# Solutions-Exercise: Strings and Regular Expressions

## 1. Binary Decoding

function solve() {

let inputString = document.getElementById('str').value;

let onesSum = getOnesSum(inputString);

let slicedInputString = inputString.slice(onesSum, inputString.length - onesSum);

let outputArray =[];

for (let i = 0; i < slicedInputString.length; i += 8) {

let currentBinaryPart = slicedInputString.slice(i, i + 8);

let currentDecimalPart = parseInt(currentBinaryPart, 2);

let currentCharacter = String.fromCharCode(currentDecimalPart);

let pattern = /[A-Za-z ]/;

if (pattern.test(currentCharacter)){

outputArray.push(currentCharacter);

}

}

let resultElement = document.getElementById('result');

resultElement.innerHTML = outputArray.join('');

function getOnesSum(input){

let sumDigit = input;

while(sumDigit.length > 1){

let currentSumDigit = sumDigit.split('').reduce((a,b) => Number(a) + Number(b)).toString();

sumDigit = currentSumDigit;

}

return Number(sumDigit);

}

}

|  |
| --- |
| function solve() { |
|  | let inputElement = document.getElementById('str'); |
|  | let outputElement = document.getElementById('result'); |
|  |  |
|  | let onesSum = findOnesSum(inputElement.value); |
|  | let endRange = inputElement.value.length - onesSum; |
|  |  |
|  | let slicedValue = inputElement.value.slice(onesSum, endRange); |
|  | let result = []; |
|  |  |
|  | for(let i = 0; i < slicedValue.length; i+=8){ |
|  | let end = i + 8; |
|  | let currentEightElements = slicedValue.slice(i , end); |
|  | let decimalValue = parseInt(currentEightElements, 2); |
|  | let resultChar = String.fromCharCode(decimalValue); |
|  | let pattern = /[A-Za-z\s]/; |
|  | if(pattern.test(resultChar)){ |
|  | result.push(resultChar); |
|  | } |
|  | } |
|  |  |
|  | outputElement.textContent = result.join(""); |
|  | inputElement.value = ""; |
|  |  |
|  | function findOnesSum(inputValue) { |
|  | let result = inputValue; |
|  | while (result.length > 1) { |
|  | let temp = result |
|  | .split("") |
|  | .map(Number) |
|  | .filter((x) => x > 0) |
|  | .reduce((a, b) => a + b) |
|  | .toString(); |
|  |  |
|  | result = temp; |
|  | } |
|  | return result; |
|  | } |
|  | } |

|  |
| --- |
| function solve() { |
|  | let inputElement = document.getElementById('str'); |
|  | let outputElement = document.getElementById('result'); |
|  |  |
|  | let onesSum = findsOnesSum(inputElement.value); |
|  | let end = inputElement.value.length - onesSum; |
|  |  |
|  | let result = inputElement.value.slice(onesSum, end); |
|  |  |
|  | let parts = result |
|  | .split(/([\d]{8})/) |
|  | .filter((x) => x) |
|  | .map((x) => binaryToString(x)) |
|  | .filter((c) => /[A-Za-z ]+/g.test(c)) |
|  | .join(""); |
|  |  |
|  | outputElement.textContent = parts; |
|  |  |
|  |  |
|  | function findsOnesSum(value) { |
|  |  |
|  | let result = value; |
|  |  |
|  | while(result.length > 1){ |
|  | let temp = result |
|  | .split('') |
|  | .reduce((a,b)=> +a + +b) |
|  | .toString(); |
|  |  |
|  | result = temp; |
|  | } |
|  | return +result; |
|  | } |
|  |  |
|  | function binaryToString(element) { |
|  | let decimal = parseInt(element, 2); |
|  | return String.fromCharCode(decimal); |
|  | } |
|  | } |

## 2. Expedition problems

function solve() {

let keyword = document.getElementById('str').value;

let inputText = document.getElementById('text').value;

let messageRegex = new RegExp(`${keyword}(.\*?)${keyword}`);

let messageText = messageRegex.exec(inputText)[1];

let message = `Message: ${messageText}`;

let coordinatesPattern = /(east|north).\*?(\d{2})[^,]\*?,[^,]\*?(\d{6})/gi;

//let coordinatesPattern = /(east|north)[\s\S]\*?(\d{2})[^,]\*?,[^,]\*?(\d{6})/gi;

let northCoordinate = '';

let eastCoordinate = '';

let match;

while((match = coordinatesPattern.exec(inputText)) !== null){

if(match[1].toLowerCase() === 'north'){

northCoordinate = `${match[2]}.${match[3]} N`;

} else if(match[1].toLowerCase() === 'east'){

eastCoordinate = `${match[2]}.${match[3]} E`;

}

}

appendToResultElement(northCoordinate);

appendToResultElement(eastCoordinate);

appendToResultElement(message);

function appendToResultElement(text){

let resultElement = document.getElementById('result');

let p = document.createElement('p');

p.textContent = text;

resultElement.appendChild(p);

}

}

|  |
| --- |
| function solve() { |
|  | let inputStrElement = document.getElementById('str'); |
|  | let inputTextElement = document.getElementById('text'); |
|  | let outputElement = document.getElementById('result'); |
|  |  |
|  | let pattern = /(north|east)(?:[\s\S])\*?([\d]{2})(?:[^,]+)\*?,(?:[^,])\*?([\d]{6})/gmi; |
|  | let matches = inputTextElement.value.match(pattern); |
|  | let matchedInfo = []; |
|  |  |
|  | if (matches.length > 0) { |
|  | matches.forEach((match) => { |
|  | pattern = /(north|east)(?:[\s\S])\*?([\d]{2})(?:[^,]+)\*?,(?:[^,])\*?([\d]{6})/gmi; |
|  | let result = pattern.exec(match); |
|  |  |
|  | if (result.length > 1) { |
|  | matchedInfo.push(result.slice(1, 4).join(" ")); |
|  | } |
|  | }); |
|  | } else { |
|  | return; |
|  | } |
|  |  |
|  | let northCoords = takeRightCoords('NORTH'); |
|  | let eastCoords = takeRightCoords('EAST'); |
|  |  |
|  | let message = inputTextElement.value.split(inputStrElement.value)[1]; |
|  |  |
|  | appendToTheFather(`${northCoords[1]}.${northCoords[2]} ${northCoords[0][0]}`); |
|  | appendToTheFather(`${eastCoords[1]}.${eastCoords[2]} ${eastCoords[0][0]}`); |
|  | appendToTheFather(`Message: ${message}`); |
|  |  |
|  | function takeRightCoords(rightWay) { |
|  |  |
|  | console.log(matchedInfo); |
|  | return matchedInfo |
|  | .reverse() |
|  | .map((coords) => coords.toUpperCase()) |
|  | .filter((coordsInfo) => coordsInfo |
|  | .includes(rightWay))[0].split(' '); |
|  | } |
|  |  |
|  | function appendToTheFather(text){ |
|  | let p = document.createElement('p'); |
|  | p.textContent = text; |
|  | outputElement.appendChild(p); |
|  | } |
|  |  |
|  | inputStrElement.value = ""; |
|  | inputTextElement.value = ""; |
|  | } |

|  |
| --- |
| function solve() { |
|  | let keyword = document.getElementById('str').value; |
|  | let text = document.getElementById('text').value; |
|  |  |
|  | let outputElement = document.getElementById('result'); |
|  |  |
|  | let regEx = new RegExp(`${keyword}(.\*?)${keyword}`, "g"); |
|  | let message = `Message: ${regEx.exec(text)[1]}`; |
|  | text = text.replace(message, ""); |
|  |  |
|  | let regex = /(east|north)[\s\S]\*?([\d]{2})[^,]\*?,[^,]\*?([\d]{6})/gi; |
|  |  |
|  | let m; |
|  |  |
|  | let east = ""; |
|  | let north = ""; |
|  |  |
|  | while ((m = regex.exec(text)) !== null) { |
|  | if(m[1].toUpperCase() === 'NORTH'){ |
|  | north = `${m[2]}.${m[3]} N`; |
|  | } else if(m[1].toUpperCase() === "EAST"){ |
|  | east = `${m[2]}.${[m[3]]} E`; |
|  | } |
|  | } |
|  |  |
|  | appendToParent(north); |
|  | appendToParent(east); |
|  | appendToParent(message); |
|  |  |
|  | function appendToParent(text) { |
|  | let p = document.createElement('p'); |
|  | p.textContent = text; |
|  | outputElement.appendChild(p); |
|  | } |
|  | } |

## 3. James Bond

function solve() {

let input = document.getElementById('arr').value;

let inputArray = JSON.parse(input);

let resultElement = document.getElementById('result');

let key = inputArray.shift();

let keyRegex = new RegExp(`((?<=(\\s)${key})|(?<=^${key}))\\s+([A-Z!%$#]{8,})([\\s.,]|$)`, 'gi');

let resultRow;

let resultArray = [];

let caseChecker = /[a-z]+/g;

for (let i = 0; i < inputArray.length; i++) {

resultArray[i] = inputArray[i];

while(resultRow = keyRegex.exec(inputArray[i])){

if(!caseChecker.exec(resultRow[3])){

inputArray[i] = inputArray[i].replace(resultRow[3], decode(resultRow[3]));

resultArray[i] = inputArray[i];

}

}

}

for(let row of resultArray){

let p = document.createElement('p');

p.textContent = row;

resultElement.appendChild(p);

}

function decode(encoded){

encoded = encoded.toLowerCase();

if(encoded.includes('!')){

encoded = encoded.replace('!', '1');

}

if(encoded.includes('%')){

encoded = encoded.replace('%', '2');

}

if(encoded.includes('#')){

encoded = encoded.replace('#', '3');

}

if(encoded.includes('$')){

encoded = encoded.replace('$', '4');

}

return encoded;

}

}

|  |
| --- |
| function solve() { |
|  |  |
|  | let input = document.querySelector('#arr').value; |
|  | input = JSON.parse(input); |
|  | let output = document.querySelector('#result'); |
|  | let key = input.shift(); |
|  |  |
|  | let keyFinder = new RegExp(`((?<=(\\s)${key})|(?<=^${key}))\\s+([A-Z!%$#]{8,})([\\s.,]|$)`, 'gi'); |
|  | let caseChecher = /[a-z]+/g; |
|  |  |
|  | let result; |
|  | let resultArr = []; |
|  |  |
|  | for (let i = 0; i < input.length; i++) { |
|  | resultArr[i] = input[i]; |
|  | while (result = keyFinder.exec(input[i])) { |
|  | if (!caseChecher.exec(result[3])) { |
|  | input[i] = input[i].replace(result[3], decode(result[3])); |
|  | resultArr[i] = input[i]; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | for (let row of resultArr) { |
|  | createAddAppend('p', row, output); |
|  | } |
|  |  |
|  | function createAddAppend(child, text, parent) { |
|  | let childElement = document.createElement(child); |
|  | childElement.textContent = text; |
|  | parent.appendChild(childElement); |
|  | } |
|  | function decode(text) { |
|  | text = text.toLowerCase(); |
|  | for (let char of text) { |
|  | if (text.includes('!')) { |
|  | text = text.replace('!', '1'); |
|  | } |
|  | if (text.includes('%')) { |
|  | text = text.replace('%', '2'); |
|  | } |
|  | if (text.includes('#')) { |
|  | text = text.replace('#', '3'); |
|  | } |
|  | if (text.includes('$')) { |
|  | text = text.replace('$', '4'); |
|  | } |
|  | } |
|  | return text; |
|  | } |
|  | } |

## 4. Airport Check

function solve() {

let inputString = document.getElementById('str').value;

let [aiportData, neededInformation] = inputString.split(',');

neededInformation = neededInformation.trim();

let namePattern = / [A-Z][a-zA-Z]\*(-[A-Z][a-zA-Z]\*\.)?-[A-Z][a-zA-Z]\* /g;

let airportPattern = / ([A-Z]{3})\/([A-Z]{3}) /g;

let flightPattern = / [A-Z]{1,3}\d{1,5} /g;

let companyPattern = /- [A-Z][A-Za-z]\*\\*[A-Z][A-Za-z]\* /g;

// let name = namePattern.exec(aiportData)[0].trim().replace('-', ' ');

let name = namePattern.exec(aiportData)[0].trim().replace(/-/g, ' ');

let flight = flightPattern.exec(aiportData)[0].trim();

let company = companyPattern.exec(aiportData)[0].replace('-', '').trim().replace('\*', ' ');

let airport = airportPattern.exec(aiportData);

let resultElement = document.getElementById('result');

switch (neededInformation) {

case 'name': resultElement.textContent = `Mr/Ms, ${name}, have a nice flight!`; break;

case 'flight': resultElement.textContent = `Your flight number ${flight} is from ${airport[1]} to ${airport[2]}.`; break;

case 'company': resultElement.textContent = `Have a nice flight with ${company}.`; break;

case 'all': resultElement.textContent = `Mr/Ms, ${name}, your flight number ${flight} is from ${airport[1]} to ${airport[2]}. Have a nice flight with ${company}.`; break;

}

}

Beginning:

|  |
| --- |
| function solve() { |
|  |  |
|  | let string = document.getElementById('str'); |
|  | let outputElement = document.getElementById('result'); |
|  |  |
|  | let [text, wantedInfo] = string.value.split(', '); |
|  | let message; |
|  |  |
|  | let namePattern = / ([A-Z]+([A-Za-z]\*)?)(-[A-Z][A-Za-z]\***\.**)?-([A-Z][A-Za-z]\*)? /g; |
|  | let airportPattern = / [A-Z]{3}**\/**[A-Z]{3} /g; |
|  | let flightNumberPattern = / [A-Z]{1,3}[\d]{1,5} /g; |
|  |  |
|  |  |
|  | if(wantedInfo === 'name'){ |
|  | let matchInfo = text.match(namePattern)[0]; |
|  | let name = matchInfo.trim().replace('-', ' '); |
|  | outputElement.textContent = `Mr/Ms, ${name}, have a nice flight!`; |
|  | } else if(wantedInfo === "flight"){ |
|  | let airport = text.match(airportPattern)[0].split('/').map((e) => e.trim()); |
|  | let flightNumber = text.match(flightNumberPattern)[0].trim(); |
|  |  |
|  | outputElement.textContent = `Your flight number ${flightNumber} is from ${airport[0]} to ${airport[1]}.`; |
|  |  |
|  |  |
|  | //TODO...? :) |
|  | } |
|  | } |

|  |
| --- |
| function solve() { |
|  | let string = document.getElementById('str').value; |
|  | let result = document.getElementById('result'); |
|  |  |
|  | let [text, wantedInfo] = string.split(', '); |
|  | let namePattern = / ([A-Z]+([A-Za-z]\*)?)(-[A-Z][A-Za-z]\***\.**)?-([A-Z][A-Za-z]\*)? /g; |
|  | let airportPattern = / [A-Z]{3}**\/**[A-Z]{3} /g; |
|  | let flightNumberPattern = / [A-Z]{1,3}[\d]{1,5} /g; |
|  | let companyPattern = /- [A-Za-z]+**\\***[A-Za-z]+ /g; |
|  |  |
|  | let nameInfo = text.match(namePattern); |
|  | let airportInfo = text.match(airportPattern); |
|  | let flightInfo = text.match(flightNumberPattern); |
|  | let companyInfo = text.match(companyPattern); |
|  |  |
|  | let name = nameInfo[0].trim().replace(/-/g, ' '  ); |
|  | let flight = flightInfo[0].trim(); |
|  | let [from, to] = airportInfo[0].trim().split('/'); |
|  | let company = companyInfo[0].replace('-', '').trim().replace('\*', ' '); |
|  | switch(wantedInfo){ |
|  | case 'name': |
|  | result.textContent = `Mr/Ms, ${name}, have a nice flight!`; break; |
|  | case 'flight': |
|  | result.textContent = `Your flight number ${flight} is from ${from} to ${to}.`; break; |
|  | case 'company': |
|  | result.textContent = `Have a nice flight with ${company}.`; break; |
|  | case 'all': |
|  | result.textContent = `Mr/Ms, ${name}, your flight number ${flight} is from ${from} to ${to}. Have a nice flight with ${company}.`; break; |
|  | } |
|  |  |
|  | } |

100/100

function solve() {

// let str = 'ahah Second-Testov )\*))&&ba SOF/VAR ela\*\* FB973 - Bulgaria\*Air -opFB900 pa-SOF/VAr//\_- T12G12 STD08:45 STA09:35 , all';

let str = document.getElementById('str').value;

let divRes = document.getElementById('result');

let chunks = str.split(',');

let input = chunks[0];

let type = chunks[1].trim();

let namePattern =/ ([A-Z]+([A-Za-z]\*)?)(-[A-Z][A-Za-z]\*\.)?-([A-Z][A-Za-z]\*)? /g;

let airportPattern = /( ([A-Z]{3})\/([A-Z]{3}) )/gm;

let flightPattern = /( [A-Z]{1,3}[0-9]{1,5} )/gm;

let companyPattern = /(- [A-Z]([A-Za-z]+)\*\\*[A-Z]([A-Za-z]\*) )/gm;

let name = namePattern.exec(input)[0].trim().replace(/-/g,' ');

let flight = flightPattern.exec(input)[0].trim();

let company = companyPattern.exec(input)[0].replace(/[ \-]/g,'').replace(/[\*]/g,' ');

let airport = airportPattern.exec(input);

switch (type) {

case 'name':

divRes.textContent = (`Mr/Ms, ${name}, have a nice flight!`);

break;

case 'flight':

divRes.textContent = (`Your flight number ${flight} is from ${airport[2]} to ${airport[3]}.`);

break;

case 'company':

divRes.textContent=(`Have a nice flight with ${company}.`);

break;

case 'all':

divRes.textContent = (`Mr/Ms, ${name}, your flight number ${flight} is from ${airport[2]} to ${airport[3]}. Have a nice flight with ${company}.`);

break;

}

}

100/100

|  |
| --- |
| function solve() { |
|  |  |
|  | let input = document.querySelector('#str').value; |
|  | let output = document.querySelector('#result'); |
|  |  |
|  | input = input.split(', '); |
|  | let command = input.pop(); |
|  | input = input.toString(); |
|  |  |
|  | let nameValidator = /(?<=\s)([A-Z][A-Za-z]\*)-([A-Z][A-Za-z]\*)(?=\s)|((?<=\s)([A-Z][A-Za-z]\*)-([A-Z][A-Za-z]\*)**\.**-([A-Z][A-Za-z]\*)(?=\s))/g; |
|  | let airportValidator = /(?<=\s)([A-Z]{3})**\/**([A-Z]{3})(?=\s)/g; |
|  | let flightValidator = /(?<=\s)([A-Z]{1,3})(\d{1,5})(?=\s)/g; |
|  | let companyValidator = /(?<=-\s)([A-Z][A-Za-z]\*)**\\***([A-Z][A-Za-z]\*)(?=\s)/g; |
|  |  |
|  | let result; |
|  |  |
|  | if (command === 'name') { |
|  |  |
|  | let passenger = validator(nameValidator); |
|  | output.textContent = `Mr/Ms, ${passenger}, have a nice flight!`; |
|  |  |
|  | } else if (command === 'flight') { |
|  |  |
|  | let flightNum = validator(flightValidator); |
|  | let airports = validator(airportValidator); |
|  | output.textContent = `Your flight number ${flightNum} is from ${airports[0]} to ${airports[1]}.`; |
|  |  |
|  | } else if (command === 'company') { |
|  |  |
|  | let company = validator(companyValidator); |
|  | output.textContent = `Have a nice flight with ${company}.`; |
|  |  |
|  | } else if (command === 'all') { |
|  |  |
|  | let passenger = validator(nameValidator); |
|  | let airports = validator(airportValidator); |
|  | let flightNum = validator(flightValidator); |
|  | let company = validator(companyValidator); |
|  | output.textContent = `Mr/Ms, ${passenger}, your flight number ${flightNum} is from ${airports[0]} to ${airports[1]}. Have a nice flight with ${company}.`; |
|  | } |
|  |  |
|  | function validator(regEx) { |
|  | result = regEx.exec(input); |
|  | if (regEx === nameValidator) { |
|  | while (result[0].includes('-')) { |
|  | result[0] = result[0].replace('-', ' '); |
|  | } |
|  | } else if (regEx === airportValidator) { |
|  | result[0] = result[0].split('/'); |
|  | } else if (regEx === companyValidator) { |
|  | result[0] = result[0].replace('\*', ' '); |
|  | } |
|  | return result[0]; |
|  | } |
|  | } |