casos deberán terminar con la palabra reservada break para evitar que se ejecute el próximo bloque.

{}

```
switch (expresión) {
    case valorA:
        // código a ejecutar si la expresión es igual a valorA
        break;
    case valorB:
        // código a ejecutar si la expresión es igual a valorB
        break;
}
```



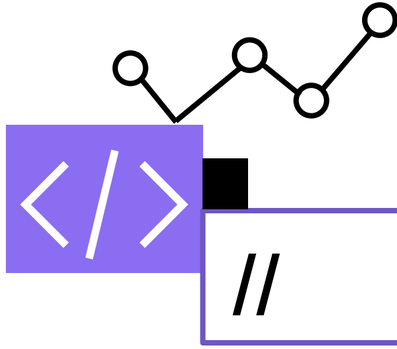
Agrupamiento de casos

El switch también nos **permite agrupar casos** y ejecutar un mismo bloque de código para cualquier caso de ese grupo.

```
{}
```

```
switch (expresión) {
 case valorA:
 case valorB:
    // código a ejecutar si la expresión es igual a ValorA o B
    break;
case valorC:
    //código a ejecutar si valorC es verdadero
    break;
}
```

Funcionamiento de un switch

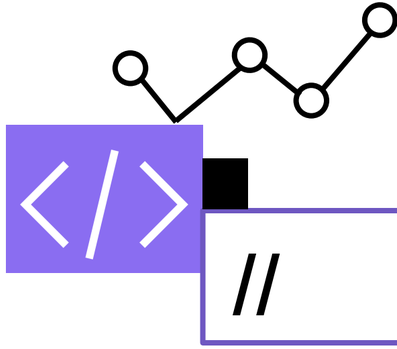


{código}

```
let edad = 5;

switch (edad) {
   case 10:
         console.log('Tiene 10 años');
         break;
   case 5:
         console.log('Tiene 5 años');
         break;
}
```

Definimos la variable edad y le asignamos el número 5.



{código}

```
let edad = 5;

switch (edad) {
   case 10:
        console.log('Tiene 10 años');
        break;
   case 5:
        console.log('Tiene 5 años');
        break;
}
```

Iniciamos el condicional con la palabra reservada switch y, entre paréntesis, la expresión/condición que queremos evaluar.

En este caso vamos a evaluar **qué valor tiene la variable edad**.

{código}

```
let edad = 5;

switch (edad) {
   case 10:
         console.log('Tiene 10 años');
         break;
   case 5:
         console.log('Tiene 5 años');
         break;
}
```

Por cada caso escribimos la palabra reservada **case** y a continuación el valor que queremos evaluar.

En este caso, **preguntamos si** el valor de la variable **edad es 10**.

Como este caso N**O es verdadero**, JavaScript ignora el código de este caso y pasa a evaluar el siguiente.

{código}

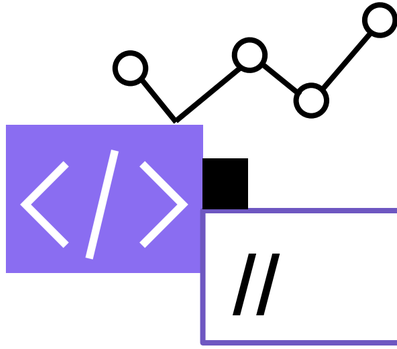
```
let edad = 5;

switch (edad) {
   case 10:
         console.log('Tiene 10 años');
         break;
   case 5:
         console.log('Tiene 5 años');
         break;
}
```

Este caso es **verdadero**, por lo tanto, se ejecutará el código del bloque.

La palabra reservada **break** corta la ejecución del switch.

Si olvidamos el break, los bloques se seguirán ejecutando sin importar si los casos se cumplen o no.

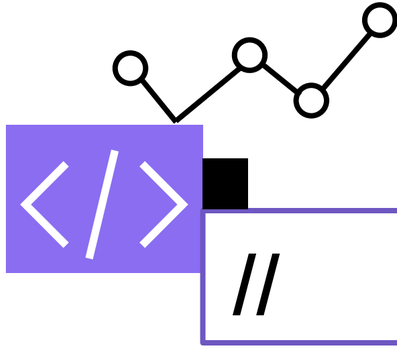


El bloque default

Si queremos considerar la posibilidad de que ninguno de los casos sea verdadero, utilizamos la palabra reservada default seguida de dos puntos : y el bloque de código que queramos que se ejecute.

Por lo general escribimos el bloque default a lo último. En ese caso, no es necesario escribir el break.

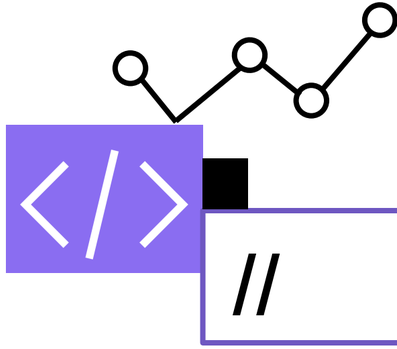
```
{}
switch (expresión) {
    case valorA:
    // código a ejecutar si valorA es verdadero
     break;
default:
    // código a ejecutar si ningún caso es verdadero
}
```



{código}

```
let fruta = 'wefwef';
switch (fruta) {
   case 'manzana':
        console.log('Qué rica la manzana');
        break;
   case 'naranja':
        console.log('¡Naranja, me
   encanta!');
        break;
   default:
        console.log('¿Qué fruta es?');
        break;
}
```

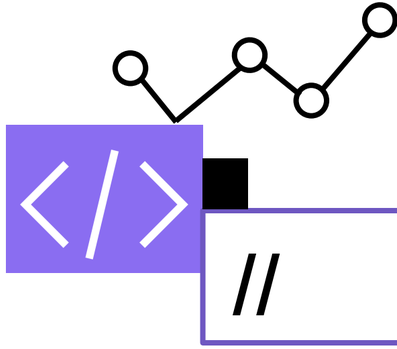
Definimos la expresión que vamos a evaluar en el switch. En este caso queremos preguntar por el valor de la variable fruta.



{código}

```
let fruta = 'wefwef';
switch (fruta) {
   case 'manzana':
        console.log('Qué rica la manzana');
        break;
   case 'naranja':
        console.log('¡Naranja, me
   encanta!');
        break;
   default:
        console.log('¿Qué fruta es?');
        break;
}
```

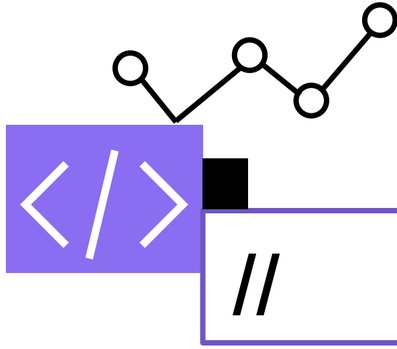
Este caso **es falso**, por lo tanto no se ejecuta su código.



{código}

```
let fruta = 'wefwef';
switch (fruta) {
   case 'manzana':
        console.log('Qué rica la manzana');
        break;
   case 'naranja':
        console.log('¡Naranja, me
   encanta!');
        break;
   default:
        console.log('¿Qué fruta es?');
        break;
}
```

Este caso **también es falso**, por lo tanto no se ejecuta su código.



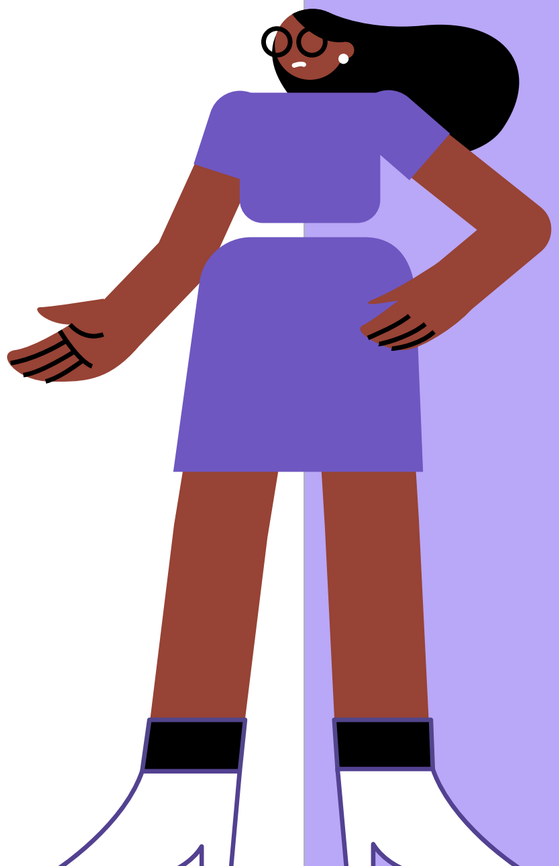
{código}

```
let fruta = 'wefwef';
switch (fruta) {
   case 'manzana':
        console.log('Qué rica la manzana');
        break;
   case 'naranja':
        console.log('¡Naranja, me
   encanta!');
        break;
   default:
        console.log('¿Qué fruta es?');
        break;
}
```

Como ningún caso fue **verdadero**, se ejecuta el código dentro del bloque default.

Conclusiones

Los **operadores lógicos** y de **comparación** nos van a resultar de mucha ayuda a la hora de realizar toma de decisiones, por lo tanto es importante tenerlo siempre en mente.



DIGITALHOUSE

¡Muchas gracias!