# Homework: JavaScript Loops, Arrays, Strings

This document defines the homework assignments from the [“JavaScript Basics“ Course @ Software University](http://softuni.bg/courses/javascript-basics/). Please submit as homework a single zip / rar / 7z archive holding the solutions (source code) of all below described problems.

## Number Checker

Write a JavaScript function **printNumbers(number)** that accepts as parameter an **integer number.** The functionfinds **all integer numbers** from 1 to n that are **not divisible by 4 or by 5**. Write a JS program **numberChecker.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 20 | 1, 2, 3, 6, 7, 9, 11, 13, 14, 17, 18, 19 |
| -5 | no |
| 13 | 1, 2, 3, 6, 7, 9, 11, 13 |

## Find Min and Max Number

Write a JavaScript function **findMinAndMax(arr)** that accepts as parameter **an array of numbers**. The function finds the **minimum** and the **maximum** number. Write a JS program **minMaxNumbers.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| [1, 2, 1, 15, 20, 5, 7, 31] | Min -> 1  Max -> 31 |
| [2, 2, 2, 2, 2] | Min -> 2  Max -> 2 |
| [500, 1, -23, 0, -300, 28, 35, 12] | Min -> -300  Max -> 500 |

## Properties

Write a JavaScript function **displayProperties()** that displays all the properties of the "document" object in alphabetical order.Write a JS program **docProperties.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |
| --- |
| **Output** |
| activeElement  alinkColor  all  …  xmlVersion |

***Note:*** The output above is a sample and may be different in your browser.

## Create Array

Write a JavaScript function **createArray()** that **allocates array** of 20 integers and initializes each element by its **index multiplied by 5**. Write JS program **arrayBuilder.js** that invokes your and prints the output at the console.

|  |
| --- |
| **Output** |
| 0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 |

## Compare Chars

Write a JavaScript function **compareChars(arr1, arr2)** that **compares two arrays of chars** lexicographically (letter by letter). Write JS program **charComparer.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['1', 'f', '1', 's', 'g', 'j', 'f', 'u', 's', 'q']  ['1', 'f', '1', 's', 'g', 'j', 'f', 'u', 's', 'q'] | Equal |
| ['3', '5', 'g', 'd']  ['5', '3', 'g', 'd'] | Not Equal |
| ['q', 'g', 'q', 'h', 'a', 'k', 'u', '8', '}', 'q', '.', 'h', '|', ';']  ['6', 'f', 'w', 'q', ':', '”', 'd', '}', ']', 's', 'r'] | Not Equal |

## Maximal Sequence

Write a JavaScript function **findMaxSequence(arr)** that finds the **maximal sequence of equal elements** in an array and returns the **result as an array**. If there is more than one sequence with the same maximal length, print the **rightmost** one.Write JS program **sequenceFinder.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| [2, 1, 1, 2, 3, 3, 2, 2, 2, 1] | [2, 2, 2] |
| ['happy'] | [happy] |
| [2, 'qwe', 'qwe', 3, 3, '3'] | [3, 3] |

## Maximal Increasing Sequence

Write a JavaScript function **findMaxSequence(arr)** that finds the **maximal increasing sequence** in an array of numbers and returns the **result as an array**. If there is no increasing sequence the function **returns 'no'**.Write JS program **maxSequenceFinder.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| [3, 2, 3, 4, 2, 2, 4] | [2, 3, 4] |
| [3, 5, 4, 6, 1, 2, 3, 6, 10, 32] | [1, 2, 3, 6, 10, 32] |
| [3, 2, 1] | no |

## Sort Array

Sorting an array means to arrange its elements in increasing order.Write a JavaScript function **sortArray(arr)** to **sort an array**. Use the **"selection sort"** algorithm: find the smallest element, move it at the first position, find the smallest from the rest, move it at the second position, etc. Write JS program **arraySorter.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| [5, 4, 3, 2, 1] | 1, 2, 3, 4, 5 |
| [12, 12, 50, 2, 6, 22, 51, 712, 6, 3, 3] | 2, 3, 3, 6, 6, 12, 12, 22, 50, 51, 712 |

## Most Frequent Number

Write a JavaScript function **findMostFreqNum(arr)** that finds the **most frequent number** in an array. If multiple numbers appear the same maximal number of times, print the **leftmost** of them. Write JS program **numberFinder.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| [4, 1, 1, 4, 2, 3, 4, 4, 1, 2, 4, 9, 3] | 4 (5 times) |
| [2, 1, 1, 5, 7, 1, 2, 5, 7, 3, 87, 2, 12, 634, 123, 51, 1] | 1 (4 times) |
| [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13] | 1 (1 times) |

## Reverse String

Write a JavaScript function **reverseString(str)** that **reverses string and returns it**. Write JS program **stringReverser.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'sample' | elpmas |
| 'softUni' | inUtfos |
| 'java script' | tpircs avaj |

## Check the Brackets

Write a JavaScript function **checkBrackets(str)** **to check if in a given expression the brackets are put correctly**. Write JS program **bracketsChecker.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| '( ( a + b ) / 5 – d )' | correct |
| ') ( a + b ) )' | incorrect |
| '( b \* ( c + d \*2 / ( 2 + ( 12 – c / ( a + 3 ) ) ) )' | incorrect |

## Substring Count

Write a JavaScript function **countSubstringOccur(arr)** that accepts as parameter an array of 2 elements **arr [keyword, text]. The function finds how many times a substring is contained in a given text (perform case insensitive search).** Write JS program **substringSearch.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['in', 'We are living in a yellow submarine. We don't have anything else. Inside the submarine is very tight. So we are drinking all the day. We will move out of it in 5 days.'] | 9 |
| ['your', 'No one heard a single word you said. They should have seen it in your eyes. What was going around your head.'] | 2 |
| ['but', 'But you were living in another world tryin' to get your message through.'] | 1 |

## Replace the White-Spaces

Write a JavaScript function **replaceSpaces(str)** that replaces the **white-space characters** in a text with **&nbsp;**. Write JS program **spaceReplacer.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'But you were living in another world tryin' to get your message through' | Butyouwerelivinginanotherworldtryin'togetyourmessagethrough |

## Palindromes

Write a JavaScript function **findPalindromes(str)** that extracts from a given text **all palindromes**, e.g. "ABBA", "lamal", "exe". Write JS program **palindromesExtract.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'There is a man, his name was Bob.' | a, bob |

## Most Frequent Word

Write a JavaScript function **findMostFreqWord(str)** that **finds the most frequent word** in a text and prints it, as well as **how many times it appears** in format "**word -> count**". Consider any non-letter character as a word separator. Ignore the character casing. If several words have the same maximal frequency, print all of them in alphabetical order. Write JS program **frequentWord.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'in the middle of the night' | the -> 2 times |
| 'Welcome to SoftUni. Welcome to Java. Welcome everyone.' | welcome -> 3 times |
| 'Hello my friend, hello my darling. Come on, come here. Welcome, welcome darling.' | come -> 2 times  darling -> 2 times  hello -> 2 times  my -> 2 times  welcome -> 2 times |

## Cards Frequencies

Write a JavaScript function **findCardFrequency(str)** that that accepts the following parameters: string of several cards (face + suit), separated by a space. The function calculates and prints at the console the frequency of each card face in format "**card\_face -> frequency**". The frequency is calculated by the formula **appearances / N** and is expressed in percentages with exactly 2 digits after the decimal point. The card faces with their frequency should be printed in the order of the card face's first appearance in the input. The same card can appear multiple times in the input, but its face should be listed only once in the output. Write JS program **cardFrequencies.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| '8♥ 2♣ 4♦ 10♦ J♥ A♠ K♦ 10♥ K♠ K♦' | 8 -> 10.00%  2 -> 10.00%  4 -> 10.00%  10 -> 20.00%  J -> 10.00%  A -> 10.00%  K -> 30.00% |
| 'J♥ 2♣ 2♦ 2♥ 2♦ 2♠ 2♦ J♥ 2♠' | J -> 22.22%  2 -> 77.78% |
| '10♣ 10♥' | 10 -> 100.00% |

# Problems for Champions

The next few problems are not mandatory. Implement them to challenge your skills.

## \* Extract Element Content

Write a JavaScript function **extractContent(str)** that extracts the **text content from given HTML fragment** (given as string). The function should return anything that is in a tag, **without the tags.** Write JS program **contentExtracter.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| '<p>Hello</p><a href='http://w3c.org'>W3C</a>' | HelloW3C |

***Hint:*** Create an element and use its **innerHTML** and **innerText** properties.

## \*Replace <a> Tag

Write a JavaScript function **replaceATag(str)** that replaces in a HTML document given as string **all the tags <a href="…">…</a>** with corresponding **tags [URL=…]…/URL]**.Write JS program **aTagReplacer.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| '<ul>  <li>  <a href=<http://softuni.bg>>SoftUni</a>  </li>  </ul>' | <ul>  <li>  [URL href=<http://softuni.bg>]SoftUni[/URL]  </li>  </ul> |

## \*\*Text Modifier

Write a JavaScript function **fixCasing(str)** that accepts a text as a parameter. The function must **change the text in all regions** as follows:

* <upcase>text</upcase> to uppercase
* <lowcase>text</lowcase> to lowercase
* <mixcase>text</mixcase> to mixed casing (randomly)

Write JS program **caseFixer.js** that invokes your function with the sample input data below and prints the output at the console. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 'We are <mixcase>living</mixcase> in a <upcase>yellow submarine</upcase>. We <mixcase>don't</mixcase> have <lowcase>anything</lowcase> else.' | We are LiVinG in a YELLOW SUBMARINE. We dOn'T have anything else. |

# Exam Problems

All problems below are given from the JavaScript Basics exam from **28-July-2014**. You can submit your solutions [here](http://judge.softuni.bg/Contests/20/JavaScript-Basics-Exam-28-July-2014). **You are not obligated** to submit any of them in your homework, but it is highly recommend that you solve some or all of them so you can be well prepared for the upcoming exam. You may read [this post](https://softuni.bg/forum/questions/details/1627) to see how to submit JS code in the Judge system.

## – \*Double Rakiya Numbers

A "**double rakiya number**" is an integer that **contains a sequence of 2 digits twice** (without overlapping). For example "2**31**56**31**2" is a "double rakiya number" because it contains "**31**" twice. Other examples of "double rakiya numbers" are: **1212**, **3333**, 2**03**1**03**, 5**21**0**21**7, **21**2121**21**, and **5555**5. Examples of non-"double rakiya numbers" are: 333, 5, 111222, 1234567131, and 12213114.

Write a JavaScript function that takes as input two numbers (**start** and **end**) and prints at the console a HTML list holding all numbers in the range [**start**…**end**], along with a link to view details about all "double-rakiya numbers" in that range. Please use the format from the below examples.

### Input

The input is passed to the first JavaScript function found in your code as **array of two strings**: the numbers **start** and **end**. The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

Print at the console the above described **HTML** **list** in the same format like the examples below. Don't add additional spaces. **Whitespace** and character **casing** are important, so please use the same as in the below examples.

### Constraints

* The numbers **start** and **end** are positive integers in the range [1…1 000 000 000] and **start** ≤ **end**.
* Allowed working time for your program: 0.2 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5  8 | <ul>  <li><span class='num'>5</span></li>  <li><span class='num'>6</span></li>  <li><span class='num'>7</span></li>  <li><span class='num'>8</span></li>  </ul> |
| 11210  11215 | <ul>  <li><span class='num'>11210</span></li>  <li><span class='rakiya'>11211</span><a href="view.php?id=11211">View</a></li>  <li><span class='rakiya'>11212</span><a href="view.php?id=11212">View</a></li>  <li><span class='num'>11213</span></li>  <li><span class='num'>11214</span></li>  <li><span class='num'>11215</span></li>  </ul> |
| 55555  55560 | <ul>  <li><span class='rakiya'>55555</span><a href="view.php?id=55555">View</a></li>  <li><span class='rakiya'>55556</span><a href="view.php?id=55556">View</a></li>  <li><span class='rakiya'>55557</span><a href="view.php?id=55557">View</a></li>  <li><span class='rakiya'>55558</span><a href="view.php?id=55558">View</a></li>  <li><span class='rakiya'>55559</span><a href="view.php?id=55559">View</a></li>  <li><span class='num'>55560</span></li>  </ul> |

## \*String Matrix Rotation

You are given a **sequence of text lines**. Assume these text lines form a **matrix of characters** (pad the missing positions with spaces to build a rectangular matrix). Write a program to **rotate the matrix** by 90, 180, 270, 360, … degrees. Print the result at the console as sequence of strings. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Rotate(90)** | **Rotate(180)** | **Rotate(270)** |
| hello  softuni  exam |  |  |  |
|  |

### Input

The input is passed to the first JavaScript function found in your code as **array of strings**:

* The first line holds a command in format "**Rotate(X)**" where **X** are the degrees of the requested rotation.
* The next lines to the end contain the **lines of the matrix** for rotation.

The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

Print at the console the **rotated matrix** as a sequence of text lines.

### Constraints

* The rotation **degrees** is positive integer in the range [0…90000], where **degrees** is **multiple of 90**.
* The number of matrix lines is in the range [1…**1 000**].
* The matrix lines are **strings** of length 1 … 1 000.
* Allowed working time: 0.2 seconds. Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| Rotate(90)  hello  softuni  exam | esh  xoe  afl  mtl  uo  n  i | Rotate(180)  hello  softuni  exam | maxe  inutfos  olleh | Rotate(270)  hello  softuni  exam | i  n  ou  ltm  lfa  eox  hse |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| Rotate(720)  js  exam | js  exam | Rotate(810)  js  exam | ej  xs  a  m | Rotate(0)  js  exam | js  exam |

## \*Sort Table

You are given a **HTML table** with 3 columns: **product**, **price** and **votes**. Write a JavaScript function to sort the table rows by **price** (as number, increasingly).

### Input

The input is passed to the first JavaScript function found in your code as **array of strings** in the format of the examples below. The HTML table will always have a header row and 3 columns: product, price and votes. **No whitespace** will be found between the tags and between the tags and the tags values.

The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

Print at the console the **HTML table** sorted by column "**Price**" (as number, increasingly). Please don't change anything in the table, just rearrange its data rows. When several rows hold **equal prices**, use the **product name as** **second sort criteria** (sort by product name alphabetically).

### Constraints

* The **number of rows** in the table is in the range [1…10 000].
* All **prices** are number in the range [0…100 000].
* Allowed working time: 0.2 seconds. Allowed memory: 16 MB.

### Examples

|  |
| --- |
| **Input** |
| <table>  <tr><th>Product</th><th>Price</th><th>Votes</th></tr>  <tr><td>Vodka Finlandia 1 l</td><td>**19.35**</td><td>+12</td></tr>  <tr><td>Ariana Radler 0.5 l</td><td>**1.19**</td><td>+33</td></tr>  <tr><td>Laptop HP 250 G2</td><td>**629**</td><td>+1</td></tr>  <tr><td>Kamenitza Grapefruit 1 l</td><td>**1.85**</td><td>+7</td></tr>  <tr><td>Ariana Grapefruit 1.5 l</td><td>**1.85**</td><td>+7</td></tr>  <tr><td>Coffee Davidoff 250 gr.</td><td>**11.99**</td><td>+11</td></tr>  </table> |
| **Output** |
| <table>  <tr><th>Product</th><th>Price</th><th>Votes</th></tr>  <tr><td>Ariana Radler 0.5 l</td><td>**1.19**</td><td>+33</td></tr>  <tr><td>Ariana Grapefruit 1.5 l</td><td>**1.85**</td><td>+7</td></tr>  <tr><td>Kamenitza Grapefruit 1 l</td><td>**1.85**</td><td>+7</td></tr>  <tr><td>Coffee Davidoff 250 gr.</td><td>**11.99**</td><td>+11</td></tr>  <tr><td>Vodka Finlandia 1 l</td><td>**19.35**</td><td>+12</td></tr>  <tr><td>Laptop HP 250 G2</td><td>**629**</td><td>+1</td></tr>  </table> |

## \*Soccer Results

You are given a sequence of soccer results in format "**homeTeam / awayTeam: homeGoals-awayGoals**". Your task is to write a JavaScript function that prints at the console a **JSON string** that holds the **teams**, and for each team **goals scored**, **goals conceded** and a list of teams **that had a match with this team** in the same format like at the below examples.

## Input

The input data is passed to the first JavaScript function found in your code as **array of strings**. Each input line holds a match description in format "**homeTeam / awayTeam: homeGoals-awayGoals**". The input data will always be valid and in the format described. There is no need to check it explicitly.

Note that any two teams may have played multiple matches.

## Output

Print at the console a **JSON string** that holds the **teams** (in alphabetical order) and for each team **goals scored**, **goals conceded** and a list of teams **that had a match with this team** (in alphabetical order). **Duplicated teams** should be printed once only (all strings are **case-sensitive**). Please follow exactly the **JSON format** from the example below.

## Constraints

* The numbers of **input lines** is between 1 and 10 000.
* The values **homeTeam** and **awayTeam** consist of Latin letters and spaces. Their **length** is between 1 and 50 characters. Leading and trailing **whitespace** should be removed.
* The values **homeGoals** and **awayGoals** are integer numbers in the range [0…99].
* **Whitespace** may be found or missing around the separators "**/**', "**:**" and "**-**".
* Allowed working time: 0.2 seconds. Allowed memory: 16 MB.

## Examples

|  |
| --- |
| **Input** |
| Germany / Argentina: 1-0  Brazil / Netherlands: 0-3  Netherlands / Argentina: 0-0  Brazil / Germany: 1-7  Argentina / Belgium: 1-0  Netherlands / Costa Rica: 0-0  France / Germany: 0-1  Brazil / Colombia: 2-1 |
| **Output** |
| {"Argentina":{"goalsScored":1,"goalsConceded":1,"matchesPlayedWith":["Belgium","Germany","Netherlands"]},"Belgium":{"goalsScored":0,"goalsConceded":1,"matchesPlayedWith":["Argentina"]},"Brazil":{"goalsScored":3,"goalsConceded":11,"matchesPlayedWith":["Colombia","Germany","Netherlands"]},"Colombia":{"goalsScored":1,"goalsConceded":2,"matchesPlayedWith":["Brazil"]},"Costa Rica":{"goalsScored":0,"goalsConceded":0,"matchesPlayedWith":["Netherlands"]},"France":{"goalsScored":0,"goalsConceded":1,"matchesPlayedWith":["Germany"]},"Germany":{"goalsScored":9,"goalsConceded":1,"matchesPlayedWith":["Argentina","Brazil","France"]},"Netherlands":{"goalsScored":3,"goalsConceded":0,"matchesPlayedWith":["Argentina","Brazil","Costa Rica"]}} |