CUESTIONARIO TEÓRICO DE OPERADORES LÓGICOS

1. **¿Qué tienen en común los operadores lógicos con los de comparación?**

   A) Son case sensitive.

   B) Solo pueden utilizarse con tipo de datos numéricos.

   C) Ambos devuelven un valor booleano como resultado.

   Los operadores lógicos y los operadores de comparación comparten la característica de devolver un valor booleano como resultado. Los operadores lógicos, como AND (`&&`), OR (`||`) y NOT (`!`), trabajan con expresiones lógicas y devuelven `true` o `false` según la evaluación de esas expresiones. Los operadores de comparación, como igualdad (`==` o `===`), desigualdad (`!=` o `!==`), mayor que (`>`), menor que (`<`), etc., comparan dos valores y también devuelven un valor booleano como resultado.

2. **¿En qué se diferencia la comparación simple de la estricta?**

   A) No tienen diferencia, solo son convenciones.

   B) La comparación estricta solo compara el valor de dato. La débil compara tipo de dato y valor.

   C) La comparación simple compara solo el tipo de dato. La estricta compara tipo de dato y valor.

   D) La comparación estricta compara el valor y el tipo de dato, mientras que la débil solo el valor.

   En JavaScript, la comparación simple (también conocida como comparación débil) utiliza el operador `==` y realiza la comparación teniendo en cuenta la conversión de tipos. Por otro lado, la comparación estricta utiliza el operador `===` y compara tanto el valor como el tipo de dato sin realizar conversiones.

3. **¿Cuál es la función del símbolo "!"?**

   A) Remarca el código para que se lea primero.

   B) Es el símbolo de negación, niega cualquier condición o sentencia que hagamos.

   C) Ambas son correctas.

   En JavaScript, el símbolo "!" se utiliza como operador de negación lógica. Si se coloca antes de una expresión, invierte su valor booleano. Por ejemplo, `!true` dará como resultado `false`, y `!false` dará como resultado `true`.

4. **¿Por qué debemos escribir el símbolo > o < antes de =?**

   A) Para que nuestro código sea legible.

   B) No es necesario, es solo una convención.

   C) Porque de lo contrario JavaScript lee primero el = como asignación y luego no sabe cómo continuar.

   D) Muy bien, es importante respetar las normas de escritura que tiene el lenguaje.

   E) Para evitar comparaciones erróneas.

5. **¿Cuál es la principal diferencia entre el "y lógico" (&&) y el "o lógico" (||) para obtener un resultado "true"?**

   A) Su sintaxis.

   B) En el && ambas condiciones deben ser correctas, mientras que en el || una condición puede ser falsa.

   C) Exactamente, el operador && funciona como un sí o sí, mientras que el operador || es más un "uno o el otro".

   D) Con || ambas sentencias deben ser verdaderas.

   E) No tienen diferencias.

¿Qué retornan las siguientes operaciones lógicas?

1. **false || true**

   A) True

   B) False

   C) Error

   El operador || analiza de izquierda a derecha y retorna cuando encuentra un valor verdadero o el último valor de la sentencia.

2. **false | 3 == 4**

   A) True

   B) False

   C) 0.

   Si bien JavaScript lee e interpreta la sentencia y retorna un 0, hay un error de sintaxis, ya que la forma correcta de usar los operadores lógicos es con símbolos dobles ("||" y "&&").

3. **false && (3 == 4)**

   A) True

   B) False

   C) Error

   El operador && devuelve el primer valor o expresión analizado como false.

4. **10 >= 15 && 10 !== 11**

   A) True

   B) False

   C) Error

   Como ven, no es necesario los paréntesis en esta situación, ya que el operador de comparación es de mayor prioridad que el operador lógico &&.

5. **12 % 2 == 0 && 12 !== 21**

   A) True

   B) False

   C) Error

   ¡Muy bien! El operador de módulo va a devolver siempre el resto de la división, y el operador de negación niega cualquier comparación.

6. **(8-15 == 8 || 7>6 = -2)**

   A) True

   B) False

   C) Error

   ¡Muy bien! Hay que estar atentos a que las comparaciones son siempre con == o ===. Un solo = es simplemente una asignación de un valor a una variable. Por lo tanto, se produce un error al querer asignarle un valor a otro valor, resultando en el error: "SyntaxError: Invalid left-hand side in assignment" que refiere a que no se puede utilizar valores en el lado izquierdo de las asignaciones.

7. **3+5 == "8" && 5-4 === 1**

   A) True

   B) False

   C) Error

8. **'Zapato' == 'trampa' || "hola" <= "chau"**

   A) True

   B) False

   C) Error

   Cuando se comparan strings, el == compara valor y <= compara por orden alfabético. Ejemplo: console.log("a" < "bbbbb"); //true

9. **"Gato" && "Perro"**

   A) Gato

   B) Perro

   C) Error

   En el caso de los strings y el operador &&, al ser ambos strings verdaderos (ya que tienen una cadena de caracteres), la respuesta es el último string de la sentencia.

10. **"Gato" || "Perro"**

    A) Gato

    B) Perro

    C) Error

    A diferencia del operador &&, en este caso —al tener que cumplirse sólo una condición—, como ambos son true, la respuesta es la primera sentencia true encontrada.