# Lab: Objects & Classes

Problems for in-class lab for the ["JavaScript Advanced" course @ SoftUni](https://softuni.bg/courses/js-advanced). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/2370/Lab-Objects-Classes>.

## Towns to JSON

You're tasked to create and print a JSON from a text table. You will receive input as an array of strings, where each string represents a row of a table, with values on the row encompassed by pipes **"|"** and optionally spaces. The table will consist of exactly 3 columns **"Town"**, **"Latitude"** and **"Longitude"**. The **latitude** and **longitude** columns will always contain **valid numbers**. Check the examples to get a better understanding of your task.

### Input

The **input** comes as an array of strings – the first string contains the table’s headings, each next string is a row from the table.

### Output

* The **output** should be an array of objects wrapped in **JSON.stringify()**.
* **Latitude** and **longitude** must be parsed to **numbers,** and represented till the **second digit after the decimal point**!

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['| Town | Latitude | Longitude |',  '| Sofia | 42.696552 | 23.32601 |',  '| Beijing | 39.913818 | 116.363625 |'] | [{"Town":"Sofia",  "Latitude":42.7,  "Longitude":23.32  }, {"Town":"Beijing",  "Latitude":39.91,  "Longitude":116.36  }] |
| ['| Town | Latitude | Longitude |',  '| Veliko Turnovo | 43.0757 | 25.6172 |',  '| Monatevideo | 34.50 | 56.11 |'] | [{"Town":"Veliko Turnovo",  "Latitude":43.08,  "Longitude":25.62  }, {"Town":"Monatevideo",  "Latitude":34.5,  "Longitude":56.11  }] |

## Sum by Town

You’re tasked with calculating the total sum of income for a number of Towns. You will receive an array of strings representing towns and their incomes, every **even** index will be a **town** and every **odd** index will be an **income** belonging to that town. Create an object that will hold all the **towns as keys** and their **total income** (the sum of their incomes) **as values** to those keys and print it as a JSON.

### Input

The **input** comes as an array of strings - each even index is the name of a town and each odd index is an income belonging to that town.

### Output

The **output** should be printed on the console - JSON representation of the object containing all towns and their total incomes.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['Sofia','20','Varna','3','Sofia','5','Varna','4'] | {"Sofia":25,"Varna":7} |
| ['Sofia','20','Varna','3','sofia','5','varna','4'] | {"Sofia":20,"Varna":3,"sofia":5,"varna":4} |

## Populations in Towns

You have been tasked to create a register for different **towns** and their **population**.

### Input

The **input** comes as array of strings. Each element will contain data for a town and its population in the following format:

“{townName} <-> {townPopulation}”

If you receive the same town twice, **you should add** the **given population** to the **current one**.

### Output

As **output**, you must print all the towns, and their population.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['Sofia <-> 1200000',  'Montana <-> 20000',  'New York <-> 10000000',  'Washington <-> 2345000',  'Las Vegas <-> 1000000'] | Sofia : 1200000  Montana : 20000  New York : 10000000  Washington : 2345000  Las Vegas : 1000000 |
| ['Istanbul <-> 100000',  'Honk Kong <-> 2100004',  'Jerusalem <-> 2352344',  'Mexico City <-> 23401925',  'Istanbul <-> 1000'] | Istanbul : 101000  Honk Kong : 2100004  Jerusalem : 2352344  Mexico City : 23401925 |

## From JSON to HTML Table

You’re tasked with creating an HTML table of students and their scores. You will receive a single string representing an **array of objects**, the **table’s headings** should be equal to the **object’s keys**, while **each object’s values** should be a **new entry** in the table. Any **text values** in an object should be **escaped**, in order to avoid introducing dangerous code into the HTML.

### Input

The **input** comes a **single string argument** (the array of objects).

### Output

The **output** should be printed on the console – for each **entry** **row** in the input print the **object** **representing** **it**.

### Note:

Object’s **keys** will always be the **same.** Check more information for the **HTML Entity** [**here**](https://developer.mozilla.org/en-US/docs/Glossary/Entity)**.**

### HTML

You are provided with an HTML file to test your table in the browser.

|  |
| --- |
| index.html |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>FromJSONToHTMLTable</**title**>  <**style**>  **table**,**th**{  **border**: **groove**;  **border-collapse**: **collapse**;  }  **td**{  **border**: 1**px solid black**;  }  **td**,**th**{  **padding**: 5**px**;  }  </**style**> </**head**> <**body**>  <**div id="wrapper"**>  </**div**>  <**script**>  **function** *fromJSONToHTMLTable*(input){  *//Write your code here* }  **window**.onload = **function**(){  **let** container = **document**.getElementById(**'wrapper'**);  container.**innerHTML** = *fromJSONToHTMLTable*([**'[{"Name":"Tomatoes & Chips","Price":2.35},{"Name":"J&B Chocolate","Price":0.96}]'**]);  };  </**script**> </**body**> </**html**> |

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | | **Output** |
| ['[{"Name":"Tomatoes & Chips","Price":2.35},{"Name":"J&B Chocolate","Price":0.96}]'] | | <table>  <tr><th>Name</th><th>Price</th></tr>  <tr><td>Tomatoes &amp; Chips</td><td>2.35</td></tr>  <tr><td>J&amp;B Chocolate</td><td>0.96</td></tr>  </table> |
| ['[{"Name":"Pesho <div>-a","Age":20,"City":"Sofia"}, {"Name":"Gosho","Age":18,"City":"Plovdiv"},{"Name":"Angel","Age":18,"City":"Veliko Tarnovo"}]'] | <table>  <tr><th>Name</th><th>Age</th><th>City</th></tr>  <tr><td>Pesho &lt;div&gt;-a</td><td>20</td><td>Sofia</td></tr>  <tr><td>Gosho</td><td>18</td><td>Plovdiv</td></tr>  <tr><td>Angel</td><td>18</td><td>Veliko Tarnovo</td></tr>  </table> | |

## Lowest Prices in Cities

You will be given several towns, with products and their price. You need to find **the lowest price** for **every product** and **the town it is sold at** for that price.

### Input

The **input** comes as array of strings. Each element will hold data about a **town**, **product**, and **its price** at that town. The **town** and **product** will be **strings**, the **price** will be a **number**. The input will come in the following format:

{townName} | {productName} | {productPrice}

If you receive the same **town** and **product** **more than once,** you should **update** the **old value** with the **new one**.

### Output

As **output** you must print **each** **product** with its **lowest price** and **the town** at which the product is **sold at that** **price**. If **two towns share** the **same lowest price**, print the one that was **entered first**.   
The output, for every product, should be in the following format:

{productName} -> {productLowestPrice} ({townName})

The **order of output** is - **order of entrance**. See the examples for more info.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['Sample Town | Sample Product | 1000',  'Sample Town | Orange | 2',  'Sample Town | Peach | 1',  'Sofia | Orange | 3',  'Sofia | Peach | 2',  'New York | Sample Product | 1000.1',  'New York | Burger | 10'] | Sample Product -> 1000 (Sample Town)  Orange -> 2 (Sample Town)  Peach -> 1 (Sample Town)  Burger -> 10 (New York) |

## Person

Write a **class** that represents a personal record. It has the following properties, all set from the constructor:

* firstName
* lastName
* age
* email

And a method toString(), which prints a summary of the information. See the example for formatting details.

### Input

The constructor function will receive valid parameters.

### Output

The toString()method should **return** a string in the following format:

**"{firstName} {lastName} (age: {age}, email: {email})"**

Submit the class definition as is, **without** wrapping it in any function.

### Example

|  |
| --- |
| Sample Input |
| let person = new Person('Anna', 'Simpson', 22, 'anna@yahoo.com');  console.log(person.toString()); |
| Output |
| Anna Simpson (age: 22, email: anna@yahoo.com) |

## Get Persons

Write a function that returns an array of **Person** objects. Use the class from the previous task, create the following instances, and return them in an array:

|  |  |  |  |
| --- | --- | --- | --- |
| First Name | Last Name | Age | Email |
| Anna | Simpson | 22 | anna@yahoo.com |
| SoftUni |  |  |  |
| Stephan | Johnson | 25 |  |
| Gabriel | Peterson | 24 | g.p@gmail.com |

For any empty cells, do not supply a parameter (call the constructor with less parameters).

### Input / Output

There will be **no input**, the data is static and matches the table above. As **output**, **return an array** with **Person** **instances**.

Submit a function that returns the required output.

## Circle

Write a **class** that represents a **Circle**. It has only one data property - it’s **radius**, and it is set trough the **constructor**. The class needs to have **getter** and **setter** methods for its **diameter** - the setter needs to calculate the radius and change it and the getter needs to use the radius to calculate the diameter and return it.

The circle also has a getter area(), which calculates and **returns** its area.

### Input

The constructor function and diameter setter will receive valid parameters.

### Output

The diameter() and area() getters should **return** numbers.

Submit the class definition as is, **without** wrapping it in any function.

### Examples

|  |  |
| --- | --- |
| Sample Input | Output |
| let c = new Circle(2);  console.log(`Radius: ${c.radius}`);  console.log(`Diameter: ${c.diameter}`);  console.log(`Area: ${c.area}`);  c.diameter = 1.6;  console.log(`Radius: ${c.radius}`);  console.log(`Diameter: ${c.diameter}`);  console.log(`Area: ${c.area}`); | Radius: 2  Diameter: 4  Area: 12.566370614359172  Radius: 0.8  Diameter: 1.6  Area: 2.0106192982974678 |

## Point Distance

Write a JS **class** that represents a **Point**. It has **x** and **y** coordinates as properties, that are set through the constructor, and a **static method** for finding the distance between two points, called distance().

### Input

The distance() method should receive two **Point** objects as parameters.

### Output

The distance() method should **return** a number, the distance between the two point parameters.

Submit the class definition as is, **without** wrapping it in any function.

### Example

|  |  |
| --- | --- |
| Sample Input | Output |
| let p1 = new Point(5, 5);  let p2 = new Point(9, 8);  console.log(Point.distance(p1, p2)); | 5 |