**REGEX ASSIGNMENT**

**1). The time has a format: hours:minutes. Both hours and minutes have two digits, like 09:00.**

**Make a regex to find time in the string: Lunch at 10:10 in the room 123:456. In this task there’s no need to check time correctness yet, so 25:99 can also be a valid result. The regex should not match 333:333.**

function checkHourMin(str) {

    var reg = /^\d{2}:\d{2}/

    if(str.match(reg)){console.log(true)}

    else{console.log(false)}

}

checkHourMin("25:43");

checkHourMin("333:333")

**Output:**

**PS C:\Users\ADMIN\Documents\c\JS\_CODE> node tryyy.js**

**true**

**false**

**=======================================================================**

**2.) Create a function that finds the word "happiness" in the given string (not case sensitive). If found, return "Hurray!", otherwise return "There is no happiness.".**

**Example**

**findHappiness(“Work makes me happy”) -> There is no happiness.**

**findHappiness(“You give me the feeling of happiness”) -> Hurray**

function findHappiness(msg) {

let reg = /happiness/g;

    if (reg.test(msg)) {

        return "Hurray";

    }

    else{

        return "There is no happiness";

    }

}

console.log(findHappiness("hello"));

console.log(findHappiness("hel happiness"));

**Output:**PS C:\Users\ADMIN\Documents\c\JS\_CODE> node assign\_16.js

There is no happiness

Hurray

**=======================================================================**

**3). Write a regular expression that matches only a prime number. Numbers will be presented as strings.**

**Example**

**“7” ➞ true**

**“134” ➞ false**

function Prime(n) {

    var reg = /^1?$|^(11+?)\1+$/;

    return !('1'.repeat(n).match(reg));

}

console.log(Prime("7"));

console.log(Prime("134"));

**Output:**

**PS C:\Users\ADMIN\Documents\c\JS\_CODE> node derivative.js**

**true**

**false**

**=======================================================================**

**4). Create a function that will return an integer number corresponding to the amount of digits in the given integer num**

**Examples**

**num\_of\_digits(1000) ➞ 4**

**num\_of\_digits(12) ➞ 2**

**num\_of\_digits(1305981031) ➞ 10**

function countOfDigits(num) {

    let str = num.toString();

    let reg = /[0-9]/g;

    let count = str.match(reg).length;

    return count;

}

console.log(countOfDigits(7452365));

**OUTPUT:**

PS C:\Users\ADMIN\Documents\c\JS\_CODE> node assign\_16.js

7

**=======================================================================**

**5). Create a function that takes in a number as a string n and returns the number without trailing and leading zeros.**

**● Trailing Zeros are the zeros after a decimal point which don't affect the value (e.g. the last three zeros in 3.4000 and 3.04000).**

**● Leading Zeros are the zeros before a whole number which don't affect the value (e.g. the first three zeros in 000234 and**

**000230).**

**removeLeadingTrailing("230.000") ➞ "230"**

**removeLeadingTrailing("00402") ➞ "402"**

**Notes**

**● Return a string.**

**If you get a number with .0 on the end, return the integer value**

**(e.g. return "4" rather than "4.0").**

**● If the number is 0, 0.0, 000, 00.00, etc... return "0".**

function removeLeadingTrailing(str) {

    let decreg = /((?<=(\.|,)\d\*?[1-9]0+$)|(\.|,)0+$)/g;

    let sr1 =  str.replace(decreg, "");

    let leading0reg = /^0+(?!$)/g || decreg;

    let sr2 = sr1.replace(leading0reg, "")

    return sr2;

}

console.log(removeLeadingTrailing("00402"))

console.log(removeLeadingTrailing("230.000"))

**OUTPUT:**

**PS C:\Users\ADMIN\Documents\c\JS\_CODE> node tryyy.js**

**402**

**230**

**=======================================================================**

**6). Create a function that takes a word and returns true if the word has two consecutive identical letters.**

**Examples**

**doubleLetters("loop") ➞ true**

**doubleLetters("yummy") ➞ true**

function doubleLetters(str) {

    var reg = /([a-z])\1+/g

    if(str.match(reg)){

        return true;

    }

    else{

        return false;

    }

}

console.log(doubleLetters("loop"));

console.log(doubleLetters("yummy"));

**Output:**

**PS C:\Users\ADMIN\Documents\c\JS\_CODE> node abc.js**

**true**

**true**

**=======================================================================**

**7). ATM machines allow 4 or 6 digit PIN codes and PIN codes cannot contain anything but exactly 4 digits or exactly 6 digits. Your task is to create a function that takes a string and returns true if the**

**PIN is valid and false if it's not.**

**Examples**

**validatePIN("1234") ➞ true**

**validatePIN("12345") ➞ false**

function validatePIN(str) {

    var reg1 = /^\d{4}$/;

    var reg2 = /^\d{6}$/;

    if(str.match(reg1) || str.match(reg2)){

        return true;

    }

    else {

        return false;

}

}

console.log(validatePIN("4362"))

console.log(validatePIN("436"))

console.log(validatePIN("431236"))

**=======================================================================**

**8). Create a function that determines whether a string is a valid hex code. A hex code must begin with a pound key # and is exactly 6 characters in length. Each character must be a digit from 0-9 or an alphabetic character from A-F. All alphabetic characters may be uppercase or lowercase.**

**Examples**

**isValidHexCode("#CD5C5C") ➞ true**

**isValidHexCode("#EAECEE") ➞ true**

**isValidHexCode("#CD5C&C") ➞ false**

function isValideHexCode(str) {

    var reg = /^#([A-F]|[a-f]|[0-9]){6}/;

    if(str.match(reg)){

        console.log("true");

    }

    else{

        console.log("false");

    }

}

isValideHexCode("#CD5C5C")

isValideHexCode("#EAECEE")

isValideHexCode("#CD5C&C")

**Output:**

**PS C:\Users\ADMIN\Documents\c\JS\_CODE> node derivative.js**

**true**

**true**

**false**

**=======================================================================**

**9). Create a function that takes an array of numbers and returns "Boom!" if the digit 7 appears in the array. Otherwise, return "there is no 7 in the array".**

**Examples**

**sevenBoom([1, 2, 3, 4, 5, 6, 7]) ➞ "Boom!"**

**// 7 contains the number seven.**

**sevenBoom([8, 6, 33, 100]) ➞ "there is no 7 in the array"**

**// None of the items contain 7 within them.**

function sevenBoom(arr) {

    if(arr.includes(7)) {

        return "Boom!";

    }

    else {

        return "there is no 7 in the array";

    }

}

console.log(sevenBoom([1,3,7,4]));

console.log(sevenBoom([1,3,17,4]));

**Output:**

**PS C:\Users\ADMIN\Documents\c\JS\_CODE> node abc.js**

**Boom!**

**there is no 7 in the array**

**=======================================================================**

**10). Create a function that takes a string, checks if it has the same number of x's and o's and returns either true or false.**

**● Return a boolean value (true or false).**

**● Return true if the amount of x's and o's are the same.**

**● Return false if they aren't the same amount.**

**● The string can contain any character.**

**● When "x" and "o" are not in the string, return true.**

**Examples**

**XO("ooxx") ➞ true**

**XO("xooxx") ➞ false**

**XO("ooxXm") ➞ true**

**// Case insensitive.**

**Notes**

**● Remember to return true if there aren't any x's or o's.**

**● Must be case insensitive.**

function XO(str) {

    var a = str.match(/x/g).length;

    var b = str.match(/o/g).length;

    if(!str.match(/xo/g)){

        return true;}

    else {

    if(a === b){

        return true;}

    else if(a!= b){

        return false;}

    }

}

console.log(XO("xooxx"))

console.log(XO("xxoo"))

console.log(XO("ooxXm"))

**Output:**

**PS C:\Users\ADMIN\Documents\c\JS\_CODE> node tryyy.js**

**false**

**true**

**true**