**Palindrome Checker**

Мое решение

function palindrome(str) {

// Good luck!

const strArr1 = str.match(/[a-z]|\d/gi).map(element => element.toLowerCase());

const strArr2 = strArr1.slice().reverse();

return strArr1.join('') === strArr2.join('');

}

Solution

function palindrome(str) {

return str.replace(/[\W\_]/g, '').toLowerCase() ===

str.replace(/[\W\_]/g,'').toLowerCase().split('').reverse().join('');

}

**Roman Numeral Converter**

Мое решение

const convertToRoman = (num) => {

const sym = [[null, 'M'], ['D', 'C'], ['L', 'X'], ['V', 'I']];

return String(num).split('').map((elem, i) => {

let n = Number(elem);

let j = i + sym.length - String(num).length;

if (n < 4) {

return sym[j][1].repeat(n);

}

if (n === 4) {

return sym[j][1] + sym[j][0];

}

if (n === 9) {

return sym[j][1] + sym[j - 1][1];

}

if (n > 5) {

return sym[j][0] + sym[j][1].repeat(n - 5);

}

if (n === 5) {

return sym[j][0];

}

}).join('');

}

**Caesars Cipher**

Мое решение

function rot13(str) { // LBH QVQ VG!

const repl = sym => {

let num = sym.charCodeAt(0);

return String.fromCharCode((num + 13) > 90 ? (num - 13) : (num + 13));

};

return str.split('').map(elem => {

if (/[A-Z]/.test(elem)) {

return repl(elem);

}

return elem;

}).join('');

}

Solution

function rot13(str) { // LBH QVQ VG!

return str.replace(/[A-Z]/g, L => String.fromCharCode((L.charCodeAt(0) % 26) + 65));

}

**Telephone Number Validator**

Мое решение

function telephoneCheck(str) {

// Good luck!

return /(^1? ?\(\d{3}\)( ?\-?)\d{3}( ?\-?)\d{4})$|^1? ?\d{3} ?\-?\d{3} ?\-?\d{4}$/g.test(str);

}

Solution

function telephoneCheck(str) {

var regex = /^(1\s?)?(\(\d{3}\)|\d{3})[\s\-]?\d{3}[\s\-]?\d{4}$/;

return regex.test(str);

}

**Cash Register**

Мое решение

const checkCashRegister = (price, cash, cid) => {

const amount = [0.01, 0.05, 0.1, 0.25, 1, 5, 10, 20, 100];

let change1 = cash - price;

const till = cid.map(el => el[1]);

if (till.reduce((acc, elem) => acc + elem, 0) === change1) {

return {'status': "CLOSED", 'change': cid};

}

for (let i = cid.length - 1; i >= 0; i--) {

while (change1 >= amount[i] && till[i] >= amount[i]) {

change1 = Math.round((change1 - amount[i]) \* 100) / 100;

till[i] = Math.round((till[i] - amount[i]) \* 100) / 100;

}

}

if (change1 > 0) {

return {'status': "INSUFFICIENT\_FUNDS", 'change': []};

}

return {'status': "OPEN", 'change': till.map((elem, i) => [cid[i][0], Math.round((cid[i][1] - elem) \* 100) / 100]).filter(el => el[1] > 0).reverse()};

};