JS Fundamental:

NOTE: IF YOU SEE A ‘ ‘IN ANY ANSWER OR EXPLANATION, DO NOT REFER TO ‘ ‘ IN JS SYNTAX. IT HAS NOTHING TO DO WITH JS SYNTAX. ANSWER EXPLANATIONS ARE THOSE STATEMENTS PUT WITHIN BRACKETS ( ).

1. What are the results of these expressions? (answer first, then use console.log() to check)

"" + 1 + 0

//Answer: 10 (It is a normal expression concat in js. Since the string is “” there is no effect. Therefore, the final print is ‘10’)

"" - 1 + 0

//Answer: -1

true + false

//Answer: true (since the ‘true’ value is 1, false is 0, if expressed in value, the result is ‘1’)

!true

//Answer: False (since it is ‘not true’ then it will print ‘False’)

6 / "3"

//Answer: 2 (JS automatically treats “3” as a number although it is a string)

"2" * "3"

//Answer: 6 (JS automatically treats both strings into numbers & multiply them)

4 + 5 + "px"

//Answer: 9px (The “px” is a string and NaN. Therefore, JS will concat it to the prioritised addition operation at the beginning of the expression)

"$" + 4 + 5

//Answer: $45 (The “$” is NaN. JS will concat it. The ‘4’ and ‘5’ are numbers, but they are not prioritised mathematically, so JS will concat them)

"4" - 2

//Answer: 2 (While “4” is a string, it is still a number. Therefore, JS will automatically treat it as a number and continue with the math operation)

"4px" - 2

//Answer: NaN (JS cannot concat “4px” to ‘2’ as the expression is not using a string operator, which is ‘+’)

" -9 " + 5

//Answer: -9   5 (The “-9” is a string that contains a character other than numbers, i.e. the negative sign, at the beginning of the expression. Therefore, it will be treated as a string. As the next operator is ‘+’ operator, JS will concat it)

" -9 " - 5

//Answer: -14 (Although the “-9” is a string that contains a character other than numbers, i.e. the negative sign, at the beginning of the expression, it is followed by another negative number. JS will treat both as negative numbers. The expression then is equal to: -9+-5, which results in ‘-14’)

null + 1

//Answer: 1 (The null means an absence of value in JS. Therefore, the result is ‘1’)

undefined + 1

//Answer: NaN (In JS, undefined means no assigned value. Therefore, when undefined is used in an expression, such as in the question, it has no assigned value. Since it is also not a string, JS can neither concat nor decide that ‘1’ is the result. The result is then thrown as ‘NaN’)

undefined == null

//Answer: true (In JS, undefined means no assigned value. Since the operator ‘==’ only refers to the value, not the type, the ‘undefined’ can pass as null. But it does not mean that null is the same as ‘undefined’).

undefined === null

//Answer: false (In JS, undefined means no assigned value. The operator ‘===’ will check not only the value but also the type. As null is not the same by type with ‘undefined’, the answer is false)

" \t \n" – 2

//Answer: -2 (The “\t” is the tab character, and the “\n” is the new line character. These are not a string either. Therefore, JS will only read the ‘-2’ as a result).

NOTE FOR QUESTION 1: The console.log file is provided as a vs file in the same folder.

2. Which of the following answers does not give the right answer? Why are they not correct? How can we fix them?

let three = "3"

let four = "4"

let thirty = "30"

//what is the value of the following expressions?

let addition = three + four

//Answer: 34 (This answer is incorrect. The variable name 'addition' suggests that we will perform addition, although the numbers are automatically converted from strings by JS and treated as a concat).

let multiplication = three * four

//Answer: 12 (This answer is correct)

let division = three / four

//Answer: 0.75 (This answer is correct)

let subtraction = three - four

//Answer: -1 (This answer is correct)

let lessThan1 = three < four

//Answer: True (This answer is correct)

let lessThan2 = thirty < four

//Answer: True (This answer is mathematically incorrect as thirty is not less than four. However, since thirty has never been defined before, it becomes undefined. JS will read it as less than the right-hand side of the comparison. Therefore, it is printed as 'True' by JS)

3. Which of the following console.log messages will print? Why?

if (0) console.log('#1 zero is true')

//Answer: Will not print. Implicit conversion to false.

if ("0") console.log('#2 zero is true')

//Answer: Will print (This will print as the zero under the conditional statement is not an implicit conversion to false. It has value and type)

if (null) console.log('null is true')

//Answer: Will not print. (The null is an intentional absence of value. Furthermore, written as solely null tells JS that it is not an identifier and it points to no object)

if (-1) console.log('negative is true')

//Answer: Will print (This will print because JS will read it as having a value, and therefore, it will print)

if (1) console.log('positive is true')

//Answer: Will print (This will print because JS will read it as having a value, and therefore, it will print)