# Homework: JavaScript DOM & Events

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.

## Webinar registration form

Create an **HTML** page that looks like the example below. Your task is to write a JavaScript **function** that shows and hides the invoice information if the checkbox “Желая фактура” is checked or not. Example:

|  |  |
| --- | --- |
| **Unchecked checkbox** | **Checked checkbox** |
|  |  |

## Email only field

Write a HTML page holding a text field, a div and a button. Write a JavaScript function that is called when the button is clicked. The function copies the value of the text field and pastes it in the div. If the value of the text field is a valid email (containing a ‘@’ character and a domain e.g. ‘softuni.bg’), the div’s background is set to light green. If the value of the text field is an invalid email, the div’s background is set to red. Example:

|  |  |
| --- | --- |
| **Invalid email** | **Valid email** |
|  |  |

## Create a paragraph generator function

Create a JavaScript function **createParagraph(id, text).** It creates a paragraph DOM element with text (the text value that has been passed as an argument). After the element has been created it is appended to a html element with id that has been passed as an argument in the function. Example:

|  |  |
| --- | --- |
| **Input HTML** | **Output HTML** |
| <html>  <body>  <div id="wrapper"></div>  <script src="script.js"></script>  </body>  </html> | <html>  <body>  <div id="wrapper">  <p>Some text</p>  </div>  <script src="script.js"></script>  </body>  </html> |
| createParagraph('wrapper', 'Some text'); |

## Create HTMLGenerator object

Create a JavaScript object named **HTMLGenerator**. The object has a couple of functions attached to it:

* **createParagraph(id, text)** -> creates a <p></p> HTML element that is attached to the id of the element, passed as an argument. The paragraph’s text is passed as an argument of the function as well.
* **createDiv(id, class)** -> creates a <div></div> HTML element that is attached to the id of the element, passed as an argument. The div’s class is passed as an argument of the function as well.
* **createLink(id, text, url)** -> creates a <a></a> HTML element that is attached to the id of the element, passed as an argument. The link’s text and url are passed as arguments of the function as well.

Example:

|  |  |
| --- | --- |
| **Predefined HTML** | |
| <**body**>  <**div** id="wrapper"></**div**>  <**div** id="book"></**div**>  <**script** src="script.js"></**script**> </**body**> | |
| **Input** | **Output** |
| HTMLGen.createParagraph('wrapper', 'Soft Uni');  HTMLGen.createDiv('wrapper', 'section');  HTMLGen.createLink('book', 'C# basics book', 'http://www.introprogramming.info/'); | <**div** id="wrapper">  <**p**>Soft Uni</**p**>  <**div** class="section"></**div**> </**div**> <**div** id="book">  <**a** href="http://www.introprogramming.info/">C# basics book</**a**> </**div**> |

# Exam Problems

All problems below are given from the JavaScript Basics exam from **4-Septmeber-2014**. You can submit your solutions [here](http://judge.softuni.bg/Contests/31/JavaScript-Basics-Exam-4-September-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.

## \*Keep The Change

Don Vlado likes to eat at expensive restaurants. In such restaurants it is accepted that a customer should tip (leave extra change when paying his bill). However, don Vlado happens to be very stingy and wants to spare every penny he can when tipping at his favorite restaurant. Help him by **calculating his exact tip**!

Don Vlado's tip very much depends on his **mood**:

* When *happy*, don Vlado tips for 10% of the bill
* When *married*, don Vlado tips for 0.05% of the bill
* When *drunk*, don Vlado tips for (15% of the bill)**n**, where **n** is the **first digit** of the tip. (e.g. if the bill is 200, **30** is **15% of the bill**. **3** is the **first digit of 30**, so Don Vlado leaves the tip **303 = 30 \* 30 \* 30 = 27000**)
* In every other scenario, don Vlado is simply grumpy and tips for only 5% of the bill

### Input

The input data will be received as an **array**. It contains two arguments – the first one is don Vlado’s **bill**. The second one is **don Vlado’s mood**.

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

### Output

The output consists of only one line – don Vlado’s tip, **rounded to 2 places after the decimal point**. Use the **toFixed()** function.

### Constraints

* The billwill be a number no greater than 100000.
* Time limit: 0.3 sec. Memory limit: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 120.44  happy | 12.04 | 1230.83  drunk | 184.62 | 716.00  bored | 35.80 |

## \*The Numbers

*"The numbers, Mason, what do they mean?"*

We’ve just received a ton of unreadable signals from the International Space Station. We've lost all contact with the team up there, and all we got are these messages. Aliens? Might be. Can you please clear up the messages for us, so we can pass them to the decryption team?

Your job is to **clear the text from any unnecessary symbols** (only the numbers are needed) and **convert the remaining number sequences to hex format**. If a hex value has less than 4 characters, you need to **add leading zeros**. Finally, you need toplace a **"0x" prefix before each hex value** and **concatenate them all with dashes** '-'.

For example, we have the following message: "**5tffwj(//\*7837xzc2---34rlxXP%$**". After trimming the unnecessary data (non-numeric characters), we've got the numbers **5**, **7837**, **2** and **34** left. We convert them to hex: **5**, **1E9D**, **2**, **22**; add leading zeros where needed: **0005, 1E9D, 0002, 0022**, place 0x before each oneand concatenate them with dashes: **0x0005-0x1E9D-0x0002-0x0022**.

(Note: hex values *MUST* be uppercase)

### Input

The input data will be received as an **array**. It contains one argument – the initial message you need to transform.

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

### Output

The output consists of only one line – the transformed message.

### Constraints

* The message will be no longer than 10000 characters.
* The message will consist of ASCII characters only.
* The numbers encoded in the message will be in the range **[0…65 535]**.
* Time limit: 0.3 sec. Memory limit: 16 MB.

### Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 5tffwj(//\*7837xzc2---34rlxXP%$”. | 0x0005-0x1E9D-0x0002-0x0022 | 482vMWo(\*&^%$213;k!@41341((()&^>><///]42344p;e312 | 0x01E2-0x00D5-0xA17D-0xA568-0x0138 | 20 | 0x0014 |

## \*To The Stars!

The year is 2185 and the SSR Normandy spaceship explores our galaxy. Unfortunately, the ship suffered severe damage in the last battle with Batarian pirates, and her navigation system is broken. Your task is to write a JavaScript program to help the Normandy safely navigate through the stars back home.

The navigation field is a 2D grid. You are given the names of **3 star systems**, along with **their coordinates(sx,sy)** and **the Normandy’s initial coordinates(nx, ny)**. Assume that a **star’s coordinates are in the center of a 2x2 rectangle**. The Normandy **always** **moves in an upwards direction, 1 unit every turn**. Your task is to inform the Normandy of its current location during its movement.



### The Normandy can only be at one location at a time. The possible locations are "<star1 name>", "<star2 name>", "<star3 name>" and "space". In other words, if the Normandy is in the range of Alpha-Centauri, her location is "alpha-centauri". If she's not in the range of any star system, her location is just "space".

Star systems will **NOT** overlap.

*Example*: the Normandy’s initial location is at (8, 1). There, she in outside of any star system, so she is in "space". She starts moving up and her next two locations at (8, 2) and (8, 3) are again in "space". After that, at (8, 4), (8, 5), (8, 6) she is in the range of Alpha-Centauri – therefore, she is in "alpha-centauri". Her final location (8, 7) is outside any star, and her location is "space".

### Input

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

* The first arguments will contain each star system's name and coordinates in the format "<**name**> <**x**> <**y**>", separated by spaces. The **name will be a single word, without spaces**.
* The fourth argument will contain the Normandy’s initial coordinates in the format "<**x**> <**y**>", separated by spaces.
* The fifth, last argument, will contain the number **n** – the number of turns the Normandy will be moving.

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

### Output

The output consists of **n + 1** lines – the Normandy’s **initial** location, plus the **locations she was in during her movement**, each on a separate line. All locations must be printed **lowercase**.

### Constraints

* The grid dimensions will be no larger than 30x30.
* All star systems will be squares with a fixed size: 2x2.
* The turns will be no more than 20.
* Time limit: 0.3 sec. Memory limit: 16 MB.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| Sirius 3 7  Alpha-Centauri 7 5  Gamma-Cygni 10 10  8 1  6 | space  space  space  alpha-centauri  alpha-centauri  alpha-centauri  space | Terra-Nova 16 2  Perseus 2.6 4.8  Virgo 1.6 7  2 5  4 | perseus  virgo  virgo  virgo  space |

## \*Cloud Manager

Write a program that reads file information from the console and **groups the files according to their extensions** in the format **<file-extension> <[file1, file2, … ]> <total memory in MB>**, where total memory is the sum of the sizes of the respective files. For example, given the files:

* sentinel **.exe** 15MB
* zoomIt **.msi** 3MB
* skype **.exe** 45MB
* trojanStopper **.bat** 23MB
* kindleInstaller **.exe** 120MB
* setup **.msi** 33.4MB
* winBlock **.bat** 1MB

The result should be:

* **.bat** [trojanStopper, winBlock] 24MB
* **.msi** [setup, zoomIt] 36.4MB
* **.exe** [kindleInstaller, sentinel, skype] 180MB

Extension lines should be **sorted by the extension names**. The files themselves should **also** **be** **ordered alphabetically**. Finally, the information is converted to JSON format and printed:

### {".bat":{"files":["trojanStopper","winBlock"],"memory":"24.00"},".exe":{"files":["kindleInstaller","sentinel","skype"],"memory":"180.00"},".msi":{"files":["setup","zoomIt"],"memory":"36.40"}}

### Input

The input is passed to the first JavaScript function found in your code as **array of several strings:** each string will contain information about a file in the format **<name> <extension> <memory>**, separated by spaces.

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

### Output

Print the file information in **JSON format** as shown in the examples. The memory should be printed with **2 places after the decimal point**.

### Constraints

* File memory will be in the range **[0..1000000]**.
* Time limit: 0.3 sec. Memory limit: 16 MB.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| sentinel .exe 15MB  zoomIt .msi 3MB  skype .exe 45MB  trojanStopper .bat 23MB  kindleInstaller .exe 120MB  setup .msi 33.4MB  winBlock .bat 1MB | {"**.bat**":{"files":["trojanStopper","winBlock"],"memory":"24.00"},"**.exe**":{"files":["kindleInstaller","sentinel","skype"],"memory":"180.00"},"**.msi**":{"files":["setup","zoomIt"],"memory":"36.40"}} |  | eclipse .tar.gz 198.00MB  uTorrent .gyp 33.02MB  nodeJS .gyp 14MB  nakov-naked .jpeg 3MB  gnuGPL .pdf 5.6MB  skype .tar.gz 66MB  selfie .jpeg 7.24MB  myFiles .tar.gz 783MB | {"**.gyp**":{"files":["nodeJS","uTorrent"],"memory":"47.02"},"**.jpeg**":{"files":["nakov-naked","selfie"],"memory":"10.24"},"**.pdf**":{"files":["gnuGPL"],"memory":"5.60"},"**.tar.gz**":{"files":["eclipse","myFiles","skype"],"memory":"1047.00"}} |