# JavaScript Strings Assignment

# More practice with strings

## Product Code Validator

Write a program that asks the user to enter a product code for a product manufactured by the Wacky Wally Widget Company. Your program should inspect the product code provided by the user and respond with the string "Valid" or "Not Valid".

Valid product codes are of the pattern 999-AAA where "999" is any sequence of three digits and "AAA" is any sequence of three alphabet characters. The Wacky Wally Widget Company treats upper and lower-case letters in their product codes as the same. That is, "AAA-999" is valid, and Aaa-999 is valid, and aaa-999 is also valid and they all refer to the same product.

## Getting started (Together as a class)

1. Retrieve assignment16\_productcode.zip provided by your instructor.
2. Extract the code into your mis2402workspace and open the index.html file in Visual Studio Code.
3. Write the code in isValidProductCode that will test to see if the parameter text contains a valid product code. The function should return the text “Valid” or “Not Valid” appropriately.

NOTE: There are lots of acceptable ways to solve this problem. But this exercise is meant to give you a chance to put your knowledge various string properties to work. So, any approach that requires you to use a string method that has not been covered in class is not advised!

1. Here are some tips to help you work through this problem.

|  |
| --- |
| * 1. It would be smart to convert the whole input string to upper of lower case.
	2. How long is a valid product code? You can check for that. If the value of text is the wrong length, then return “Not Valid”. (Recall that using “return” causes the function to quit immediately!)
	3. What character should appear in the middle of a valid product code? Check for that. If that character is not the expected value, then return “Not Valid”.
	4. Check each of the first three characters of the text variable. Are they digits? (The parseInt() function can help you do that. If the parseInt(x) is equivalent to x, then x is an integer.
 |
| * 1. Check the last three characters of the text variable. You can use comparison to check a string. For example, ‘a’ is less than ‘b’ and ‘Z’ is greater than ‘M’. Look – you can prove it!
 |  |

1. Test your work. Sample screenshots are shown below.

 

 



CONTNUED

1. Upload your work. Be sure that you can find your work on the class server by typing in its URL in the browser. Test your work \*again\* on the class server.

For example:
http://misdemo.temple.edu/tux99999/assignment16\_productcode

On your own

Palindrome Checker

A palindrome word is a word that is spelled the same backwards and forwards.

Some valid palindrome words: Racecar, madam, Kayak

Notice that in the above examples the case of the letters does not matter ... Racecar is still a palindrome word even though the "R" and "r" are technically different characters.

Write a program that checks to see if a word is a palindrome.

You can be very liberal in your definition of a "word" here... a word is just any pattern of characters. So, abcdef and abccba are both words. The word abcdef is not a palindrome. The word abccba is a palindrome.

The isPalindrome function you write should return a string. The string should be either "Palindrome" or "Not a palindrome"

1. Retrieve assignment16\_palindrome.zip provided by your instructor.
2. Extract the code into your mis2402workspace and open the index.html file in Visual Studio Code.
3. Complete the isPalindrome function so that it returns either “Palindrome” or “Not a palindrome” appropriately.
4. Test your work.
5. Upload your work. Be sure that you can find your work on the class server by typing in its URL in the browser. Test your work \*again\* on the class server.

For example:
http://misdemo.temple.edu/tux99999/assignment16\_palindrome

How will this assignment be graded?

This assignment will be evaluated by an automated process.

* If your work is not found at the expected location on misdemo, you will get a score of **zero**.
* If your work generates **all output** correctly, you will get a score of 100%.
* If your work generates **almost all output** correctly (**only one** bad output), you will get a score of 80%
* If your work generates **some output** correctly (some right output, some wrong output), you will get a score of 60%
* If your work generates **only one output** correctly, you will get a score of 40%
* If your work does not generate any correct output, you will get a score of **zero**.