JavaScript

Condicionales operadores y arrays

Condicionales

Llegó el momento de tomar decisiones...

If

If

```
if (a > 5) {
        console.log("El número es mayor a 5");
}
```

Hasta ahora vimos que el código que se ejecuta línea a línea, una detrás de otra. Pero a veces se hace necesario romper esa secuencia y crear posibilidades que nos permitan tomar **diferentes caminos** en el código dependiendo de ciertas **condiciones**.

If

```
if (a > 5) {
        console.log("El número es mayor a 5");
}
```

Hasta acá sabemos que si a es mayor a 5, es decir tiene un valor mínimo de 6, se ejecutará el console.log.

Pero, ¿qué pasa si a no es mayor a 5?

If

```
if (a > 5) {
        console.log("El número es mayor a 5");
}
```

Hasta acá sabemos que si a es mayor a 5, es decir tiene un valor mínimo de 6, se ejecutará el console.log.

Pero, ¿qué pasa si a no es mayor a 5? El código seguirá normalmente sin ejecutarse el console.log.

If

```
if (a > 5) {
        console.log("El número es mayor a 5");
}
```

¿Y si quiero que **también se ejecute** otra cosa cuando a sea ***menor*** a 5?

Else

```
if (a > 5) {
        console.log("El número es mayor a 5");
} else {
    console.log("El número es menor o igual 5");
}
```

¡Ahora el código tiene **dos caminos** a elegir! Uno está dado por nuestra **condición** y el otro se ejecutará sólo si nuestra condición no es verdadera.

Else

```
if (a > 5) {
        console.log("El número es mayor a 5");
} else {
    console.log("El número es menor o igual 5");
}
```

¿Puedo tener más de una condición?

If else

```
if (a > 5) {
        console.log("El número es mayor a 5");
} else if(a < 5) {
    console.log("El número es menor a 5");
} else {
    console.log("El número es igual a 5");
}
```

El **else if** añade una condición que se leerá en caso de que la primera no sea verdadera.

En resumen

if: se usa para ejecutar un bloque código si la condición es verdadera.

if…else: se usa para ejecutar un bloque de código si la condición es verdadera u otro bloque de código si la condición es falsa.

if…else if…else: se usa para seleccionar uno de los muchos bloques de código a ejecutar.

Anidación

```
if (a < 5) {
        if (a === 1) {
                alert("El número es menor a 5 y es 1");
    } else {
        alert("El número es menor a 5, pero no es 1");
    }
}
```

Podemos anidar tantos **if** como creamos necesarios.

Operador condicional ternario

```
let edad = 17;

let esMayorEdad = edad > 18 ? true : false;
```

Es el único operador en JavaScript que tiene tres operandos. Se usa como atajo para la instrucción **if**.

¡A practicar!

Consigna: Crea un algoritmo que solicite al usuario uno o más valores ingresados por prompt(), compare las entradas y, en función de ciertas condiciones, muestre por consola o alert() el resultado según los valores ingresados y las condiciones cumplidas.

Ejemplo:

- Pedir número mediante prompt y si es mayor a 1000 mostrar un alert.
- Pedir un texto mediante prompt, y si es igual a "Hola" mostrar un alerta por consola.
- Pedir un número por prompt y evaluar si está entre 10 y 50. En caso positivo mostrar un alert.

Arrays

Hora de ver cómo podemos tener un grupo de variables.

Arrays

En javascript un array es un objeto especial que permite crear grupos ordenados de datos y nos provee herramientas para trabajar con ellos.

Los arrays pueden agrupar cualquier tipo de dato.

¿Cómo se definen?

Un array se define utilizando "[]" alrededor de los elementos que contenga.

Se puede definir un array sin elementos utilizando simplemente "[]".

Un array vacío:

```
let grupo = [];
```

Un array con elementos:

```
let mascotas = ['gato', 'perro', 'tarantula']; //array de strings
```

Todo tipo de datos

Un array puede almacenar todo tipo de dato

```
// Array de Strings
let mascotas = [‘gato’, ‘perro’, ‘tarantula’];

// Array de numerales
let numeros = [4, 8, 10, 3];

// Array de booleans
let verdades = [true, false, true];
```

Pueden ser de distinto tipo

Un array en javascript no necesita que los elementos que contiene sean del mismo tipo entre ellos.

```
let persona = [26592093, ‘Perico’, ‘Perez’, 1981, false];
```

¡Arrays de arrays!

Como vimos un array puede contener cualquier dato, incluso otros arrays!

```
let resultados = [
    [‘gato’, ‘miau’],
    [‘perro’, ‘guau’],
    [‘ardilla’, ‘wheee’]
];
```

¿Cómo se acceden?

Los arrays aseguran un orden de los datos que contienen y nos permiten acceder a sus distintos valores por su ***índice***.

En javascript ***el primer elemento está en el índice 0 (cero)***.

Para especificar el índice del array lo especificamos entre "[]" inmediatamente después del nombre de la variable.

```
let datos = ['dato1', 'dato2', 'dato3'];
console.log(datos[0]);

'dato1'
```

La propiedad 'length'

A veces necesitamos saber que tantos elementos tiene un array y la propiedad 'length' provee específicamente ese valor.

```
let colorMascotas = [
    ['perro', 'marron'],
    ['gato', 'blanco'],
    ['pez', 'anaranjado']
];

console.log(colorMascotas.length);                    '3'
```

Crea un array de strings con 5 elementos que quieras.
Luego, mostrá en consola:

1. El array
2. Cuántos elementos tiene el array
3. El índice 0 del array

¡A practicar!

Crear una **lista** de artículos para el supermercado de mínimo 5 items.

Con el método **prompt()** pedile al usuario que ingrese un artículo para agregar a la lista.

Si el artículo ya está en la lista (método: **array.includes()**) avisarle al usuario que el artículo ya está.

Si no, agregar el artículo a la lista y mostrarle al usuario la lista de todo lo que tiene que comprar.

Switch

Switch

```
let fruta = "manzana";
switch (fruta) {
    case "manzana":
      color = "Rojo";
     break;
    case "kiwi":
      color = "Verde";
     break;
     default:
           color = "Blanco";
}
```

Si bien **if...else** va a ser efectivo en muchos casos, a veces vamos a necesitar realizar varias elecciones.

Switch

```
let fruta = "manzana";
switch (fruta) {
    case "manzana":
      color = "Rojo";
     break;
    case "kiwi":
      color = "Verde";
     break;
     default:
           color = "Blanco";
}
```

Switch nos evita usar muchos if..else juntos cuando hay muchas decisiones que tomar.

Switch

En todos los casos hay que cuidar de no tener muchos ifs anidados o switchs con muchas cláusulas. Ya que hace más difícil entender (y mantener) el código.

¡A practicar!

El método **new Date().getDay()** nos devuelve un número que nos indica qué día es (domingo = 0).

Hacé un **switch** que nos devuelva el nombre del día según el número.

Mostrá en consola o en el documento qué día es hoy.

Operadores de comparación

Igual (==) / Estrictamente igual (===)

```
let a = 3;

a == ‘3’ //True
a == 3 //True
a === ‘3’ //False
a === 3 //True
```

¿Qué está pasando?

Igual (==) / Estrictamente igual (===)

```
let a = 3;

a == ‘3’ //True
a == 3 //True
a === ‘3’ //False
a === 3 //True
```

El operador **==** compara si son iguales, en cambio el operador **===** compara si son iguales y del mismo tipo.

Desigual (!=) / Estrictamente desigual (!==)

```
let a = 3;

a != ‘3’ //False
a != 3 //False
a !== ‘3’ //True
a !== 3 //False
```

Como vemos, ocurre lo mismo que con los operadores **==** y **===.** Son estrictamente desiguales cuando son de tipo diferente.

Mayor (>)

```
let a = 3;

a > ‘4’ //False
a > 2 //True
a > 3 //False
```

Devuelve true si el operando izquierdo es mayor que el derecho. Como vemos, no importa el tipo de dato a comparar.

Menor (<)

```
let a = 3;

a < ‘4’ //True
a < 2 //False
a < 3 //False
```

Devuelve true si el operando izquierdo es menor que el derecho. Al igual que en mayor, no importa el tipo a comparar.

Mayor igual (>=)

```
let a = 3;

a >= ‘4’ //False
a >= 3 //True
```

Es igual que el mayor pero también tiene en cuenta si es igual.

Menor igual (<=)

```
let a = 3;

a <= ‘4’ //True
a <= 3 //True
```

Es igual que el menor pero también tiene en cuenta si es igual.

¡A practicar!

Consigna

Usando el método **prompt** pedile al usuario que ingrese un número.

Definí 3 caminos:
Si el número es mayor a 10.
Si el número es menor o igual a 10.
Si no es un número.

En cada camino, usá el método **document.write** para dejar un mensaje según la elección.

Operadores lógicos

AND (&&)

```
let mascota = “Perro”;
let edad = 1;
if (mascota == “Perro” && edad < 2) {
        console.log("Tu perro es cachorro");
}
```

Retorna true si *todos sus argumentos son true*. De lo contrario retorna false.

OR (||)

```
let mascota = “Caballo”;
if (mascota == “Perro” || mascota == “Gato”) {
        document.write("Tu mascota será bienvenida al
alojamiento.");
} else {
    document.write("Lo sentimos, solo recibimos perros o gatos");
}
```

Retorna true si *cualquiera de sus argumentos es true*. De lo contrario retorna false.

NOT (!)

```
let tieneMascota = false;

if (!tieneMascota ) {
        console.log("No tenés mascotas.");
}
```

Convierte el operando al tipo booleano: true/false y retorna el valor contrario.

Operadores en JS

OPERADORES LÓGICOS Y RELACIONALES	DESCRIPCIÓN	EJEMPLO
==	Es igual	a == b
===	Es estrictamente igual	a === b
!=	Es distinto	a != b
!==	Es estrictamente distinto	a !== b
<, <=, >, >=	Menor, menor o igual, mayor, mayor o igual	a <=b
&&	Operador and (y)	a && b
||	Operador or (o)	a || b
!	Operador not (no)	!a

¡A practicar!

Mariana nos dice que para ella un número es de la suerte si ***cumple con las siguientes tres condiciones***:

- ★ ese número es positivo
- ★ ese número es múltiplo de 2 o de 3
- ★ ese número no es el 15

Escribí la función **esNumeroDeLaSuerte()** la cual recibiendo un número, le diga a Mariana si es un número de la suerte (el número debe cumplir con las tres condiciones antes mencionadas).

Pista:

Para saber si un número es múltiplo de otro usamos el **operador módulo (%)** que devuelve el resto de la división entera entre dos números.

Entonces...

11 % 5 --> devuelve el resto de dividir 11 por 5. En este caso devuelve 1. Como es distinto de 0, 11 NO es múltiplo de 5.

12 % 3 --> dará 0. Esto implica que 12 es múltiplo de 3.

¿Preguntas?

¡Gracias!