**JavaScript Day 2 Assignment**

Q1. What will the code below output to the console and why?

console.log (1 + “2” + “2”);

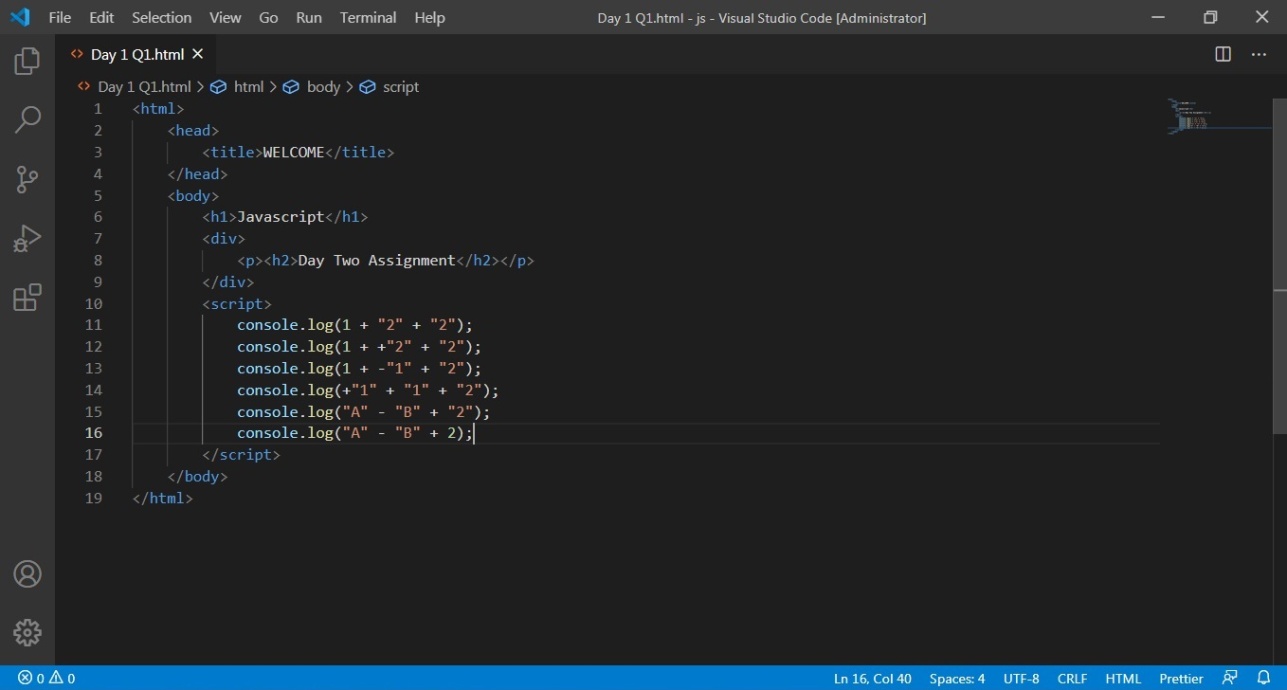
console.log (1 + +“2” + “2”);

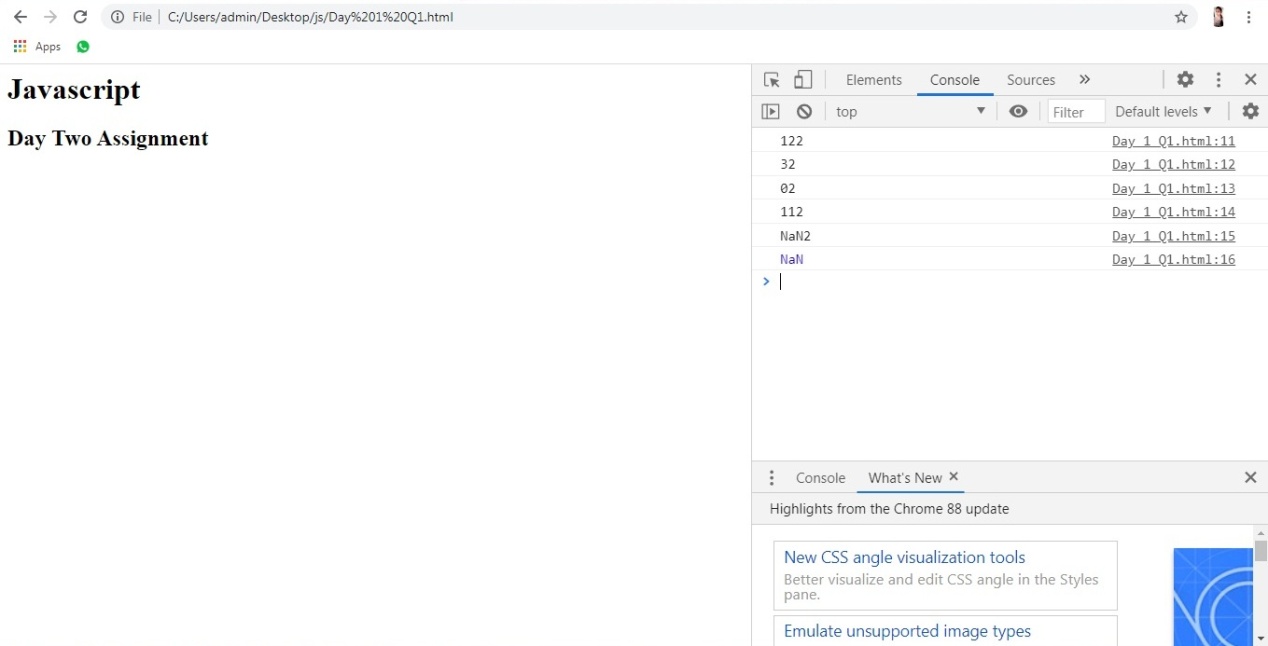
console.log (1 + -“1” + “2”);

console.log (+”1” + “1” + “2”);

console.log (“A” - “B” + “2”);

console.log (“A” - “B” + 2);





Here’s why…

The fundamental issue here is that JavaScript (ECMAScript) is a loosely typed language and it performs automatic type conversion on values to accommodate the operation being performed. Let’s see how this plays out with each of the above examples.

Example 1: 1 + “2” + “2” Outputs: “122” Explanation: The first operation to be performed in 1 + “2”. Since one of the operands (“2”) is a string, JavaScript assumes it needs to perform string concatenation and therefore converts the type of 1 to “1”, 1 + “2” yields “12”. Then, “12” + “2” yields “122”.

Example 2: 1 + +”2” + “2” Outputs: “32” Explanation: Based on order of operations, the first operation to be performed is +”2” (the extra + before the first “2” is treated as a unary operator). Thus, JavaScript converts the type of “2” to numeric and then applies the unary + sign to it (i.e., treats it as a positive number). As a result, the next operation is now 1 + 2 which of course yields 3. But then, we have an operation between a number and a string (i.e., 3 and “2”), so once again JavaScript converts the type of the numeric value to a string and performs string concatenation, yielding “32”.

Example 3: 1 + -“1” + “2” Outputs: “02” Explanation: The explanation here is identical to the prior example, except the unary operator is – rather than +. So “1” becomes 1, which then becomes -1 when the – is applied, which is then added to 1 yielding 0, which is then converted to a string and concatenated with the final “2” operand, yielding “02”.

Example 4: +”1” + “1” + “2” Outputs: “112” Explanation: Although the first “1” operand is typecast to a numeric value based on the unary + operator that precedes it, it is then immediately converted back to a string when it is concatenated with the second “1” operand, which is then concatenated with the final “2” operand, yielding the string “112”.

Example 5: “A” - “B” + “2” Outputs: “NaN2” Explanation: Since the – operator can not be applied to strings, and since neither “A” nor “B” can be converted to numeric values, “A” - “B” yields NaN which is then concatenated with the string “2” to yield “NaN2”.

Example 6: “A” - “B” + 2 Outputs: NaN Explanation: As explained in the previous example, “A” - “B” yields NaN. But any operator applied to NaN with any other numeric operand will still yield NaN.