**Regular Expression (RegEx)**

**Regular expressions**

Regular expressions are patterns used to match character combinations in strings. In JavaScript, regular expressions are also objects. These patterns are used with the exec() and test() methods of RegExp, and with the match(), matchAll(), replace(), replaceAll(), search(), and split() methods of String. This chapter describes JavaScript regular expressions.

**Creating a regular expression**

You construct a regular expression in one of two ways:

Using a regular expression literal, which consists of a pattern enclosed between slashes, as follows:

**const re = /ab+c/;**

Regular expression literals provide compilation of the regular expression when the script is loaded. If the regular expression remains constant, using this can improve performance.

Or calling the constructor function of the RegExp object, as follows:

**const re = new RegExp('ab+c');**

Using the constructor function provides runtime compilation of the regular expression. Use the constructor function when you know the regular expression pattern will be changing, or you don't know the pattern and are getting it from another source, such as user input.

**Example:   
/** **\W /g 🡪 Regular Expression to replace Alpha Numerical Character  
/\_/g 🡪 Regular Expression to replace Under Score Character.**

**#LeetCode Problem:-**

A phrase is a palindrome if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.

Given a string s, return true if it is a palindrome, or false otherwise.

Example 1:  
Input: s = "A man, a plan, a canal: Panama"

Output: true  
Explanation: "amanaplanacanalpanama" is a palindrome.

Example 2:  
Input: s = "race a car"

Output: false  
Explanation: "raceacar" is not a palindrome.

Example 3:  
Input: s = " "

Output: true  
Explanation: s is an empty string "" after removing non-alphanumeric characters.

Since an empty string reads the same forward and backward, it is a palindrome.  
  
**Solution:**var isPalindrome = function(s) {

***//Regex Expression: Using | (Or Operator)*** ***we can use multiple regular expression***   
 let str = s.replace(/\W|\_/g, '').toLowerCase();

let i = 0;

let j = str.length-1;

while(i<=j){

if(str[i]===str[j]){

i++;

j--;

}else

return false;

}

return true;

};

isPalindrome(“A man, a plan, a canal: Panama");

**Output:**true