# Homework: JavaScript Syntax

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.

## Powerful Cars

Write a JavaScript function **convertKWtoHP(number)** that accepts number variable to convert car’s **kW** into **hp** (horse power). Write a JS program **powerfulCars.js** that converts a few sample values to **hp** (see the examples below). The result should be printed on the console, rounded up to the second sign after the decimal point. Run the program through **Node.js**. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 75 | 100.58 |
| 150 | 201.15 |
| 1000 | 1341.02 |

***Hint:*** Use the link: <http://www.rapidtables.com/convert/power/how-hp-to-kw.htm>.

## Simple Calculations

Write a JavaScript function **roundNumber(number)** that rounds floating-point number using **Math.round()**, **Math.floor()**. Write a JS program **roundingNumbers.js** that rounds a few sample values. Run the program through **Node.js**. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 22.7 | 22  23 |
| 12.3 | 12  12 |
| 58.7 | 58  59 |

## Check if Number is Even

Write a JavaScript function **evenNumber(number)** that checks if an **integer number** is **even.** Write JS program **evenChecker.js** to check if a few numbers are even. The result should be printed on the console (**true or false**). Run the program through **Node.js**. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3 | false |
| 127 | false |
| 588 | true |

## Check if Number is Prime

Write a JavaScript function **isPrime(number)** that checks if an **integer number** is **prime.** Write JS program **primeChecker.js** that checks if a few numbers are prime. The result should be printed on the console (**true** or **false**) on the console. Run the program through **Node.js**. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 7 | true |
| 254 | false |
| 587 | true |

## Division by 3

Write a JavaScript function **divisionBy3(number)** that finds the **sum of digits of integer number n (n > 9)** and **checks if the sum is divided by 3 without remainder.** Write JS program **divisionChecker.js** to check a few numbers. The result should be printed on the console **(“the number is divided by 3 without remainder”** or **“the number is not divided by 3 without remainder”**). Run the program through **Node.js**. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 12 | the number is divided by 3 without remainder |
| 188 | the number is not divided by 3 without remainder |
| 591 | the number is divided by 3 without remainder |

## Bit Checker

Write a JavaScript function **bitChecker(number)** that finds if the bit 3 an **integer number** (counting from 0) is 1. Write JS program **checkingBits.js** to check a few numbers. The result (**true** or **false**) should be printed on the console. Run the program through **Node.js**. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 333 | true |
| 425 | true |
| 2567564754 | false |

## The Lifetime Supply Calculator

Write a JavaScript function **calcSupply(age, maxAge, food, foodPerDay)** that accepts the following parameters: **your age** (years), **your maximum age** (years), your **favorite food name** (e.g. "chocolate"), **estimate amount of your favorite food per day** (a number).The functioncalculates how many of the food you will eat for the rest of your life. Write JS program **lifetimeSupplyCalc.js** that calculates the amount of a few foods that you will eat. The result should be printed on the console. Run the program through **Node.js**. *Note: we assume that there are no leap years.* Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 38  118  chocolate  0.5 | 14600kg of chocolate would be enough until I am 118 years old. |
| 20  87  fruits  2 | 48910kg of fruits would be enough until I am 87 years old. |
| 16  102  nuts  1.1 | 34529kg of nuts would be enough until I am 102 years old. |

## Cylinder Volume

Write a JavaScript function **calcCylinderVol(arr)** that accepts the following parameters: **radius** and the **height** of a **straight circular cylinder**. The function calculates the volume of the cylinder. Write JS program **cylinderVol.js** that calculates the volume of a few cylinders. The result should be printed on the console. Run the program through **Node.js**. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| [2, 4] | 50.265 |
| [5, 8] | 628.319 |
| [12, 3] | 1357.168 |

## Tree or House

Write a JavaScript function **treeHouseCompare(arr)** that accepts the following parameters: integers **a** and **b**. The function **compares the area of the house and the area of the tree** (Figure 1) and **returns the name** of the figure with bigger area (**house** or **tree**) and the **value of the area**. Write JS program **treehouse.js** that compares a few houses and trees. The result should be printed on the console. Run the program through **Node.js**. Examples:

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Figure 1** |
| [3, 2] | house/12.00 |
| [3, 3] | tree/15.57 |
| [4, 5] | tree/43.24 |

## Digit Checker

Write a JavaScript function **checkDigit(number)** that **finds if the third digit** (right-to-left) of an **integer number n (n>1000)** is 3. Write JS program **digitChecker.js** that checks a few numbers. The result (**true** or **false**) should be printed on the console. Run the program through **Node.js**. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1235 | false |
| 25368 | true |
| 123456 | false |

## Digit as Word

Write a JavaScript function **convertDigitToWord(value)** that **prints the name** of an **integer number n (0<n<10)** in English using switch statement. Write JS program **digitAsWord.**js that prints a few digits on the console. Run the program through **Node.js**. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 8 | eight |
| 3 | three |
| 5 | five |

## Variables

Write a JavaScript function **variablesTypes(arr)** that accepts the following parameters: **name**, **age**, **isMale** (**true** or **false**), array of your favorite foods. The function must return the values of the variables and their types. Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| variablesTypes(['Pesho', 22, true, ['fries', 'banana', 'cake']]) | "My name: Pesho //type is string  My age: 22 //type is number  I am male: true //type is boolean  My favorite foods are: fries,banana,cake //type is object" |

***Constrains:*** Use **typeof()** to find the type of the variables.

## \*Digital Soothsayer

Write a JavaScript function **soothsayer(numsArr, langsArr, citiesArr, carsArr)** that accepts the following parameters (source arrays): **array of numbers**, **array of programming languages**, **array of cities**, **array of cars**. Each array must consist of five elements. The function must return an array **result[]** that consists of one random item from each source array. Write a JS program that prints on the console the following output: “You will work result[0] years on result[1]. You will live in result[2] and drive result[3].”. Run the program through **Node.js**. Examples:

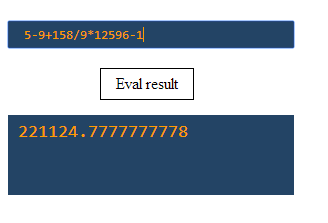
|  |  |
| --- | --- |
| **Input** | **Output** |
| soothsayer([[3, 5, 2, 7, 9], ['Java', 'Python', 'C#', 'JavaScript', 'Ruby'], ['Silicon Valley', 'London', 'Las Vegas', 'Paris', 'Sofia'], ['BMW', 'Audi', 'Lada', 'Skoda', 'Opel']]) --> result = [5, JavaScript, Sofia, Opel] | You will work 5 years on JavaScript. You will live in Sofia and drive Opel. |

# Problems for Champions

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

## \*Calculate Expression

Write a **HTML page** (with **text field**, **button**, and **paragraph**). Write JS program **calcExpression.js** that **calculates** any expression put in the text field and **prints** it in the paragraph. Link the JS file to the HTML file. (100% accuracy is not required). Example:



***Hint:*** Use **eval()** for calculating the result, but first remove the potentially dangerous characters.

# Exam Problems

All problems below are given from the JavaScript Basics exam from **27-July-2014**. You can submit your solutions [here](http://judge.softuni.bg/Contests/19/JavaScript-Basics-Exam-27-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.

## \*Build a Table

Write a JavaScript function that takes as input an array of two numbers (**start** and **end**) and prints at the console a HTML table of 3 columns. The first column should hold a number **num**, changing from **start** to **end**. The second column should hold **num\*num**. The third column should hold "**yes**" if **num** is Fibonacci number or "**no**" otherwise. The table should have header cells titled "**Num**", "**Square**" and "**Fib**". See the below examples.

### Input

The input data comes as **array of two 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 **table** 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 input is passed to the first JavaScript function found in your code as array of 2 elements.
* The numbers **start** and **end** are positive integers in the range [1…1 000 000] and **start** ≤ **end**.
* Allowed working time for your program: 0.2 seconds.
* Allowed memory: 16 MB.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  6 | <table>  <tr><th>Num</th><th>Square</th><th>Fib</th></tr>  <tr><td>2</td><td>4</td><td>yes</td></tr>  <tr><td>3</td><td>9</td><td>yes</td></tr>  <tr><td>4</td><td>16</td><td>no</td></tr>  <tr><td>5</td><td>25</td><td>yes</td></tr>  <tr><td>6</td><td>36</td><td>no</td></tr>  </table> |
| **Input** | **Output** |
| 55  56 | <table>  <tr><th>Num</th><th>Square</th><th>Fib</th></tr>  <tr><td>55</td><td>3025</td><td>yes</td></tr>  <tr><td>56</td><td>3136</td><td>no</td></tr>  </table> |

## \*Reveal Triangles

You are given a sequence of **text lines**, holding small Latin letters. Your task is to **reveal all triangles** in the text by changing their letters with '**\***'. Triangles consist of equal letters in the form of triangle:

|  |  |  |  |
| --- | --- | --- | --- |
| a  aaa | x  xxx  xxxxx | p  ppp  ppppp  ppppppp | etc. |

Triangles can span **different sizes**: 2 lines, 3 lines, 4 lines, etc. Triangles can **overlap** (some letters can take part in several triangles).

### Input

The input data comes as **array of strings**, holding the text lines.

### Output

Print at the console the input data after replacing all triangles by '**\***'.

### Constraints

* The input will be passed to the first JavaScript function found in your code as **array of strings**.
* Each input line will hold 1…100 Latin letters.
* The number of input lines will be in the range [1..100].
* Allowed working time: 0.2 seconds. Allowed memory: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| a**b**cde**x**gh  **bbb**d**xxx**h  abc**xxxxx** | a\*cde\*gh  \*\*\*d\*\*\*h  abc\*\*\*\*\* | a**a**  **aaa**  **aaaa**  **aaaaa** | a\*  \*\*\*  \*\*\*\*  \*\*\*\*\* | a**x**  **xxx**  b  b**bb**  **bbbb** | a\*  \*\*\*  b  b\*\*  \*\*\*\* | d**ff**dsgy**e**fg  **ffff**ey**eee**  jb**fff**ays  dag**fff**dsss  dfdf**a**  dad**aaa**dddf  sd**aaaaa**  d**aaaaaaa**sf | d\*\*dsgy\*fg  \*\*\*\*ey\*\*\*  jb\*\*\*ays  dag\*\*\*dsss  dfdf\*  dad\*\*\*dddf  sd\*\*\*\*\*  d\*\*\*\*\*\*\*sf |

Hint: to simplify your work, you can reveal only triangles of size "2 lines", because all bigger triangles consist of several overlapping triangles of size "2 lines".

## \*Extract Hyperlinks

Write a JavaScript function to **extract all hyperlinks** (**<href=…>**) from given text. The text comes as **array of strings**, passed as parameter to your function. Print at the console the **href** values in the text.

The input text is **standard HTML code**. It may hold many tags and can be formatted in many different forms (with or without whitespace). The **<a>** elements may have many attributes, not only **href**. You should extract only the values of the **href** attributes of all **<a>** elements.

### Input

The input data comes as **array of strings**, holding the text lines.

### Output

Print at the console the **href** values in the text, each at a separate line, in the order they come from the input.

### Constraints

* The input will be passed to the first JavaScript function found in your code as **array of strings**.
* The input will be **well formed HTML fragment** (all tags and attributes will be correctly closed).
* Attribute values will never hold tags and hyperlinks, e.g. "**<img alt='<a href="hello">' />**" is invalid.
* Commented links are also extracted.
* The number of input lines will be in the range [1..100].
* Allowed working time: 0.2 seconds. Allowed memory: 16 MB.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| <a href="http://softuni.bg" class="new"></a> | http://softuni.bg |
| <p>This text has no links</p> |  |
| <!DOCTYPE html>  <html>  <head>  <title>Hyperlinks</title>  <link href="theme.css" rel="stylesheet" />  </head>  <body>  <ul><li><a **href="/"** id="home">Home</a></li><li><a  class="selected" **href=/courses**>Courses</a>  </li><li><a **href =**  **'/forum'** >Forum</a></li><li><a class="href"  onclick="go()" **href= "#"**>Forum</a></li>  <li><a id="js" **href =**  **"javascript:alert('hi yo')"** class="new">click</a></li>  <li><a id='nakov' **href =**  **http://www.nakov.com** class='new'>nak</a></li></ul>  <a **href="#empty"**></a>  <a id="href">href='fake'<img src='http://abv.bg/i.gif'  alt='abv'/></a><a **href="#"**>&lt;a href='hello'&gt;</a>  <!-- This code is commented:  <a href="#commented">commentex hyperlink</a> -->  </body> | /  /courses  /forum  #  javascript:alert('hi')  http://www.nakov.com  #empty  #  #commented |

## \*Concerts

You are given a **timetable** for the upcoming rock concerts. It consists of **bands**, **towns**, **dates** and **venues**, separated by '**|**'. Your task is to write a JavaScript function that prints at the console a **JSON string** that holds the **towns**, **venues** for each town and **list of bands** for each venue (see the example below).

### Input

The input data is passed to the first JavaScript function found in your code as **array of strings**. Each input line holds a concert description: **band**, **town**, **date** (in format **dd-MMM-YYYY**) and **venue**, separated by '**|**'. 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 a **JSON string** that holds the **towns** (in alphabetical order), the **venues** for each town (in alphabetical order) and a **list of bands** for each venue (in alphabetical order). **Duplicates** should be removed (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 **band**, **town** and **venue** consist of Latin letters, spaces and punctuation marks. Their length is between 1 and 300 characters. Leading and trailing **whitespace** should be removed.
* Allowed working time for your program: 0.2 seconds.
* Allowed memory: 16 MB.

### Examples

|  |
| --- |
| **Input** |
| ZZ Top | London | 2-Aug-2014 | Wembley Stadium  Iron Maiden | London | 28-Jul-2014 | Wembley Stadium  Metallica | Sofia | 11-Aug-2014 | Lokomotiv Stadium  Helloween | Sofia | 1-Nov-2014 | Vassil Levski Stadium  Iron Maiden | Sofia | 20-June-2015 | Vassil Levski Stadium  Helloween | Sofia | 30-July-2015 | Vassil Levski Stadium  Iron Maiden | Sofia | 26-Sep-2014 | Lokomotiv Stadium  Helloween | London | 28-Jul-2014 | Wembley Stadium  Twisted Sister | London | 30-Sep-2014 | Wembley Stadium  Metallica | London | 03-Oct-2014 | Olympic Stadium  Iron Maiden | Sofia | 11-Apr-2016 | Lokomotiv Stadium  Iron Maiden | Buenos Aires | 03-Mar-2014 | River Plate Stadium |
| **Output** |
| {"Buenos Aires":{"River Plate Stadium":["Iron Maiden"]},"London":{"Olympic Stadium":["Metallica"],"Wembley Stadium":["Helloween","Iron Maiden","Twisted Sister","ZZ Top"]},"Sofia":{"Lokomotiv Stadium":["Iron Maiden","Metallica"],"Vassil Levski Stadium":["Helloween","Iron Maiden"]}} |
| **Comments** |
| Concerts in Buenos Aires @ River Plate Stadium: Iron Maiden  Concerts in London @ Olympic Stadium: Metallica  Concerts in London @ Wembley Stadium: Helloween, Iron Maiden, Twisted Sister, ZZ Top  Concerts in Sofia @ Lokomotiv Stadium: Iron Maiden, Metallica  Concerts in Sofia @ Vassil Levski Stadium: Helloween, Iron Maiden |