

1. Did any of the typof outputs surprise you?
   1. Yes! When I originally wrote the code (missing the “typeof” as we discussed via email) it put out the code as it did with the statements in part 2. (i.e. displayed Hi Lindsey! ,4.5, 0, -10, true) but when I added in the “typeof” with the variables, it now only displays the “string” in the terminal.
2. What is happening in step 5.1 that is causing a line break?
   1. After doing a little bit of external research, I believe that the break is being caused by what is called an “escape” which is the( \n ) in the statement.
3. What do you think is happening on 5.2?
   1. After doing some investigation and research external to our reading provided, I believe this is called a Template Literal. “To embed JavaScript expressions or variables with the help of the ${…} syntax.
   2. <https://www.programiz.com/javascript/template-literal#:~:text=Example%3A%20JavaScript%20Template%20Literals,is%20%24%7Bnumber1%20%2B%20number2%7D>.
   3. <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals>
4. Tell me why the output is what it is for 5.2-5.8
   1. 5.2= Division of 100 by 2 is 50
   2. 5.3= The ( + ) operator links the values within the “…” together
      1. Concatenate: link (thinks) together in a chain or series.
   3. 5.4= 8 X 0 = 0 , null is treated as a 0
   4. 5.5= “5” – 1=4….”5” = 5
   5. 5.6= “5” + 1 = 51---due to concatenation, rather than output being 6
   6. 5.7= Unlike the last 2 statements, “5” is shown as “five” \* 2 in this statement, since there are no declarations saying that “five” should = “5” , the system is unable to recognize it as a number within the equation?
   7. 5.8 = False does not equal True, so output is False