Exercise: DOM Introduction

Problems for in-class lab for the "JS Advanced" Course @SoftUni". Submit your solutions in the SoftUni judge system at https://judge.softuni.bg/Contests/2761/DOM-Introduction-Exercise

Environment Specifics

Please, be aware that every JS environment may **behave differently** when executing code. Certain things that work in the browser are not supported in **Node.js**, which is the environment used by **Judge**.

The following actions are **NOT** supported:

- `.forEach()` with **NodeList** (returned by `querySelector()` and `querySelectorAll()`)
- `.forEach()` with **HTMLCollection** (returned by `getElementsByClassName()` and `element.children`)
- Using the **spread-operator** (`...`) to convert a **NodeList** into an array
- `append()` in Judge (use only `appendChild()`)
- `prepend()`
- Always turn the collection into a **JS array** (forEach, forOf, et.)

If you want to perform these operations, you may use `Array.from()` to first convert the collection into an array.

1. Subtraction

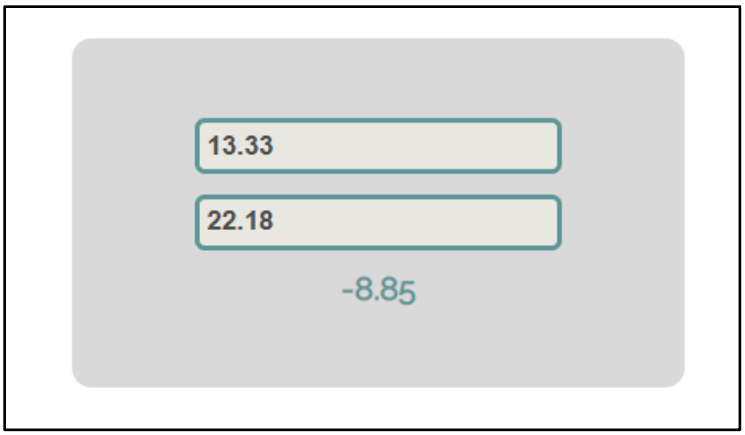
An HTML page holds **two text fields** with ids **"firstNumber"** and **"secondNumber"**. Write a function to **subtract** the values from these text fields and display the result in the **div** named **"result"**.

HTML and JavaScript Code

Implement the above to provide the following functionality:

- Your function should take the values of "**firstNumber**" and "**secondNumber**", **convert** them to numbers, **subtract** the first number from the second one and then append the result to the ** ** with **id="result"**.
- Your function should be able to work with **any 2 numbers** in the inputs, not only the ones given in the example.

Example



Hints

We see that the **textboxes** and the **div** have **id** attributes on them.

```
<div id="wrapper">
    <input type="text" id="firstNumber" value="13.33" disabled>
    <input type="text" id="secondNumber" value="22.18" disabled>
    <div id="result"></div>
</div>
```

We can take the numbers directly from the input field by using the **`getElementById()`** function. After we have taken the elements from the DOM, it's time to do the actual work. We get the values of the two **textboxes**, as one would expect, type is **text**. To get a **number**, we need to use a function to **parse them**.

```
let num1 = document.getElementById('firstNumber').value;
let num2 = document.getElementById('secondNumber').value;
```

All that's left for you to do is append the result to the **div**.

2. Pascal or Camel Case

An **HTML** file is given and your task is to write a function that takes **two string parameters** as an input and transforms the **first parameter** to the type required by the **second parameter**.

- **The first parameter** will be the text that you need to modify depending on the second parameter. The words in it will **always** be **separated by space**.
- **The second parameter** will be either "**`Camel Case`**" or "**`Pascal Case`**". In case of different input, your **output** should be **`"Error!"`**

When the button is clicked the function should convert the first string to either of the cases. The **output** should consist of only **one word** - the string you have modified. Once your **output** is done, you should set it as HTML to the ** element**. For more information, see the examples below:

Example

Input	Output
`"this is an example", "Camel Case"`	`thisIsAnExample`
`"secOND eXamPLE", "Pascal Case"`	`SecondExample`
`"Invalid Input", "Another Case"`	`Error!`

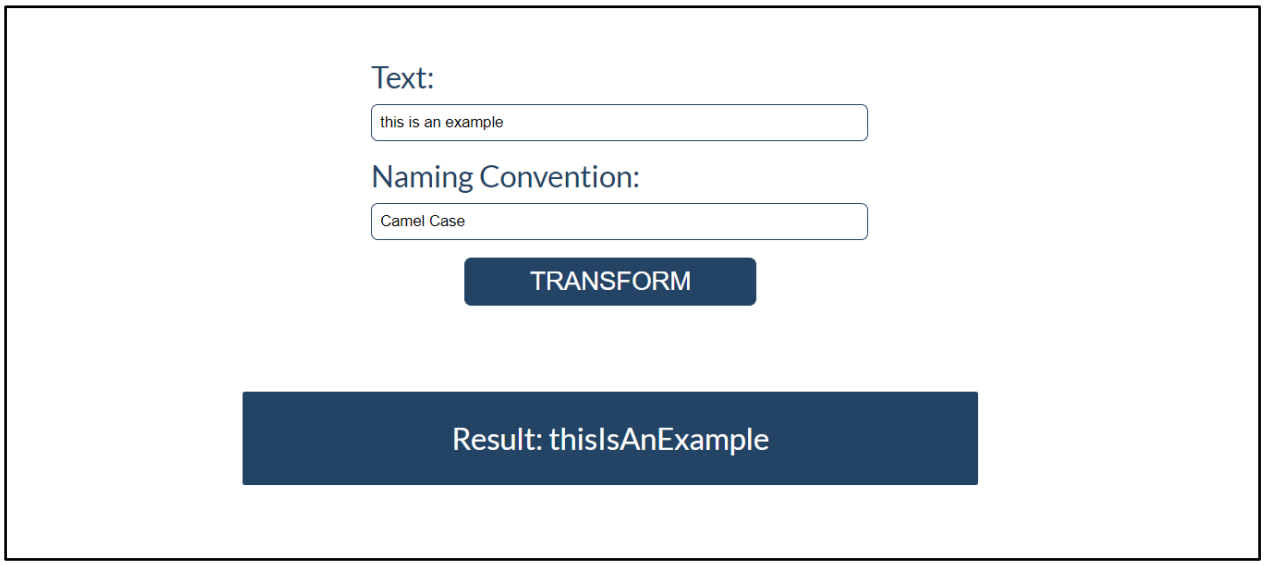
Hints

First, take the two values from the input fields:

```
let input = document.getElementById("text").value;
let currentCase = document.getElementById("naming-convention").value;
```

Then, write a function that generates the result:

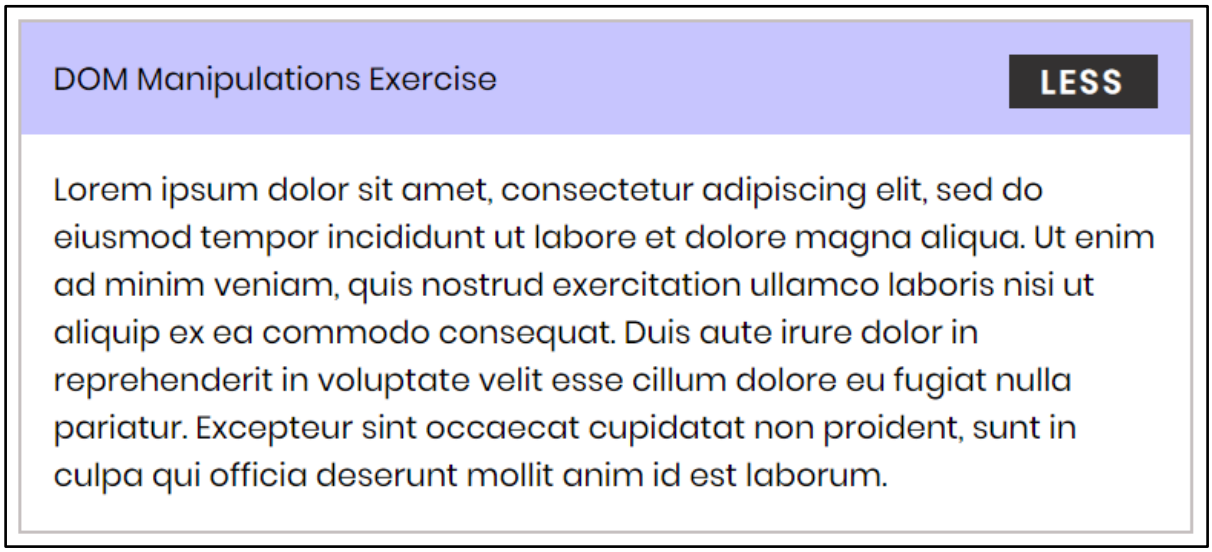
- First, convert all the **letters to lowercase**
- Depending on the command, make the input either **`Pascal Case`** or **`Camel Case`**



3. Accordion

An **HTML** file is given and your task is to show **more**/**less** information. By clicking the **[More] button**, it should **reveal** the content of a **hidden** div and **changes** the text of the button to **[Less]**. When the same link is clicked **again** (now reading **Less**), **hide** the div and **change** the text of the link to **More**. Link action should be **toggleable** (you should be able to click the button an infinite amount of times).

Example



Hints

- To **change** the text content of a button, you could use **`getElementsByClassName`**. However, that returns a **collection** and we need only **one** element from it, so the correct way is to **use** **`getElementsByClassName("button")[0]`** as it will return the needed span element.
- After that we should change the **display style** of the div with an **`id`** "**extra**". If the display style is "**none**", we should **change** it to "**block**" and the **opposite**.

- Along with all of this, we should **change** the text content of the **button** to **`[Less]`**/**`[More]`**.

4. Search in List

An HTML page holds a **list** of towns, a **search** box, and a **`[Search]`** button. Implement the **`search`** function to **bold** and **underline** the items from the list which include the text from the **search** box. Also, print the number of items the current search **matches** in the format **`` `${matches} matches found` ``**.

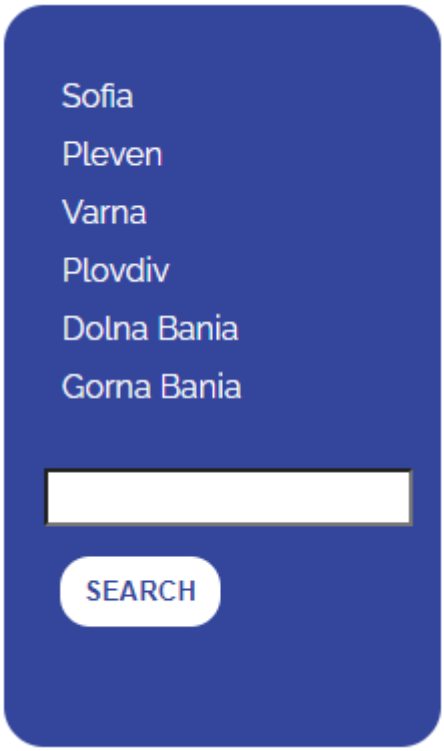
Note: It is necessary to clear the results of the previous search.

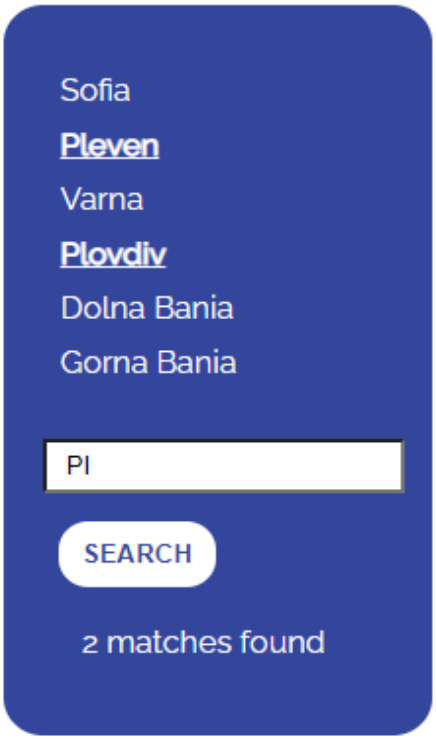
Write your **JavaScript** code in this file:

search.js

```
function search() {
    // TODO
}
```

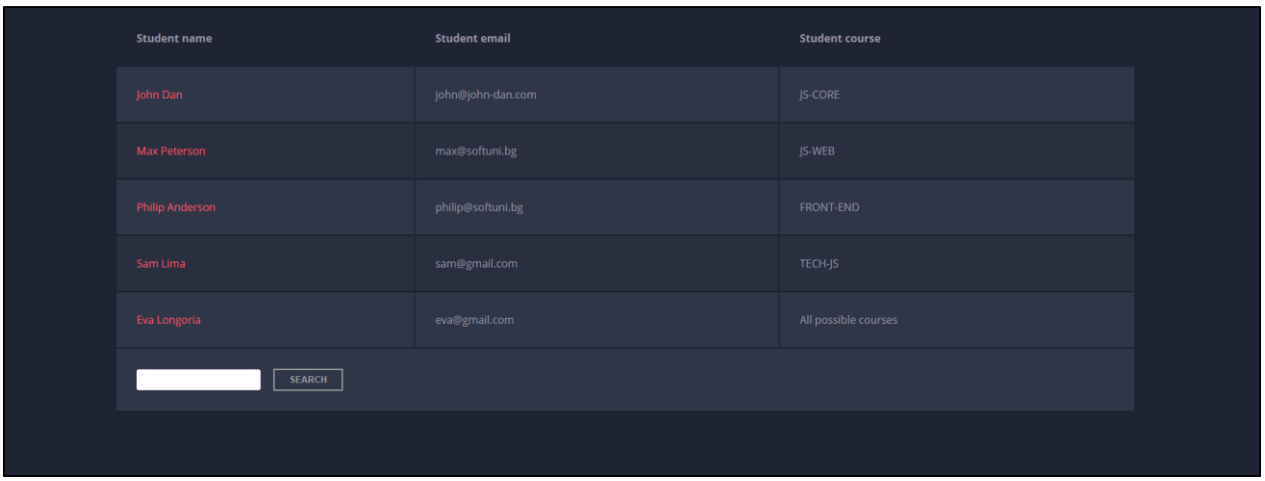
Screenshots





5. Table - Search Engine

Write a function that **searches** in a **table** by given input.



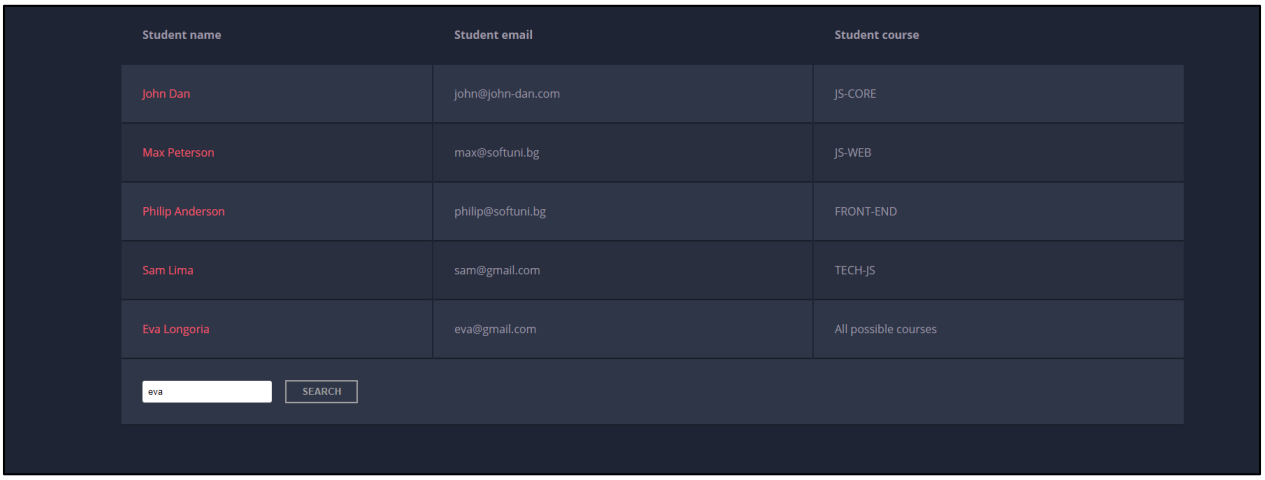
When the **"Search" button** is **clicked**, go through all cells in the table except for the first row (Student name, Student email and Student course) and check if the given input has a match (check for both **full words** and **single letters**).

If any of the rows contain the submitted string, add a **`class select`** to that row. Note that more than one row may contain the given string.

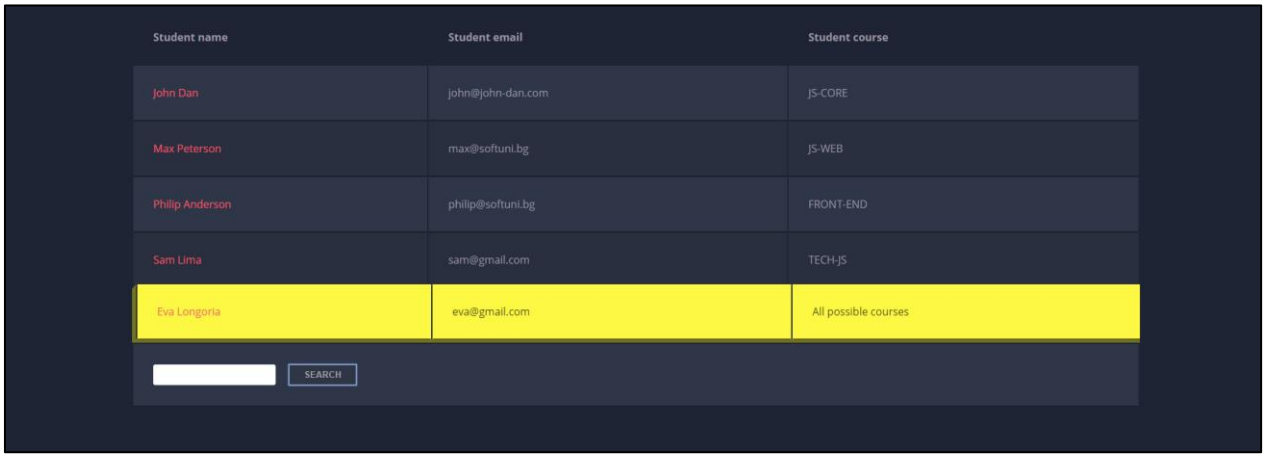
Otherwise, if there is no match, **nothing should happen**.

Note: After every search ("**Search**" button is clicked), **clear the input field** and **remove all already selected classes** (if any) from the previous search, for the **new search** to contain only the **new result**.

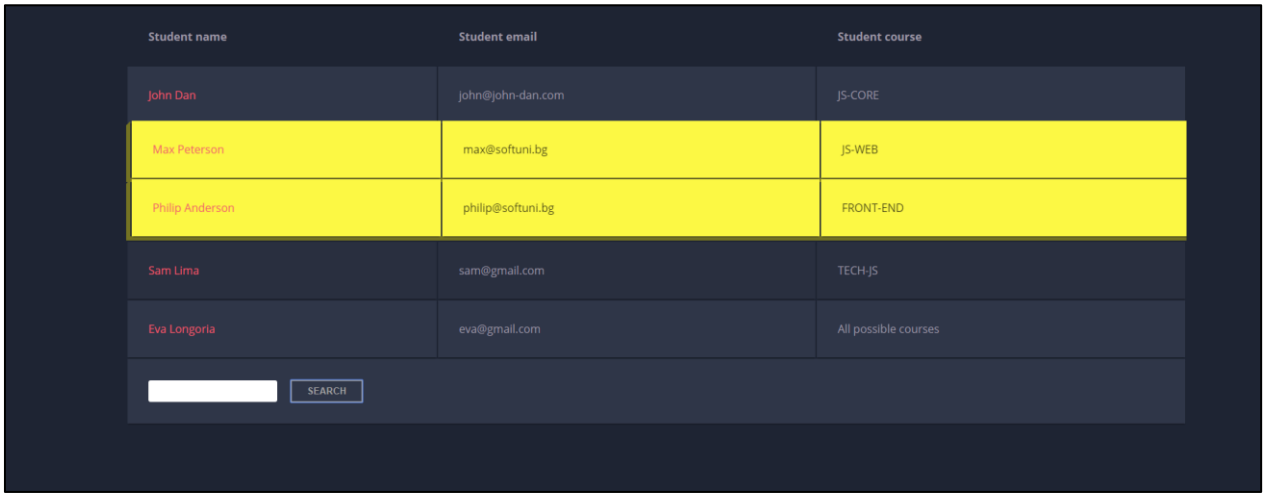
For instance, if we try to find **eva:**



The result should be:



If we try to find all students who have email addresses in **softuni** domain, the expected result should be:



6. Format the Text

Create a functionality that gets a text from **textarea**, formats the given **text** - you need to find out how many **sentences** there are in the text, simply **split** the whole text by **'.'**
Also, every sentence must have at **least 1 character**.

```
<body>
    <h4>Create a functionality which formats the given text into paragraphs</h4>
    <div id="exercise">
        <textarea id="input" cols="30" rows="12"></textarea>
        <button type="button" id="formatItBtn" onclick="solve()">Format</button>
        <div id="output"></div>
    </div>
</body>
```

Generate HTML paragraphs as a string (Use interpolation **string** to create paragraph element: **`<p> {text} </p>`**) and append it to the div with an **id = "output"**.

```
<div id="output">
    <p>JavaScript, often abbreviated as JS, is a high-level, interpreted programming language.It is a
        language which is also characterized as dynamic, weakly typed, prototype-based and
        multi-paradigm.Alongside
        HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web.
    </p>
    <p>JavaScript enables interactive web pages and thus is an essential part of web applications.
        The vast majority of websites use it, and all major web browsers have a dedicated JavaScript
        engine to execute it.As a multi-paradigm language, JavaScript supports event-driven, functional,
        and imperative (including object-oriented and prototype-based)programming styles.
    </p>
    <p>It has an API for working with text, arrays, dates, regular expressions, and basic
        manipulation of the DOM, but the language itself does not include any I/O, such as networking,
        storage, or graphics facilities, relying for these upon the host environment in which it is
        embedded.
    </p>
</div>
```

When the [**Format**] button is **clicked**, get the text inside the **textarea** with an **`id="input"`** and **format it**. The formatting is **done** as **follows:**

- Create a **new paragraph element** that holds no more than **3 sentences** from the given input**.**
- **Hint: Use interpolation string to create paragraph element. (`<p> {text} </p>`)**
- If the given input contains **less** or **3 sentences**, you need to create only 1 paragraph, fill it with these sentences and append this paragraph to the div with an **`id="output"`**.

Otherwise, when you have more than 3 sentences**,** create enough paragraphs to get all sentences from the **textarea.**

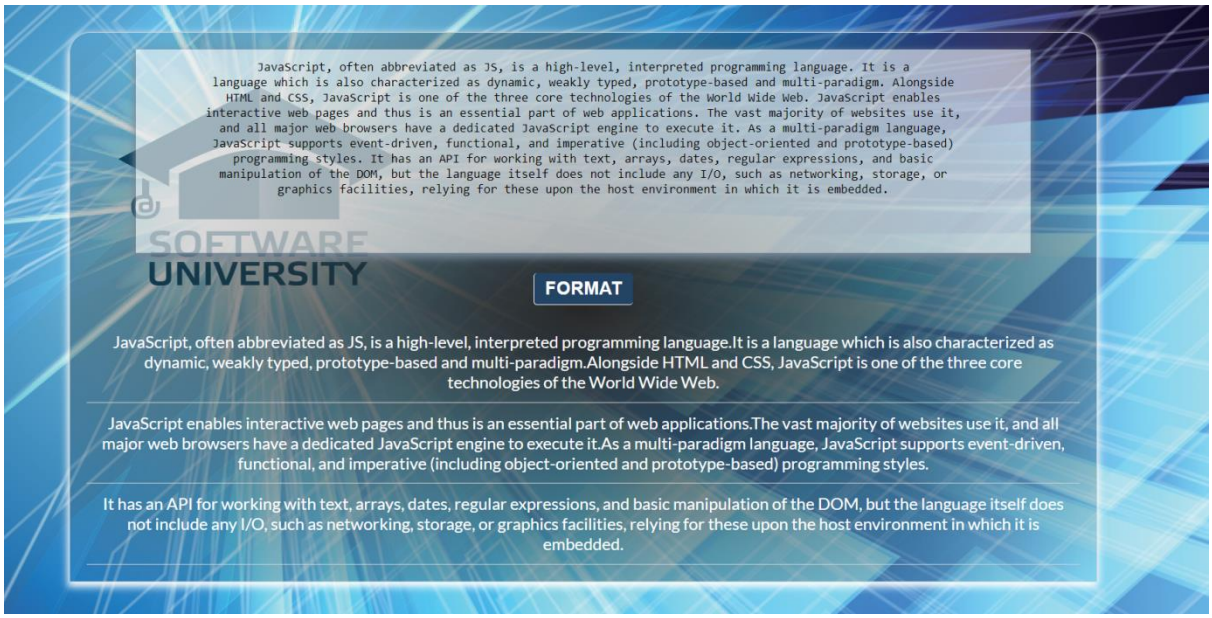
Just remember to **restrict** the **sentences** in **each paragraph to 3.**

Example:

- If the input textarea **contains 2 sentences**, create only **1 paragraph** with these 2 sentences

- If the input textarea **contains 7 sentences,** create **3 paragraphs**
  - The **first paragraph** must contain **the first 3 sentences**

- The **second paragraph** must contain **the other three sentences** of the whole text
- The **third paragraph** will contain **only the last sentence**



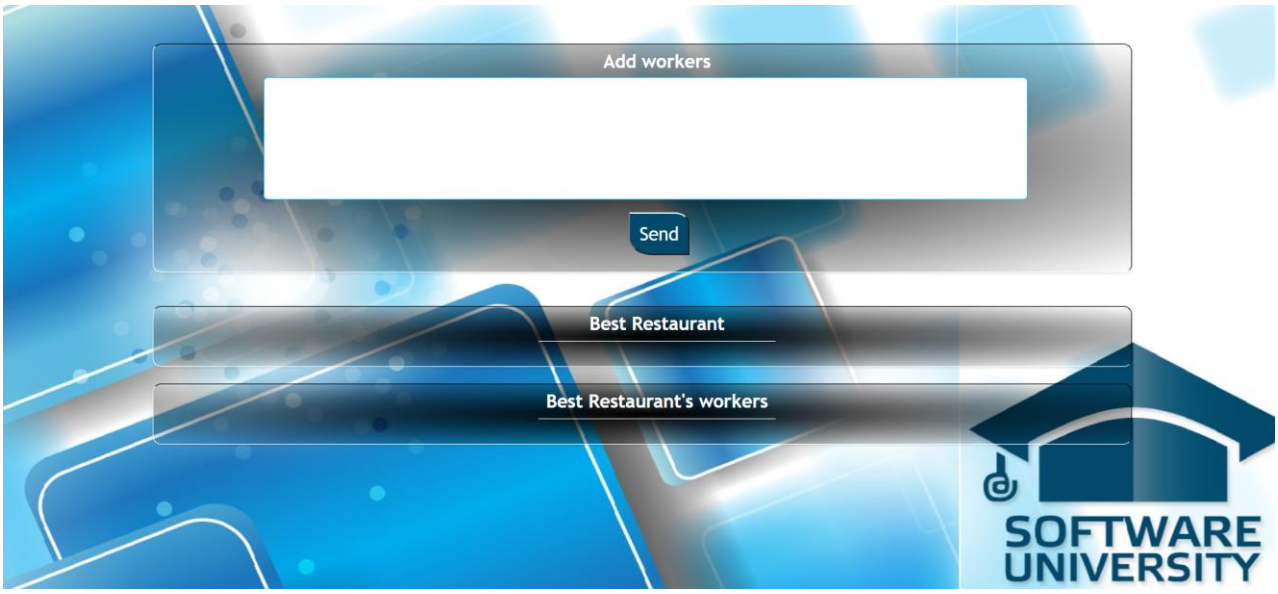
Output

Input	Output
JavaScript, often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.	JavaScript, often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.
JavaScript, often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm. Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it. As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has an API for working with text, arrays, dates, regular expressions, and basic manipulation of the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is	JavaScript, often abbreviated as JS, is a high-level, interpreted programming language.It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications.The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it.As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has an API for working with text,

embedded.	arrays, dates, regular expressions, and basic manipulation of the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.</p>

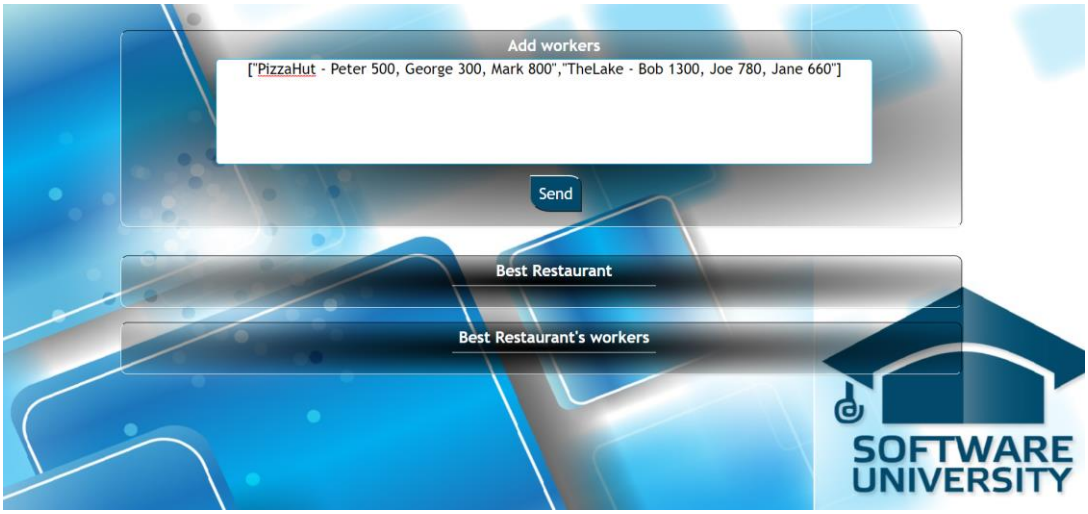
7. Hell's Kitchen

You will be given an **array of strings**, which represents a **list** of **all** the **restaurants** with their workers.



When the [**Send**] button is clicked:

- Display the **best restaurant** of all the **added restaurants** with its **average salary** and **best salary**.
- If there is a restaurant in the input array that is added more than once, you need to add new workers to the old ones and **update** the values of the **average salary** and the **best salary**.
- The best restaurant is the restaurant with the **highest average** salary. If two restaurants have the **same** average salary the best restaurant is the **first** one added.
- Display **all** workers in the **best restaurant** with their **salaries**.
  The best restaurant's workers should be **sorted** by their **salaries** in **descending** order.



Input

The input will be received from the given **textarea** in the form of an **array** of **strings**. Each string represents a **restaurant** with its **workers**: **["Mikes - Steve 1000, Ivan 200, Paul 800", "Fleet - Maria 850, Janet 650"]**

```
<div id="inputs">
  <h2>Add workers</h2>
  <textarea></textarea>
  <br>
  <button type="submit" id="btnSend">Send</button>
</div>
```

Output

- Output contains **two strings**
  - The first one is **the best restaurant** in the format:

    `Name: {restaurant name} Average Salary: {restaurant avgSalary} Best Salary: {restaurant bestSalary}`

    avgSalary and **bestSalary** must be formatted to the **second decimal point.**

- The second one is all the workers in that restaurant in the following format:

`Name: {worker name} With Salary: {worker salary} Name: {worker2 name} With Salary: {worker2 salary} Name: {worker3 name} With Salary: {worker3 salary}…`

Output strings must be set like **text content** in the following elements:

```
<div id="outputs">
    <div id="bestRestaurant">
        <h2>Best Restaurant</h2>
        <span></span>
        <p></p>
    </div>
    <div id="workers">
        <h2>Best Restaurant's workers</h2>
        <span></span>
        <p></p>
    </div>
</div>
```

Constraints

- The workers will be always **unique**

Examples

Input	Output	Comment
["PizzaHut - Peter 500, George 300, Mark 800", "TheLake - Bob 1300, Joe 780, Jane 660"]	Name: TheLake Average Salary: 913.33 Best Salary: 1300.00 Name: Bob With Salary: 1300 Name: Joe With Salary: 780 Name: Jane With Salary: 660	The added restaurants are: TheLake and PizzaHut. TheLake has average salary: (1300+780+660)/3= 913.33, and PizzaHub has average salary: (500+300+800)/2=533.33. So the best restaurant is TheLake.
["Mikes - Steve 1000, Ivan 200, Paul 800","Fleet - Maria 850, Janet 650"]	Name: Fleet Average Salary: 750.00 Best Salary: 850.00 Name: Maria With Salary: 850 Name: Janet With Salary: 650	

8. Generate Report

You will be given a **web page**, containing a **table** and **output area**.

Employee	Department	Status	Date Hired	Benefits	Compensation	Rating
Poole, Tracy	Facilities/Engineering	Full Time	15.7.2019	R	71 670	4
Ramos, Jan	Human Resources	Full Time	17.6.2017	DMR	66 740	2
Jennings, Gary	Account Management	Full Time	4.8.2009	DM	45 100	2
Ortega, Jeffrey	Quality Control	Contract	20.3.2018		26 020	5
Shields, Robert	Product Development	Contract	23.11.2016		45 830	4
Gregory, Jon	Human Resources	Full Time	6.5.2017	R	79 150	2
Sheppard, Curtis	Quality Control	Full Time	15.3.2006	D	61 850	2
Williamson, Sumed	Manufacturing	Contract	10.2.2018		57 110	3
Moreno, Chris	Quality Assurance	Full Time	29.9.2015	R	72 060	2
Munoz, Michael	Quality Assurance	Full Time	17.3.2010	DMR	29 210	5
Kirby, Michael	Account Management	Half-Time	22.11.2005	R	22 475	4
Jenkins, Scott	Account Management	Full Time	16.8.2016	DMR	54 190	4
Ross, Janice	Marketing	Half-Time	25.4.2006	R	26 790	2
Kelley, Nancy	Quality Control	Contract	20.10.2013		64 263	3
Blackwell, Brandon	Quality Control	Contract	29.10.2005		58 250	2
Bowers, Tammy	Sales	Half-Time	20.12.2016	DMR	49 405	4
Fleming, Irv	Environmental Compliance	Half-Time	7.2.2013	DMR	11 025	1
Skinner, Jason	IT	Full Time	15.2.2014	R	73 030	5
Wade, Kevin	Green Building	Full Time	29.7.2014	DMR	71 120	4
Barrett, John	Quality Control	Full Time	20.10.2006	R	35 460	1

Generate Report

When the **"Generate Report"** button is **pressed**:

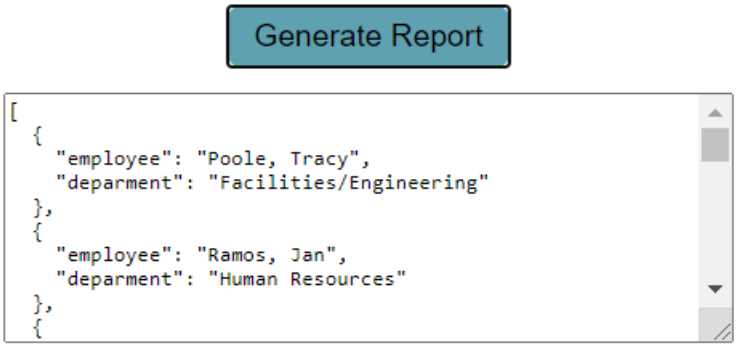
- You must generate a **JSON report** from the data inside the table, by **only taking the columns**, which are **selected**.

Each table header has a **checkbox**. If the checkbox is **checked**, then the data from this column must be included in the **report**. **Unchecked** columns must be **omitted**.

```
<th>
  "Employee "
  <input type="checkbox" name="employee">
</th>
```

For **every row** (excluding the header):

- Create an **object** with **properties for each** of its columns.
- The name of each property is the name attribute of the column's header, and the value is the text content of the cell.
- Store the result in an array and output it as a JSON string display it inside the **<textarea>** with **id** "**output**". See the example for details.



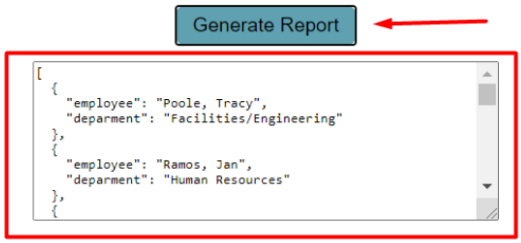
Input/Output

There will be input, your program must execute based on the page content. The output must be a **JSON string**, displayed in the **<textarea>** with **id** "**output**".

```
<div>
  <textarea id="output"></textarea>
</div>
```

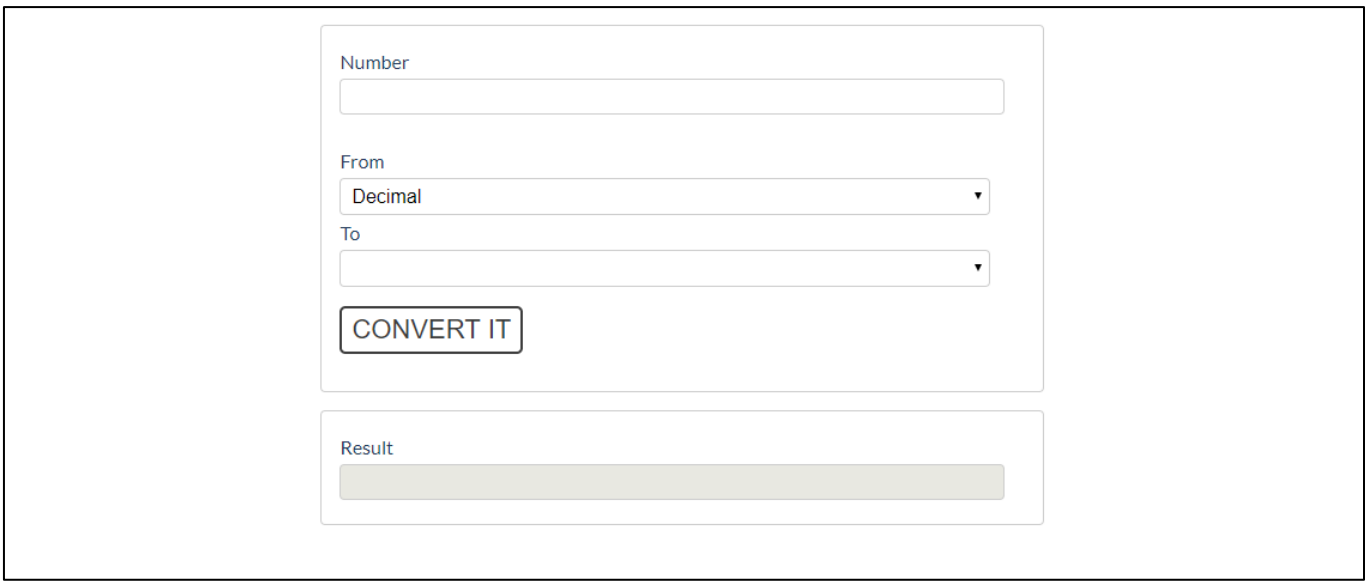
Example

Employee	Department	Status	Date Hired	Benefits	Compensation	Rating
Poole, Tracy	Facilities/Engineering	Full Time	15.7.2019	R	71 670	4
Ramos, Jan	Human Resources	Full Time	17.6.2017	DMR	66 740	2
Jennings, Gary	Account Management	Full Time	4.8.2009	DM	45 100	2
Ortega, Jeffrey	Quality Control	Contract	20.3.2018		26 020	5
Shields, Robert	Product Development	Contract	23.11.2016		45 830	4
Gregory, Jon	Human Resources	Full Time	6.5.2017	R	79 150	2
Sheppard, Curtis	Quality Control	Full Time	15.3.2006	D	61 850	2
Williamson, Sumed	Manufacturing	Contract	10.2.2018		57 110	3
Moreno, Chris	Quality Assurance	Full Time	29.9.2015	R	72 060	2
Munoz, Michael	Quality Assurance	Full Time	17.3.2010	DMR	29 210	5
Kirby, Michael	Account Management	Half-Time	22.11.2005	R	22 475	4
Jenkins, Scott	Account Management	Full Time	16.8.2016	DMR	54 190	4
Ross, Janice	Marketing	Half-Time	25.4.2006	R	26 790	2
Kelley, Nancy	Quality Control	Contract	20.10.2013		64 263	3
Blackwell, Brandon	Quality Control	Contract	29.10.2005		58 250	2
Bowers, Tammy	Sales	Half-Time	20.12.2016	DMR	49 405	4
Fleming, Irv	Environmental Compliance	Half-Time	7.2.2013	DMR	11 025	1
Skinner, Jason	IT	Full Time	15.2.2014	R	73 030	5
Wade, Kevin	Green Building	Full Time	29.7.2014	DMR	71 120	4
Barrett, John	Quality Control	Full Time	20.10.2006	R	35 460	1



9. *Number Convertor

Write a function that **converts** a **decimal number** to **binary** and **hexadecimal**.



The given number will always be in **decimal format.** The "**From**" select menu will only have a **Decimal** option,

but the "**To**" select menu will have **two options**: **Binary** and **Hexadecimal**.

This means that our program should have the functionality to **convert decimal** to **binary** and **decimal** to **hexadecimal**. When you convert to **hexadecimal** it must be **upper case**.

Note that "**To" select menu** by default is empty. You have to insert the two options (**`'Binary'`** and **`'Hexadecimal'`**) inside before continue. Also, they should have **values** ('**`binary`**' and '**`hexadecimal`**').

- When the [**`Convert it`**] button is **clicked**, the expected result should appear in the [**`Result`**] input field.