OPERADORES EN JAVASCRIPT

Al igual que Java tendremos operadores para trabajar con datos, aunque hay algunos operadores que son distintos a los que conocemos en Java.

OPERADOR DE ASIGNACIÓN	
=	Operador de Asignación Simple
OPERADORES ARITMÉTICOS	
+	Operador de Suma
-	Operador de Resta
*	Operador de Multiplicación
**	Exponenciación
/	Operador de División
%	Operador de Módulo
OPERADORES UNARIOS	
++	Operador de Incremento.
--	Operador de Decremento.
+=	y += x

-=	y -= x
*=	y *= x
/=	y /= x
%=	y %= x
**=	y **= x

OPERADORES LOGICOS Y RELACIONALES

==	Es igual Ejemplo: 3 == "3"
===	Es estrictamente igual Ejemplo: 3 === 3
!=	Distinto
!==	Estrictamente Distinto
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que

OPERADORES CONDICIONALES	
&&	AND
||	OR
!	Operador Lógico de Negación.

OPERADORES DE COMPARACIÓN DE TIPO	
typeof	Devuelve el tipo de dato de una variable
Instanceof	Devuelve true si el objeto es una instancia de.

Los operadores en JavaScript se utilizan para realizar operaciones sobre valores y variables.

- Operadores Aritméticos: Estos operadores se utilizan para realizar operaciones matemáticas comunes.
- Operadores de Asignación: Estos operadores se utilizan para asignar valores a las variables.
- Operadores de Comparación: Estos operadores se utilizan para comparar dos valores y devolver un valor booleano (true o false).
- Operadores Lógicos: Estos operadores se utilizan para realizar operaciones lógicas en valores booleanos.
- Operadores Bit a Bit: Estos operadores trabajan a nivel de bits y se utilizan para realizar operaciones bit a bit.
- Operadores de Tipo: Estos operadores se utilizan para identificar el tipo de una variable.
- Operador Ternario: Este operador se utiliza como una forma compacta de la declaración if...else.

? : (Operador Ternario): Evalúa una condición y devuelve uno de dos valores.

```
let esMayor = (edad >= 18) ? "Mayor de edad" : "Menor de edad";
```

TYPEOF

La función typeof se utilizada para obtener el tipo de dato que tiene una variable.

```
console.log(typeof 42);            // expected output: "number"
console.log(typeof 'blubber');     // expected output: "string"
console.log(typeof true);          // expected output: "boolean"
```

CONDICIONALES EN JAVASCRIPT

Existen los condicionales que nos van a ayudar a modificar el flujo de ejecución del programa.

IF

El condicional if es un condicional lógico que evalúa el camino a tomar en base a la evaluación de una condición. Supongamos el siguiente ejemplo, mi sobrino quiere subirse a una montaña rusa, pero para ello tiene que **aprobar las dos siguiente condiciones:** tener mas de 18 años y medir mas de 160 cm. La evaluación de esas dos condiciones da por verdadero se podrá subir de lo contrario no podrá.

```
let edad = 15;
let altura = 166;
if (edad>18 && altura>160) {
console.log("Puedes subirte :D");
} else {
console.log("No te podes subir”);
}
```

Como se puede ver, si la condición a evaluar se cumple, es decir, da verdadero, mostrara el mensaje “Puedes subirte :D”, en caso que de falso mostrara “No te puedes subir ”. Por otra parte, **JavaScript permite también agregar la condición else if**

En JavaScript, `else if` **es una construcción utilizada en las estructuras de control condicionales para manejar múltiples condiciones**. Permite evaluar varias expresiones booleanas en secuencia y ejecutar diferentes bloques de código dependiendo del resultado de esas evaluaciones.

La estructura básica de una declaración `if...else if...else` es la siguiente:

```
if (condicion1) {
  // Código a ejecutar si condicion1 es verdadera

} else if (condicion2) {
  // Código a ejecutar si condicion1 es falsa y condicion2 es verdadera

} else if (condicion3) {
  // Código a ejecutar si condicion1 y condicion2 son falsas, y condicion3 es verdadera

} else {
  // Código a ejecutar si todas las condiciones anteriores son falsas
}
```

IF TERNARIO

El **if** ternario nos permite resolver en una línea una expresión lógica asignando un valor. Proviene del lenguaje C, donde se escriben muy pocas líneas de código y donde cuanto menos escribamos más elegantes seremos. Este operador es un claro ejemplo de ahorro de líneas y caracteres al escribir los scripts. Lo veremos rápidamente, pues la única razón que lo veamos es para que sepan que existe y si lo encuentran en alguna ocasión sepan identificarlo y cómo funciona.

Variable = (condición) ? valor1 : valor2

Este ejemplo no sólo realiza una comparación de valores, además asigna un valor a una variable. Lo que hace es evaluar la condición (colocada entre paréntesis) y si es positiva asigna el valor1 a la variable y en caso contrario le asigna el valor2.

Veamos un ejemplo:
momento = (hora_actual < 12) ? "Antes del mediodía" : "Después del mediodía"

SWITCH

La declaración switch evalúa una expresión, comparando el valor de esa expresión con una instancia case, y ejecuta declaraciones asociadas a ese case, así como las declaraciones en los case que siguen.
El programa primero busca la primer instancia case cuya expresión se evalúa con el mismo valor de la expresión de entrada (usando comparación estricta, ===) y luego transfiere el control a esa cláusula, ejecutando las declaraciones asociadas. Si no se encuentra una cláusula de case coincidente, el programa busca la cláusula default opcional, y si se encuentra, transfiere el control a esa instancia, ejecutando las declaraciones asociadas. Al igual que Java, la declaración break es opcional y está asociada con cada etiqueta de case y asegura que el programa salga del switch una vez que se ejecute la instrucción coincidente y continúe la ejecución en la instrucción siguiente. Si se omite el break el programa continúa la ejecución en la siguiente instrucción en la declaración de switch .

```
switch (expr) {
case 'Naranjas':
console.log('El kilogramo de naranjas cuesta $0.59.');
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 +
'.');
}
```

ESTRUCTURAS REPETITIVAS

Veamos como son las estructuras repetitivas en JavaScript

WHILE

Crea un bucle que ejecuta una sentencia especificada mientras cierta condición se evalúe como verdadera. Dicha condición es evaluada antes de ejecutar la sentencia

```
let a = 0;
while(a != 10){
console.log(++a);
}
```

DO WHILE

La sentencia (hacer mientras) crea un bucle que 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 a = 0;
do{
console.log(++a);
}while(a!=10);
```

FOR

El bucle FOR se utiliza para repetir una o más instrucciones un determinado número de veces. De entre todos los bucles, el FOR se suele utilizar cuando sabemos seguro el número de veces que queremos que se ejecute. La sintaxis del bucle for va a ser la misma que en Java:

```
for ([expresion-inicial]; [condicion]; [expresion-final]){
}
for(let i = 0; i < 10; i++){
console.log("El valor de i es " + i);
}
```

BREAK

Termina el bucle actual, sentencia switch o label y transfiere el control del programa a la siguiente sentencia.

```
for (let i = 0; i < 10; i++) {
if(i == 5){
break;
}
console.log("Estamos por la vuelta "+i);
}
```

CONTINUE

Termina la ejecución de las sentencias de la iteración actual del bucle actual o la etiqueta y continua la ejecución del bucle con la próxima iteración.

En contraste con la sentencia break, continue no termina la ejecución del bucle por completo; en cambio,

- En un bucle while, salta de regreso a la condición.
- En un bucle for, salta a la expresión actualizada.

La sentencia continue puede incluir una etiqueta opcional que permite al programa saltar a la siguiente iteración del bucle etiquetado en vez del bucle actual. En este caso, la sentencia continue necesita estar anidada dentro de esta sentecia etiquetada.

```
for (let i = 0; i < 10; i++) {
if(i == 5){
continue;
}
console.log("Estamos por la vuelta " + i);
}
```

LABEL

Proporciona a una sentencia con un identificador al que se puede referir al usar las sentencias break o continue. Por ejemplo, puede usar una etiqueta para identificar un bucle, y entonces usar las sentencias break o continue para indicar si un programa debería interrumpir el bucle o continuar su ejecución.

label o etiqueta : sentencia

Ejemplo:

```
exterior: for (let i = 0; i < 10; i++) {
for (let j = 0; j < 10; j++) {
if(i == 4 && j == 4){
console.log("Vamos a cortar ambos for");
break exterior;
}
console.log(i+j+10*i);
}}
```