PEC 1: Conceptualización e investigación de usuarios

NOTA: Los programas deben tener en las primeras líneas la asignación de las variables que contengan los valores de los enunciados, de manera que cambiando estos valores los resultados del programa cuando se ejecute sean correctos en cualquier caso. Los resultados deben mostrarse por consola.

1. Actividad 1

Analizad el siguiente programa en *JavaScript* y calculad qué valores acabarán teniendo las variables. Debéis considerar las ocho líneas como un único programa: cada expresión puede contener variables cuyos valores se han calculado en líneas anteriores. Explicad el porqué de cada resultado:

Resolución del ejercicio:

```
let x01=0; // Aquí se le asigna cero (número) como valor a la variable x01

let x02=(x01=="0"); /* x02 es true porque los operandos son iguales después de la conversión de tipos que hace JavaScript
                       antes de comparar.
                       Si los operandos son de diferente tipos, intenta convertirlos al mismo tipo antes de comparar.
                       Al comparar un número con string, convierte al string en un valor numérico.
                       Por ejemplo:
                       '3' == var1
                       3 == '3'*/

let x03=(x01===0); // x03 es True porque hay una comparación estricta 0 === 0, y hasta ahora x01 = 0 (número)

let x04=(x01="0"); // Aquí x04 toma el valor de x01, que a su vez se le asigna el valor "0" (string) al utilizar un operador de asignación

let x05=(x04=="0"); // X05 es True porque x04 = "0", asignado arriba

let x06=(x04==="0"); // x06 es True porque hay una comparación estricta "0" === "0" (ambos string)

let x07=(x04===0); // x07 es False porque al hacer comparación estricta string === número, los operadores NO son del mismmo tipo

let x08=(x04==0); /* x08 es true porque nos pasa lo mismo que x02.
                       Los operandos son iguales porla conversión de tipos que hace JavaScript antes de comparar.
                       Si los operandos son de diferente tipos, intenta convertirlos al mismo tipo antes de comparar.
                       Al comparar un número con string, convierte al string en un valor numérico.
                       Por ejemplo:
                       '3' == var1
                       3 == '3'*/
```

let x01=0; // Aquí se le asigna cero (número) como valor a la variable x01

let x02=(x01=="0"); /* x02 es true porque los operandos son iguales después de la conversión de tipos que hace JavaScript antes de comparar. Si los operandos son de tipos diferentes, intenta convertirlos al mismo tipo antes de comparar. Al comparar un número con un string, convierte al string en un valor numérico.
Por ejemplo: '3' == var1
3 == '3'* /

let x03=(x01===0); // x03 es True porque hay una comparación estricta 0 === 0, y hasta ahora x01 = 0 (número)

let x04=(x01="0"); // Aquí x04 toma el valor de x01, que a su vez se le asigna el valor "0" (string) al utilizar un operador de asignación

let x05=(x04=="0"); // X05 es True porque x04 = "0", asignado arriba

let x06=(x04==="0"); // x06 es True porque hay una comparación estricta "0" === "0" (ambos string)

let x07=(x04===0); // x07 es False porque al hacer comparación estricta string === número, los operadores NO son del mismo tipo

let x08=(x04==0); /* x08 es true porque nos pasa lo mismo que x02.
Los operandos son iguales debido a la conversión de tipos que realiza JavaScript antes de comparar.
Si los operandos son de tipos diferentes, intenta convertirlos al mismo tipo antes de comparar.
Al comparar un número con string, convierte al string en un valor numérico.
Por ejemplo:
'3' == var1
3 == '3' */