Solutions – Exam Preparation for exam 10.II.2019

# Solutons-JS Fundamentals: Test Exam

## 1.Problem Stadium

function solve() {

let teamTicketPrices = {'A':10, 'B':7, 'C':5};

let vipTicketPrices = {'A':25, 'B':15, 'C':10};

let summary = {fans:0, profit:0};

let buttons = document.getElementsByClassName('seat');

let outputTextarea = document.getElementById('output');

let summaryElement = document.getElementById('summary');

Array.from(buttons).forEach(b => {

b.addEventListener('click', clickButton);

});

let summaryButton = summaryElement.children[0];

summaryButton.addEventListener('click', showSummary);

function clickButton(event){

let button = event.target;

let seatNumber = Number(button.textContent);

let sector = String.fromCharCode(Number(button.parentNode.cellIndex) + 65);

let zone = button.parentNode.parentNode.parentNode.parentNode.parentNode.className;

if(button.style.backgroundColor === ''){

summary.fans++;

if(zone !== 'VIP'){

summary.profit += teamTicketPrices[sector];

} else {

summary.profit += vipTicketPrices[sector];

}

button.style.backgroundColor = 'rgb(255, 0, 0)';

outputTextarea.value += ` Seat ${seatNumber} in zone ${zone} sector ${sector} was taken.\n`;

} else {

outputTextarea.value += ` Seat ${seatNumber} in zone ${zone} sector ${sector} is unavailable.\n`;

}

}

function showSummary(){

let summarySpan = summaryElement.children[1];

summarySpan.textContent = `${summary.profit} leva, ${summary.fans} fans.`;

}

}

|  |
| --- |
| function solve() { |
|  | let ticketPrices = { |
|  | 'TEAM': { |
|  | 'A': 10, |
|  | 'B': 7, |
|  | 'C': 5 |
|  | }, |
|  | 'VIP': { |
|  | 'A': 25, |
|  | 'B': 15, |
|  | 'C': 10 |
|  | }, |
|  | 'summary': { |
|  | 'fans': 0, |
|  | 'profit': 0 |
|  | } |
|  | }; |
|  |  |
|  | let buttons = document.getElementsByClassName('seat'); |
|  | let textArea = document.getElementById('output'); |
|  | let summary = document.getElementById('summary'); |
|  |  |
|  | Array.from(buttons).forEach((btn) => { |
|  | btn.addEventListener('click', buttonClick); |
|  | }); |
|  |  |
|  | function buttonClick(e){ |
|  | let button = e.target; |
|  | let seatNumber = +button.textContent; |
|  | let zone = button.parentNode.parentNode.parentNode.parentNode.parentNode.className; |
|  | let sector = String.fromCharCode(+(e.target.parentNode.cellIndex) + 65); |
|  |  |
|  | if(button.style.backgroundColor === ''){ |
|  | ticketPrices.summary.fans +=1; |
|  |  |
|  | let key = zone !== "VIP" ? 'TEAM' : 'VIP'; |
|  |  |
|  | ticketPrices.summary.profit += ticketPrices[key][sector]; |
|  | e.target.style.backgroundColor = "rgb(255,0,0)"; |
|  |  |
|  | textArea.value += ` Seat ${seatNumber} in zone ${zone} sector ${sector} was taken.\n`; |
|  | } else { |
|  | textArea.value += ` Seat ${seatNumber} in zone ${zone} sector ${sector} is unavailable.\n`; |
|  | } |
|  | } |
|  |  |
|  | summary.children[0].addEventListener('click', printTheSummary); |
|  |  |
|  | function printTheSummary(){ |
|  | summary.children[1].textContent = `${ticketPrices.summary.profit} leva, ${ticketPrices.summary.fans} fans.`; |
|  | } |
|  | } |

function solve() {

let btns = Array.from(document.getElementsByTagName('button'));

let output = document.getElementsByTagName('textarea')[0];

let priceArr = [[10, 7, 5], [10, 7, 5], [25, 15, 10]];

let zone = ['Levski', 'Litex', 'VIP'];

let typeSeat = ['A', 'B', 'C'];

let stat = [0, 0];

for (let i = 0; i < btns.length - 1; i++) {

btns[i].addEventListener('click', () => {

let priceInd = (i % 3);

let seatNum = Math.floor(i / 3) + 1;

let zoneNum = Math.floor(i / 15);

if (btns[i].style.backgroundColor) {

output.textContent += ` Seat ${seatNum - zoneNum \* 5} in zone ${zone[zoneNum]} sector ${typeSeat[priceInd]} is unavailable.\n`

} else {

btns[i].style.backgroundColor = 'rgb(255,0,0)';

output.textContent += ` Seat ${seatNum - zoneNum \* 5} in zone ${zone[zoneNum]} sector ${typeSeat[priceInd]} was taken.\n`;

stat[0] += 1;

stat[1] += priceArr[zoneNum][priceInd];

}

})

}

document.getElementsByTagName('button')[45].addEventListener('click', () => {

let otputPrint = document.querySelector('#summary span');

otputPrint.textContent = `${stat[1]} leva, ${stat[0]} fans.`

})

}

## 2.Problem Crossword

function solve() {

let inputElement = document.getElementById('input');

let buttons = document.getElementsByTagName('button');

let fifterButton = buttons[0];

let sortButton = buttons[1];

let rotateButton = buttons[2];

let getButton = buttons[3];

//let outputP = document.getElementById('output').firstChild;

let outputP = document.getElementById('output').children[0];

fifterButton.addEventListener('click', filter);

sortButton.addEventListener('click', sort);

rotateButton.addEventListener('click', rotate);

getButton.addEventListener('click', get);

function filter(){

let inputSplited = inputElement.value.split('');

//let inputSplited = document.getElementById('input').value.split('');

let secondCommand = document.getElementById('filterSecondaryCmd').value;

let indexPosition = Number(document.getElementById('filterPosition').value) - 1;

switch(secondCommand){

//case 'UPERCASE':

case 'uppercase':

//outputP.textContent += inputSplited.filter(c => c === c.toUpperCase() && !Number(c))[indexPosition];

outputP.textContent += inputSplited.filter(c => c === c.toUpperCase() && isNaN(c))[indexPosition];

break;

//case 'LOWERCASE':

case 'lowercase':

//outputP.textContent += inputSplited.filter(c => c === c.toLowerCase() && !Number(c))[indexPosition];

outputP.textContent += inputSplited.filter(c => c === c.toLowerCase() && isNaN(c))[indexPosition];

break;

//case 'NUMS':

case 'nums':

//outputP.textContent += inputSplited.filter(c => Number(c))[indexPosition];

outputP.textContent += inputSplited.filter(c => !isNaN(c))[indexPosition];

break;

}

}

function sort(){

//let inputSplited = inputElement.value.split('').sort();

//let inputSplited = inputElement.value.split('').sort((a, b) => a.localeCompare(b));

let inputSplited = inputElement.value.split('');

let secondCommand = document.getElementById('sortSecondaryCmd').value;

let indexPosition = Number(document.getElementById('sortPosition').value) - 1;

// if (secondCommand === 'Z'){

// inputSplited = inputSplited.reverse();

// }

if (secondCommand === 'A'){

inputSplited.sort((a, b) => a.localeCompare(b));

} else {

inputSplited.sort((a, b) => b.localeCompare(a));

}

outputP.textContent += inputSplited[indexPosition];

}

function rotate(){

let inputSplited = inputElement.value.split('');

let secondCommand = document.getElementById('rotateSecondaryCmd').value;

let indexPosition = Number(document.getElementById('rotatePosition').value) - 1;

for (let i = 0; i < secondCommand % inputSplited.length; i++) {

inputSplited.unshift(inputSplited.pop());

}

outputP.textContent += inputSplited[indexPosition];

}

function get(){

//let input = inputElement.value.split('');

let input = inputElement.value;

let indexPosition = Number(document.getElementById('getPosition').value) - 1;

outputP.textContent += input[indexPosition];

}

}

|  |
| --- |
| function solve() { |
|  | let buttons = document.getElementsByTagName('button'); |
|  | let output = document.querySelector('#output p'); |
|  |  |
|  | buttons[0].addEventListener('click', filter); |
|  | buttons[1].addEventListener('click', sort); |
|  | buttons[2].addEventListener('click', rotate); |
|  | buttons[3].addEventListener('click', get); |
|  |  |
|  | function filter() { |
|  | let input = document.getElementById('input').value.split(''); |
|  | let secondCmd = document.getElementById('filterSecondaryCmd').value; |
|  | let position = (+document.getElementById('filterPosition').value) - 1; |
|  |  |
|  | switch (secondCmd) { |
|  | case 'uppercase': |
|  | output.textContent += input.filter((c) => c === c.toUpperCase() && isNaN(c))[position]; |
|  | break; |
|  | case 'lowercase': |
|  | output.textContent += input.filter((c) => c === c.toLowerCase() && isNaN(c))[position]; |
|  | break; |
|  | case 'nums': |
|  | output.textContent += input.filter((c) => !isNaN(c))[position]; |
|  | break; |
|  | } |
|  | } |
|  |  |
|  | function sort() { |
|  | let input = document.getElementById('input').value.split('').sort((a,b) => a.localeCompare(b)); |
|  | let secondCmd = document.getElementById('sortSecondaryCmd').value; |
|  | let position = (+document.getElementById('sortPosition').value) - 1; |
|  |  |
|  | if(secondCmd === 'Z'){ |
|  | input = input.reverse(); |
|  | } |
|  | output.textContent += input[position]; |
|  | } |
|  |  |
|  | function rotate() { |
|  | let input = document.getElementById('input').value.split(''); |
|  |  |
|  | let secondCmd = +document.getElementById('rotateSecondaryCmd').value; |
|  | let position = (+document.getElementById('rotatePosition').value) - 1; |
|  |  |
|  | let rotation = secondCmd % input.length; |
|  |  |
|  | while(rotation > 0){ |
|  | let char = input.pop(); |
|  | input.unshift(char); |
|  | rotation -=1; |
|  | } |
|  |  |
|  | output.textContent += input[position]; |
|  | } |
|  |  |
|  | function get() { |
|  | let input = document.getElementById('input').value.split(''); |
|  | let position = (+document.getElementById('getPosition').value) - 1 |
|  | output.textContent += input[position] |
|  | } |
|  | } |

## 3.Problem Extractor

function solve() {

let inputElement = document.getElementById('input');

let outputElement = document.getElementById('output');

let button = document.getElementsByTagName('button')[0];

button.addEventListener('click', extract);

function extract(){

let inputString = inputElement.value;

let charactersToTakeCountString = /[0-9]+/.exec(inputString)[0];

// let stringToTake = inputString.substr(0, Number(charactersToTakeCountString));

// stringToTake = stringToTake.substr(charactersToTakeCountString.length);

let stringToTake = inputString.substring(charactersToTakeCountString.length, Number(charactersToTakeCountString) + charactersToTakeCountString.length);

let delimeter = stringToTake[stringToTake.length - 1];

let [charactersToRemove, stringToCheck] = stringToTake.split(delimeter).filter(c => c !== '');

let regex = new RegExp(`[${charactersToRemove}]`, 'g');

let outputString = stringToCheck.replace(regex, '');

outputString = outputString.replace(/#/g, ' ');

outputElement.value = outputString;

}

}

|  |
| --- |
| function solve() { |
|  | let inputElement = document.getElementById('input'); |
|  | let outputElement = document.getElementById('output'); |
|  | let button = document.querySelector('#exercise button'); |
|  |  |
|  | button.addEventListener('click', doExtraction); |
|  |  |
|  | function doExtraction(){ |
|  |  |
|  | let charToTake = (/[0-9]+/.exec(inputElement.value))[0]; |
|  | let takenSubstring = inputElement.value.slice(charToTake.length, (+charToTake + charToTake.length)); |
|  | let delimeter = takenSubstring.slice(-1); |
|  | let parts = takenSubstring.split(delimeter).filter((x) => x !== ''); |
|  |  |
|  | parts[1] = parts[1].replace(new RegExp(`[${parts[0]}]`, 'g'), ''); |
|  | parts[1] = parts[1].replace(/[#]/g, " "); |
|  |  |
|  | outputElement.value = parts[1]; |
|  | } |
|  | } |

function solve() {

let button = document.querySelector("#exercise > button");

button.addEventListener("click", extract);

function extract() {

let input = document.getElementById("input").value;

let number = /[0-9]+/.exec(input)[0];

let str = input.substring(number.length, +number + number.length);

let symbol = str[str.length - 1];

let arr = str.split(symbol);

let regex = new RegExp(`[${arr[0]}]`);

let dirtyString = arr[1];

while (dirtyString.match(regex)) {

dirtyString = dirtyString.replace(regex, "");

}

while (dirtyString.includes("#")) {

dirtyString = dirtyString.replace("#", " ");

}

document.getElementById("output").textContent = dirtyString;

}

}

## 4.Problem Trucks

function solve() {

let buttons = document.getElementsByTagName('button');

let addNewTruckButton = buttons[0];

let addNewTiresButton = buttons[1];

let goToWorkButton = buttons[2];

let endOfTheShiftButton = buttons[3];

let fieldsets = document.getElementsByTagName('fieldset');

let newTruckFieldset = fieldsets[0];

let newTiresFieldset = fieldsets[1];

let workFieldSet = fieldsets[2];

let backupTiresSetsFieldset = fieldsets[3];

let trucksFieldset = fieldsets[4];

let outputTextarea = document.querySelector('textarea');

//let truckObject = { 'backupTireSets': [] };

let truckObject = { backupTireSets: [] };

addNewTruckButton.addEventListener('click', addNewTruck);

addNewTiresButton.addEventListener('click', addNewTires);

goToWorkButton.addEventListener('click', goToWork);

endOfTheShiftButton.addEventListener('click', showEndOfShiftInfo);

function addNewTruck(){

let plateNumber = document.getElementById('newTruckPlateNumber').value;

let tires = document.getElementById('newTruckTiresCondition').value.split(' ').map(Number);

//if(!truckObject[plateNumber]){

if(!truckObject.hasOwnProperty(plateNumber)){

truckObject[plateNumber] = {tires, distance: 0};

}

let truck = createElement('div', plateNumber, 'truck');

let truckDiv = trucksFieldset.lastElementChild;

truckDiv.appendChild(truck);

}

function addNewTires(){

let tires = document.getElementById('newTiresCondition').value.split(' ').map(Number);

truckObject.backupTireSets.push(tires);

let tireSet = createElement('div', tires.join(' '), 'tireSet');

tireDiv = backupTiresSetsFieldset.lastElementChild;

tireDiv.appendChild(tireSet);

}

function goToWork(){

let plateNumber = document.getElementById('workPlateNumber').value;

let distance = Number(document.getElementById('distance').value);

if(truckObject.hasOwnProperty(plateNumber)){

let results = areTiresGoodEnogh(truckObject[plateNumber].tires, distance);

if(results.finalResult){

truckObject[plateNumber].distance += distance;

truckObject[plateNumber].tires = results.testedTires;

} else if (truckObject.backupTireSets.length > 0){

let backupSet = truckObject.backupTireSets[0];

let results = areTiresGoodEnogh(backupSet, distance);

if(results.finalResult){

truckObject[plateNumber].distance += distance;

truckObject[plateNumber].tires = results.testedTires;

truckObject.backupTireSets.shift();

let usedTires = document.querySelector('div.tireSet');

usedTires.remove();

}

}

}

}

function showEndOfShiftInfo(){

Object.keys(truckObject).filter(plateNumber => plateNumber !== 'backupTireSets').forEach(plate => {

outputTextarea.value += `Truck ${plate} has traveled ${truckObject[plate].distance}.\n`;

});

outputTextarea.value += `You have ${truckObject.backupTireSets.length} sets of tires left.\n`;

}

function areTiresGoodEnogh(tires, disnance){

let parsedDistance = disnance /1000;

let result = {testedTires: [], finalResult: false};

tires.forEach(tire => {

result.testedTires.push(tire - parsedDistance);

});

if(result.testedTires.every(t => t >= 0)){

result.finalResult = true;

}

return result;

}

function createElement(type, text, className){

let element = document.createElement(type);

element.textContent = text;

element.classList.add(className);

return element;

}

}

|  |
| --- |
| function solve() { |
|  |  |
|  | let buttons = document.getElementsByTagName('button'); |
|  | let fieldSets = document.getElementsByTagName('fieldset'); |
|  | let output = document.querySelector('textarea'); |
|  |  |
|  | let obj = { |
|  | 'backupTireSets': [] |
|  | }; |
|  |  |
|  | buttons[0].addEventListener('click', addNewTruck); |
|  | buttons[1].addEventListener('click', addNewTireSet); |
|  | buttons[2].addEventListener('click', goToWork); |
|  | buttons[3].addEventListener('click', endOfShiftInfo); |
|  |  |
|  | function addNewTruck() { |
|  | let plateNumber = document.getElementById('newTruckPlateNumber').value; |
|  | let tires = document.getElementById('newTruckTiresCondition').value.split(' ').map(Number); |
|  |  |
|  | if (!obj[plateNumber]) { |
|  | obj[plateNumber] = { |
|  | tires, |
|  | 'distance': 0 |
|  | } |
|  |  |
|  | let currentTruck = createElement('div', plateNumber, 'truck'); |
|  | let parentTruck = fieldSets[4].lastElementChild; |
|  | parentTruck.appendChild(currentTruck); |
|  | } |
|  | } |
|  |  |
|  | function addNewTireSet() { |
|  | let tires = document.getElementById('newTiresCondition').value.split(' ').map(Number); |
|  | obj.backupTireSets.push(tires); |
|  |  |
|  | let curretTireSet = createElement('div', tires.join(' '), 'tireSet'); |
|  | let parentTruck = fieldSets[3].lastElementChild; |
|  | parentTruck.appendChild(curretTireSet); |
|  | } |
|  |  |
|  | function goToWork() { |
|  | let plateNumber = document.getElementById('workPlateNumber').value; |
|  | let distance = +document.getElementById('distance').value; |
|  |  |
|  | if (obj.hasOwnProperty(plateNumber)) { |
|  |  |
|  | let results = areTiresGoodEnogh(obj[plateNumber].tires, distance); |
|  |  |
|  | if (results.finalResult) { |
|  | obj[plateNumber].distance += distance; |
|  | obj[plateNumber].tires = results.testedTires; |
|  | } else if (obj.backupTireSets.length > 0) { |
|  | let backupSet = obj.backupTireSets[0]; |
|  |  |
|  | let results = areTiresGoodEnogh(backupSet, distance); |
|  |  |
|  | if (results.finalResult) { |
|  | obj[plateNumber].distance += distance; |
|  | obj[plateNumber].tires = results.testedTires; |
|  | obj.backupTireSets.shift(); |
|  | let usedTires = document.querySelector('div.tireSet'); |
|  | usedTires.remove() |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | function endOfShiftInfo(){ |
|  | Object.keys(obj).filter((plateNumber) => plateNumber !== 'backupTireSets').forEach((plate) => { |
|  | output.value += `Truck ${plate} has traveled ${obj[plate].distance}.\n` |
|  | }) |
|  |  |
|  | output.value += `You have ${obj.backupTireSets.length} sets of tires left.\n` |
|  | } |
|  |  |
|  | function areTiresGoodEnogh(tires, distance) { |
|  |  |
|  | let dist = distance / 1000; |
|  |  |
|  | let result = { |
|  | 'testedTires': [], |
|  | 'finalResult': false |
|  | }; |
|  |  |
|  | tires.forEach((tire) => { |
|  | result.testedTires.push(tire - dist); |
|  | }); |
|  |  |
|  | if (result.testedTires.every((e) => e >= 0)) { |
|  | result.finalResult = true; |
|  | } |
|  |  |
|  | return result; |
|  | } |
|  |  |
|  | function createElement(type, text, className) { |
|  | let element = document.createElement(type); |
|  | element.textContent = text; |
|  | element.classList.add(className); |
|  | return element; |
|  | } |
|  | } |