

# M1 - 4 - JS Condicionales

## Condicionales

Los programas de JavaScript que hemos hecho hasta el momento estaban basados en variables y operadores de manera que nuestro código se ejecutaba de manera lineal, instrucción por instrucción, es decir, línea a línea.

Ahora podremos hacer que ciertas líneas se ejecuten, según ciertos comportamientos.

### Variables booleanas

Vimos cómo podíamos crear variables con números y texto, pero también existen otras variables que tienen dos resultados posibles, verdadero o falso.

```
» let a = 5;
← undefined
» let b = 'hola';
← undefined
» let c = true;
← undefined
» let d = false;
← undefined
» typeof a;
← "number"
» typeof b;
← "string"
» typeof c;
← "boolean"
```

Estas variables las podemos utilizar para que algunas partes de nuestro código se ejecuten y otras no. Esto lo hacemos con la ayuda de las estructuras condicionales.

## if

Es la primera de las estructuras condicionales, nos permite “bifurcar” la respuesta de nuestro código según la condición que le pongamos.

```
if(condición) {  
    ...  
}
```

Si lo que hay dentro de los paréntesis es verdadero '**true**' , el código que está entre las llaves se ejecuta. Las llaves marcan el inicio y el final del bloque de código que se ejecutará cuando lo que hay entre paréntesis es verdadero. Cuando la condición sea falsa **false**', ese bloque de código no se va a ejecutar.

The image shows a code editor with a script block containing an if-else statement. The first condition is true, so 'hola' is logged. The second condition is false, so 'adiós' is not logged. Below the code, a terminal window shows the execution of the script and the resulting output 'hola'.

```
<script>  
    if(true) {  
        console.log('hola');  
    }  
    if(false) {  
        console.log('adiós');  
    }  
</script>  
  
» if(true) {  
    console.log('hola');  
}  
hola  
← undefined  
» if(false) {  
    console.log('adiós');  
}  
← undefined  
» |
```

Pero entonces ¿ Para qué sirve el '**false**' si sabemos que nunca se va a ejecutar?

Para ello, utilizaremos dos tipos de operadores : Lógicos y Relacionales, que nos ayudarán a definir una condición. Esa condición se evaluará y será entonces cuando su valor sea verdadero 'true' o falso 'false' .

## Operadores relacionales

Se llaman operadores relacionales, por que relacionan variables, su resultado siempre es un valor booleano → 'true' o 'false'

- > - mayor que
- < - menor que
- $\geq$  - mayor o igual que
- $\leq$  - menor o igual que
- === o == - igual que, ahora veremos la diferencia entre ambos
- !== o != - distinto de, ahora veremos la diferencia entre ambos

Veamos unos ejemplos:

- `4 > 3` : 4 es mayor que 3, esto es verdad por lo tanto el resultado es `true`
- `3 < 3` : 3 no es menor que 3, esto es mentira por lo tanto el resultado es `false`

En el segundo caso si hubiéramos indicado lo siguiente → `3 <= 3` esto sí que sería verdad, puesto que contempla las dos opciones, MENOR o IGUAL . 3 no es menor que 3, pero sí es igual, por lo que mientras se cumpla una de las dos es suficiente para que esto sea verdad.

```
» 4 > 3
← true
```

```
» 3 < 3
← false
```

### Diferencia entre ' === !==' y ' == !='

Usamos los dos iguales '`== !=`' para comparar datos/valores, es decir, si 2 es igual a 2, si 2 es distinto de 3... Pero ¿Qué problema hay? que no toma en cuenta el **TIPO** de dato, por lo tanto, podríamos comparar un string cuyo valor fuese '`'3'`' con `3`, que sería un int. JavaScript convierte los dos valores al mismo tipo de dato y los compara.

Usamos los 3 iguales ‘`==`’ ‘`!=`’ para comparar no solo el dato/valor, si no también el **TIPO** de dato. Es decir, comprobará si el valor de ambos es el mismo y si el tipo de dato también es el mismo. Para que sea verdad ‘`true`’ ambos deben ser verdad, tanto el mismo valor como el mismo tipo de dato. Por lo tanto, nosotros vamos a utilizar es `==` `!=` para evitar problemas y errores.

```
>> 3 == 3
← true
>> 3 == 5
← false
>> |
```

```
>> "3" == 3
← true
>> "3" === 3
← false
>> "3" != 3
← false
>> "3" !== 3      ↵
← true
```

## Estructuras condicionales con expresiones booleanas

Cómo habíamos comentado anteriormente, veremos cómo estas ‘expresiones booleanas’ las podemos utilizar en las estructuras condicionales, para definir una condición. La finalidad del siguiente ejercicio es pedir al usuario su edad y comprobar si es mayor o menor de edad. Para ello, y a través de un ‘prompt’ mostramos el mensaje al usuario y recogemos la respuesta en la variable ‘edad’.

Siguiendo la estructura del if, entre paréntesis definimos nuestra condición para que las instrucciones que están entre llaves se ejecuten.

En este caso, nosotros queremos indicar al usuario que es mayor de edad, por lo que la condición para que eso se ejecute deberá ser que efectivamente que el usuario sea mayor de edad ¿No? ¿Y cómo sabemos si una persona es mayor de edad? Por medio de su edad, si tiene 18 años o más, es mayor de edad, con lo que la condición que debemos definir deberá ser que la ‘edad’ introducida por el usuario sea **MAYOR o IGUAL a 18**, si eso es verdad, es decir, si es mayor de edad, entonces se ejecutará el código que está entre las llaves, mostrando por consola un mensaje al usuario que dice ‘Eres mayor de edad’.

```
let edad = parseInt(window.prompt('Cuantos años tienes'));

if(edad >= 18) {
    console.log('Eres mayor de edad');
}
```

## if...else

En el caso anterior hemos visto un ejemplo de cómo si la edad del usuario era 18 años o más, mostrábamos un mensaje indicando que era mayor de edad. Pero ¿Si no se cumplía esta condición? Es decir, ¿Si el usuario tenía menos de 18 años? En ese caso el programa no hacía nada. Para eso, utilizamos la estructura ‘if...else’

```
if(condición) {
    ...
}
else {
    ...
}
```

Si la condición del if se cumple, es ‘**true**’, entonces el código que está entre las llaves del ‘if’ se ejecutará, y **SI NO...** (que es lo que viene a significar ‘else’) se ejecutará el código que está entre las llaves del ‘else’.

Siguiendo con el ejemplo anterior, si la edad del usuario es **MAYOR o IGUAL a 18**, entonces se ejecutará el código que tenemos entre las llaves del ‘if’ mostrando un mensaje de alerta en pantalla → Eres mayor de edad.

**SI NO** se mostrará el mensaje que indicamos entre las llaves del ‘else’ → Eres menor de edad.

```
<script>
    let edad = parseInt(window.prompt('Cuantos años tienes'));

    if(edad >= 18)
    {
        window.alert('Eres mayor de edad');
    }
    else
    {
        window.alert('Eres menor de edad');
    }
</script>
```

Este ejercicio, lo podríamos haber hecho también utilizando una variable ‘mensaje’ a la cual le asignamos un mensaje diferente dependiendo de la condición, y después, mostramos dicho mensaje.

```
<script>
    let edad = parseInt(window.prompt('Cuantos años tienes'));
    let mensaje;
    if(edad >= 18) {
        mensaje = 'Eres mayor de edad';
    } else {
        mensaje = 'Eres menor de edad';
    }
    window.alert(mensaje);
</script>
</body>
```

## if... else if...else

```
if(condición) {
    ...
} else if (condición){
    ...
} else if (condición){
    ...
}
```

Podemos encadenar las estructuras condicionales para realizar varias condiciones seguidas. Por ejemplo, queremos preguntar la nota que ha sacado el usuario en un examen y según la nota mostrar un mensaje u otro. Para ello tendremos varios niveles que definiremos, y según el nivel en el que se encuentre la nota que nos introduzca le mostraremos un mensaje.

Niveles y mensajes asociados:

- 0 < 5 → Suspenso
- 5 < 6 → Suficiente
- 6 < 7 → Bien
- 7 < 9 → Notable
- 9 < 10 → Sobresaliente

Lo primero que tenemos que hacer, es definir las condiciones que contemplaran dichos niveles. Empezaremos por el ‘suspenso’, el suspenso será para las notas inferiores a 5, entonces tenemos dos posibilidades para definir esta condición :

- nota <= 5 → nota menor o igual que 5
- nota < 5 → nota menor que 5

Lo que tenemos que tener en cuenta es que si la nota del usuario es un 5, su nota sería ‘suficiente’, no suspenso, con lo que tendremos que utilizar la opción 2. La condición se interpretaría como : si la nota es menor que 5, mostraremos el mensaje → ‘suspenso’.

```
if(nota < 5) {  
    window.alert('Suspenso');  
}
```

Si no, pasaremos a comprobar si su nota se encuentra entre el siguiente nivel. El siguiente nivel, es para aquellos que hayan sacado una nota inferior a 6, y de nuevo nos encontramos con varias opciones:

- nota <= 6
- nota < 6

La solución es la misma que la anterior y que las siguientes que nos vamos a plantear, el 6 supondría ya un ‘bien’ con lo que si utilizamos la opción 1 y el usuario ha sacado un 6, le indicaremos que su nota es suficiente, y eso es incorrecto según lo planteado. Por ello, utilizaremos de nuevo la opción 2, y haremos lo mismo con las opciones siguientes:

```
if(nota < 5) {  
    window.alert('suspenso');  
} else if (nota < 6){  
    window.alert('suficiente');  
} else if (nota < 7){  
    window.alert('bien');  
} else if (nota < 9){  
    window.alert('notable');  
} else if (nota <= 10){  
    window.alert('sobresaliente');  
}
```

En el último caso, si que queremos incluir el 10 en la condición puesto que si el usuario saca un 10 también sería sobresaliente.

Veamos este ejemplo desde el Visual Studio Code:

```
let nota = parseInt(window.prompt('Qué nota has sacado'));
```

En este ejemplo, veremos cómo hemos sustituido la última condición por un else, veremos qué diferencia hay. Cuando introducimos un else final a una estructura 'if...else if' lo que indicamos es que si ninguna de las anteriores condiciones se ha cumplido se va a ejecutar el código que se encuentre entre las llaves de este else.

Esto por un lado nos permite ejecutar algo sí o sí, es decir, si alguna de las condiciones se cumple se ejecutará la instrucción que se encuentre entre las llaves que corresponda con esa condición. El else, nos permite además contemplar la posibilidad de que ninguna de las anteriores condiciones se haya cumplido y ejecutar una instrucción en consecuencia.

En este caso, sería mucho más correcto utilizar el 'else if' final con la condición 'nota <= 10', por que si no, a través del 'else' final, el usuario podría introducir un 20, lo cual sería algo erróneo y le seguiríamos mostrando el mensaje de 'sobresaliente'. Lo que sería un fallo en la lógica de nuestro programa.

```
if (nota < 5) {  
    window.alert('Suspensos');  
} else if (nota < 6) {  
    window.alert('Suficiente');  
} else if (nota < 7) {  
    window.alert('Bien');  
} else if (nota < 9) {  
    window.alert('Notable');  
} else {  
    window.alert('Sobresaliente');  
}
```

Aún así, hay en casos que sí que nos va a interesar introducir un else final, por ejemplo para mostrar un mensaje de advertencia en el caso de que el usuario haya introducido mal los datos:

```
if(nota < 5) {  
    window.alert('suspenso');  
}else if (nota < 6){  
    window.alert('suficiente');  
}else if (nota < 7){  
    window.alert('bien');  
}else if (nota < 9){  
    window.alert('notable');  
}else if (nota <= 10){  
    window.alert('sobresaliente');  
}else {
```

```
    window.alert('El número introducido no es correcto');
}
```

Realmente esta estructura ‘if ... else if’ es una estructura abreviada de encadenaciones de la estructura ‘if ...else’

```
if(nota < 5) {
    window.alert('Suspensos');
} else {
    if(nota < 6) {
        window.alert('Suficiente');
    } else {
        if(nota < 7) {
            window.alert('Bien');
        } else {
            if(nota < 9) {
                window.alert('Notable');
            } else {
                window.alert('Sobresaliente')
            }
        }
    }
}
```

## switch

Por último, tenemos la estructura ‘switch’ que sirve para contemplar múltiples condiciones (igual que el else if) pero de una manera mucho **más eficiente y específica**.

```
witch(variable) {
    case valor1:
        ...
    case valor2:
        ...
    case valorN:
        ...
}
```

A través del ‘switch’ comprobamos los posibles valores de una variable en específico. Dentro de los paréntesis, introducimos el nombre de la variable que queremos comparar. Vamos a comparar el valor de la variable con los diferentes valores que contemplamos con cada uno de los casos ‘case’. Entre las llaves del switch, introducimos tantos casos ‘case’ como valores queremos comparar, separado por un espacio el valor en específico que ese

caso va a contemplar. Después de cada caso, escribimos dos puntos y después, las instrucciones que queremos que se ejecuten si ese caso se cumple. Ahora bien, esta opción no nos permite aplicar rangos, es decir, nos permite comparar la variable con valores específicos:

- si nota es igual a 0
- si nota es igual a 6
- o si nota es igual a 8

No contempla más opciones ni rangos. Esta estructura nos viene bien en esos casos en los que queremos comprobar o valorar opciones muy concretas, por ejemplo, si le pedimos al usuario el día de la semana en el que estamos, sólo hay 7 opciones, y son específicas : lunes, martes, miércoles, jueves...

```
switch(nota) {  
    case 0:  
        window.alert('suspenso');  
  
    case 6:  
        window.alert('bien');  
  
    case 8:  
        window.alert('notable');  
}
```

Ahora bien, si ejecutamos lo anterior e introducimos un 0 veremos como se nos va a ejecutar: suspenso, bien y notable.

Esto es por que una vez un caso se cumple y se ejecuta la sentencia que le corresponde, se ejecuta el resto de instrucciones que tenemos por debajo también.

Si volvemos a ejecutarlo e introducimos un 6, veremos por pantalla : bien y notable.

Para que no nos pase esto, debemos añadir un '**break;**' al final de cada caso. El 'break' asegura que el programa salga del switch una vez que se ejecute la instrucción coincidente y continúe la ejecución en la instrucción siguiente.

```
switch(nota) {  
    case 0:  
        window.alert('suspenso');  
        break;  
  
    case 6:  
        window.alert('bien');  
        break; // Se agrega el break para evitar el efecto de cascada  
  
    case 8:  
        window.alert('notable');  
        break;  
}
```

```
let nota = parseInt(window.prompt('Qué nota has sacado'));

switch(nota) {
    case 0:
    case 1:
    case 2:
    case 3:
    case 4:
        window.alert('suspenso');
        break;
    case 5:
        window.alert('suficiente');
        break;
    case 6:
        window.alert('bien');
        break;
    case 7:
    case 8:
        window.alert('notable');
        break;
    case 9:
    case 10:
        window.alert('sobresaliente');
        break;
    default:
        window.alert('no entiendo ese valor');
        break;
}
```

Además del 'break,' podemos añadir también '**default**', el default es la instancia en la que va a entrar el programa, ejecutando las instrucciones correspondientes, en caso de que NINGUNO de los casos anteriores haya sido coincidente, es decir, que ninguno de los valores que contempla cada caso coincida con el valor de la variable.

También podemos agrupar valores en casos en los que no indicamos el 'break'. En este ejemplo si el usuario introduce un 0,1,2,3, o 4, se va a ejecutar la misma instrucción, la que indicamos en el caso 4. Y lo mismo pasaría con el caso 7 y 8, y el 9 y el 10.

# Operadores lógicos

A través de los operadores AND ‘`&&`’ y OR ‘`||`’ podemos combinar varias condiciones. Utilizamos el operador AND representado por el símbolo ‘`&&`’ para indicar que para que la condición se cumpla, todas las condiciones que estamos concatenando deben ser VERDAD ‘`true`’. Por ejemplo, si estamos llevando la gestión de una empresa de alquileres de coche y para alquilar el coche es necesario ser mayor de edad y, además, tener el carnet de conducir.

```
let mayorEdad = true;
let carnetCoche= false;
let alquilerCoche = mayorEdad && carnetCoche; // resultado = false

mayorEdad = true;
carnetCoche = true;
alquilerCoche = mayorEdad && carnetCoche ; // resultado = true

mayorEdad = false
carnetCoche = false
alquilerCoche = mayorEdad && carnetCoche: // resultado = false
```

En este caso el código entre llaves sólo se cumplirá si ambas condiciones son verdad, es decir, si la nota es mayor o igual que 0 ‘Y’ si la nota es menor que 5. Esto lo que nos permite es crear un rango, las instrucciones entre las llaves se van a ejecutar si la nota se encuentra entre el 0 y el 4,999...

```
let nota = parseInt(window.prompt('Introduce una nota'));

if(nota >= 0 && nota < 5) {
    window.alert('Suspensión');
}
```

Utilizamos el operador OR representado por el símbolo ‘`||`’ para indicar que para que la condición se cumpla, necesitamos que al menos UNA de las condiciones que estamos concatenando deben ser VERDAD ‘`true`’. Por ejemplo, si estamos llevando la gestión de un aeropuerto, para llevar control de los pasajeros, estos necesitan identificarse a través de un DNI o un Pasaporte. Con que tengan uno de los dos vale.

```
let Dni = true;
let Pasaporte = false;
let identificado = Dni || Pasaporte; // resultado = true
```

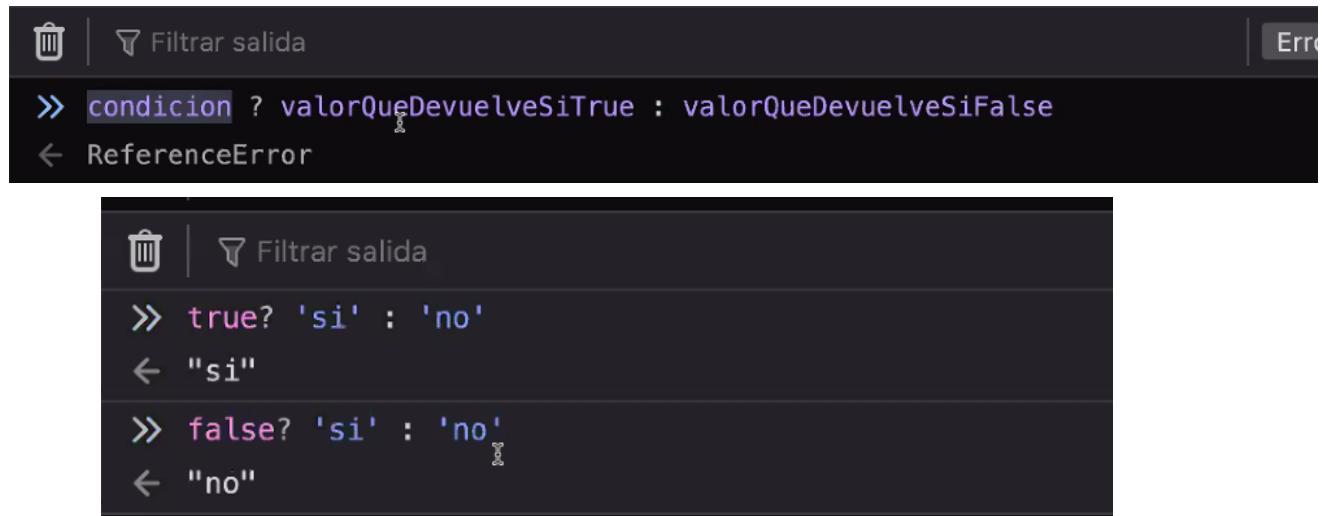
```
Dni = false;  
Pasaporte = false;  
identificado = valor1 || valor2; // resultado = false
```

Veamos otro ejemplo, pedimos al usuario la nota de un examen, cómo la nota va del 0 al 10 primero comprobaremos si efectivamente la nota introducida se encuentra en ese rango. Si la nota es menor que 0 o mayor que 10 no se encontrará en ese rango y, por lo tanto, le mostraremos un mensaje indicando que la nota introducida es incorrecta.

```
let nota = parseInt(window.prompt('Introduce una nota'));  
  
if(nota < 0 || nota > 10) {  
    window.alert('No entiendo esa nota');  
}
```

## Condicional ternario

El nombre viene porque es el único operador que coge 3 operandos. El primero sería una condición (un valor booleano) seguido del símbolo de interrogación y luego, el valor que devuelve si la condición es true dos puntos y el valor que devuelve si la condición es false.



The screenshot shows two terminal windows demonstrating the ternary operator. The top window shows a syntax error for a variable named 'condicion'. The bottom window shows two successful evaluations of the ternary operator.

```
Filtrar salida  
» condicion ? valorQueDevuelveSiTrue : valorQueDevuelveSiFalse  
← ReferenceError
```

```
Filtrar salida  
» true? 'si' : 'no'  
← "si"  
» false? 'si' : 'no'  
← "no"
```

```

<script>
    let edad = parseInt(window.prompt('¿Cuantos años tienes?'));
    let mensaje;
    if(edad >= 18) {
        mensaje = 'Eres mayor de edad';
    } else {
        mensaje = 'Eres menor de edad';
    }
    window.alert(mensaje);

    let edad = parseInt(window.prompt('¿Cuantos años tienes?'));
    let mensaje = edad >= 18 ? 'Eres mayor de edad' : 'Eres menor de edad';
</script>

```

```

    | Filtrar salida
    | >> edad
    | < 40
    | >> mensaje
    | < "Eres mayor de edad"
    | >> window.edad
    | < 40
    | >> window.mensaje
    | < "Eres mayor de edad"
    | >>

```

## Comentarios en Javascript

```

    |      /*
    |      asdfasdf
    |      fasdfa
    |      fasdfas
    |      */

```

```

let nota = parseInt(window.prompt('Qué nota has sacado')); // aqui un comentario

```

Dos opciones:

- desde `/*` y acaba en `*/` → Para agrupar varias líneas en el comentario
- opción `//` → señalaría una línea de comentario empezando desde el momento que se indican las barras.

