DOM and Events - Exercises

Problems for in-class lab for the "JS Front-End" course @ SoftUni. Submit your solutions in the SoftUni judge system at https://judge.softuni.org/Contests/3795/DOM-and-Events-Exercises

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()`
- `replaceWith()`
- `replaceAll()`
- `closest()`
- `replaceChildren()`
- 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

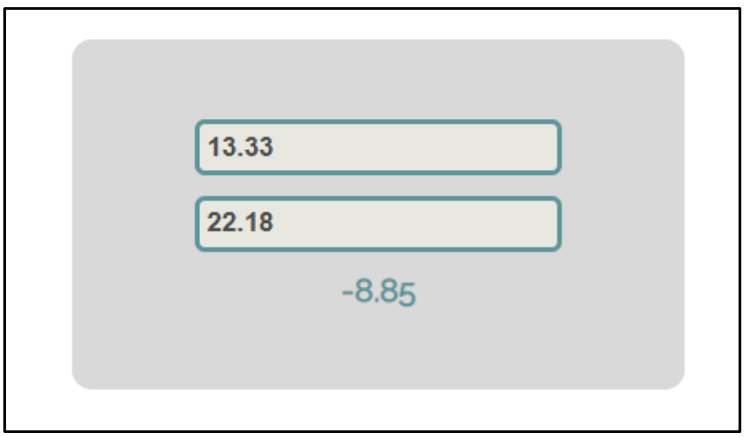
An HTML page holds **two text fields** with ids **"firstNumber"** and **"secondNumber"**. Write a function to **subtract** the values from these text fields and display the result in the **div** named **"result"**.

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, the type is **text**. To get a **number**, we need to use a function to **parse them**.

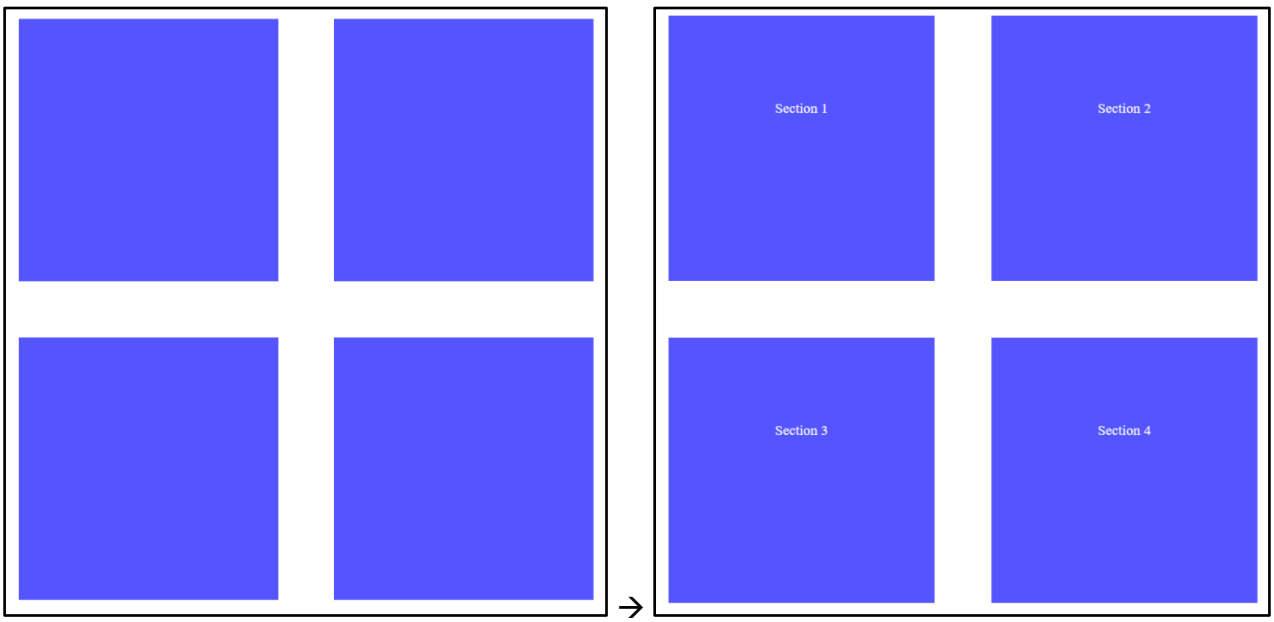
```
let num1 = document.getElementById('firstNumber').value;
let num2 = document.getElementById('secondNumber').value;
```

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

2. Sections

You will receive an **array** of strings. For each string, create a **div** with a **paragraph** with the **string** in it. Each paragraph is initially **hidden (display:none)**. Add a **click event listener** to **each div** that **displays** the **hidden** paragraph. Finally, you should **append** all divs to the element with an **`id "content"`**.

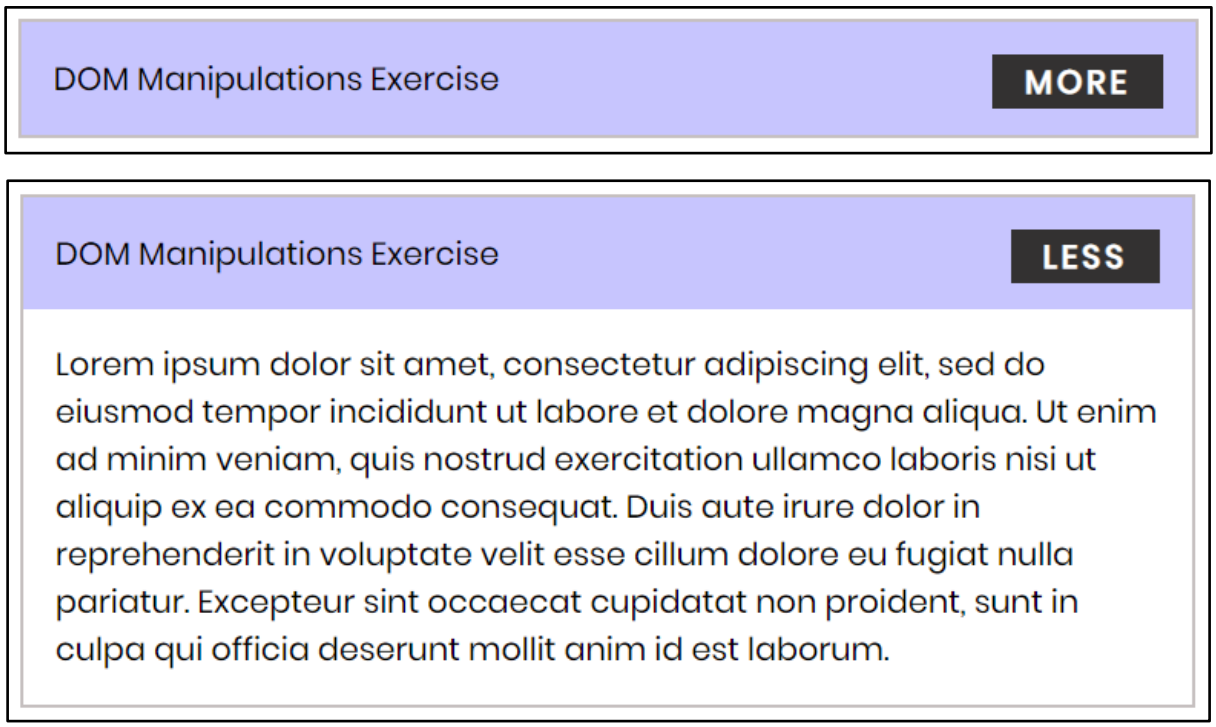
Example



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

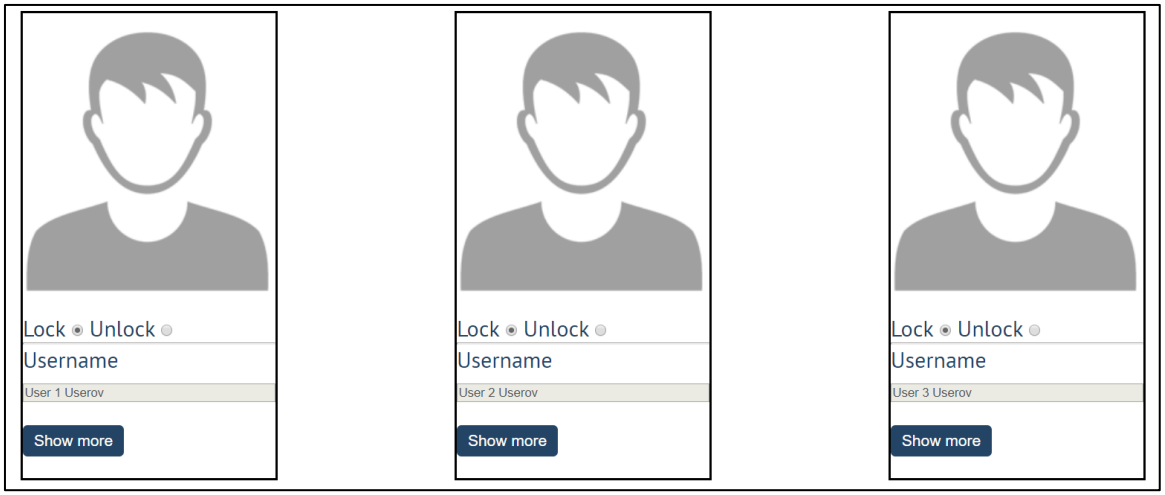


Hints

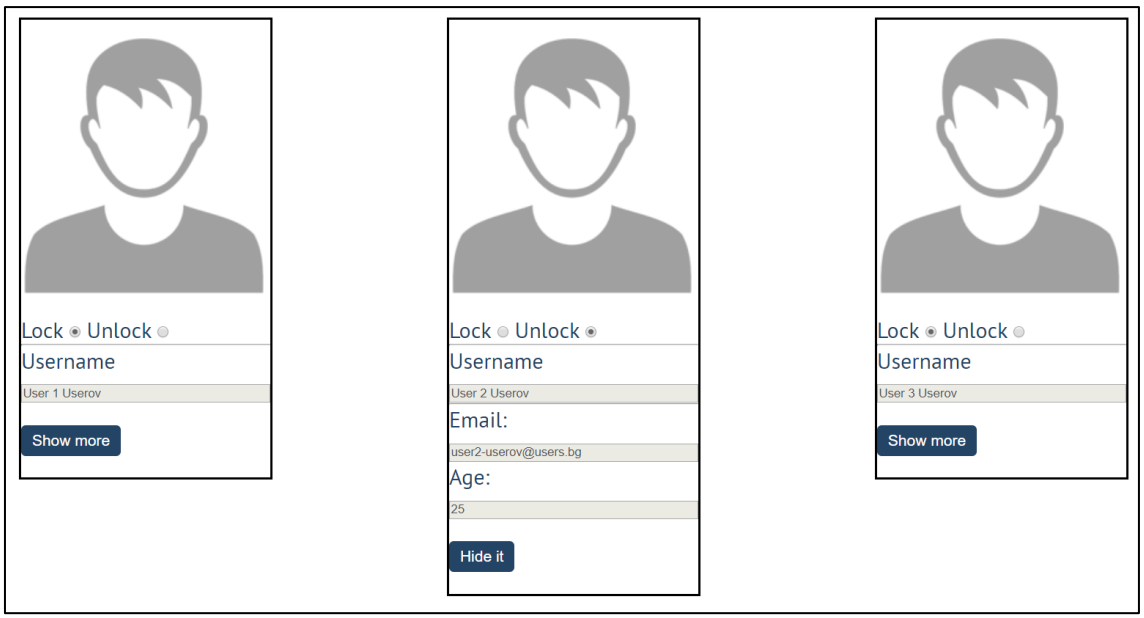
- To **change** the text content of a button, you could use **`getElementsByClassName`**. However, that returns a **collection** and we need only **one** element from it, so the correct way is to **use** **`getElementsByClassName("button")[0]`** as it will return the needed span element.
- After that, 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. Locked Profile

In this problem, you should **create a JS functionality** that **shows** and **hides** the additional information about users.



When one of the [**Show more**] **buttons** is clicked, the **hidden information** inside the div should be shown, only if **the profile is not locked**! If the current profile is **locked,** nothing should happen.



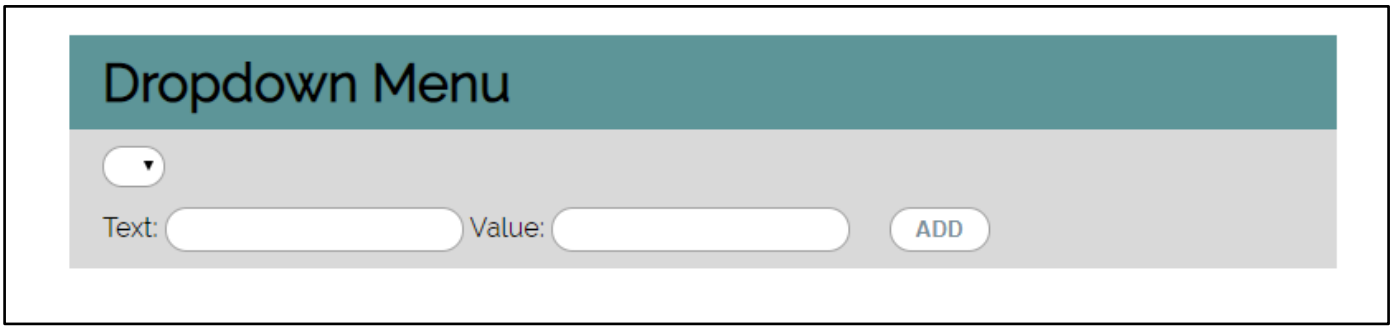
If the **hidden information is displayed** and we **lock the profile again**, the [**`Hide it`**] button should **not be working**!

Otherwise, when the profile is **unlocked** and we click on the [**`Hide it`**] button, the new fields must hide again.

5. Fill Dropdown

Your task is to take values from **input** fields with **ids "newItemText"** and **"newItemValue"**. Then you should create and append an **`<option>`** to the **`<select>`** with **id "menu".**

Example

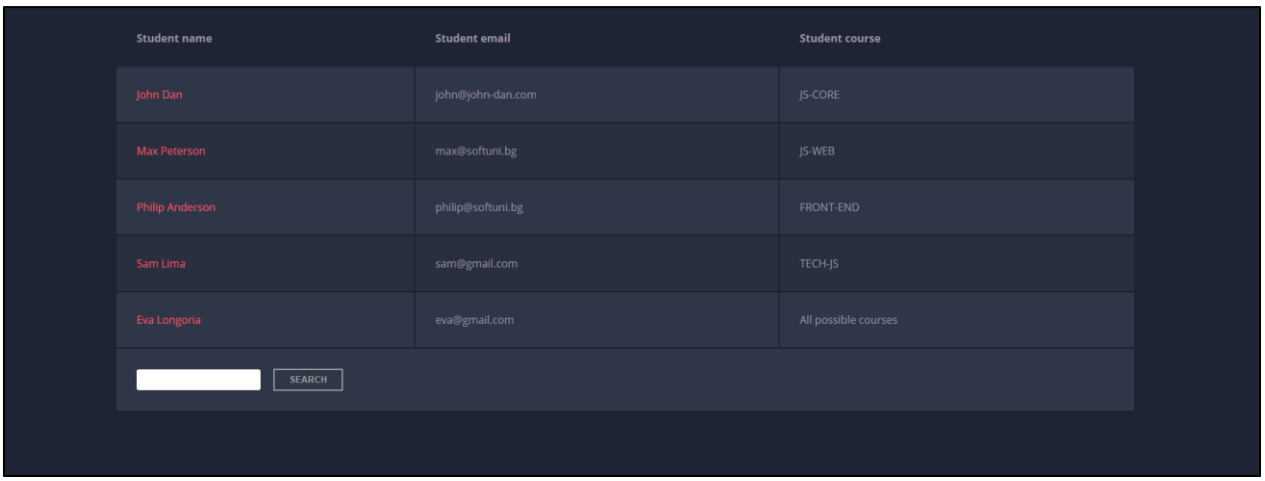


Hints

- Your function should take the values of **newItemText** and **newItemValue**. After that, you should create a new **option** element and set its **textContent** and its **value** to the newly taken ones.
- Once you have done all of that, you should **append** the newly created **option** as a **child** to the **select** item with id **"menu".**

6. Table - Search Engine

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



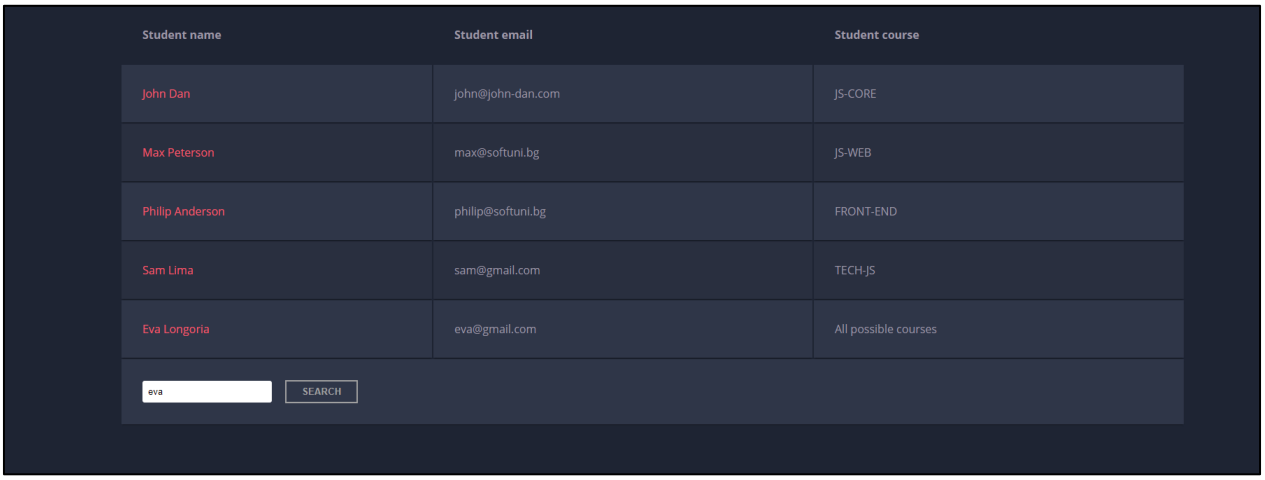
When the **"`Search`" button** is **clicked**, go through all cells in the table except for the first row (Student name, Student email, and Student course) and check if the given input has a match (check for both **full words** and **single letters**).

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.

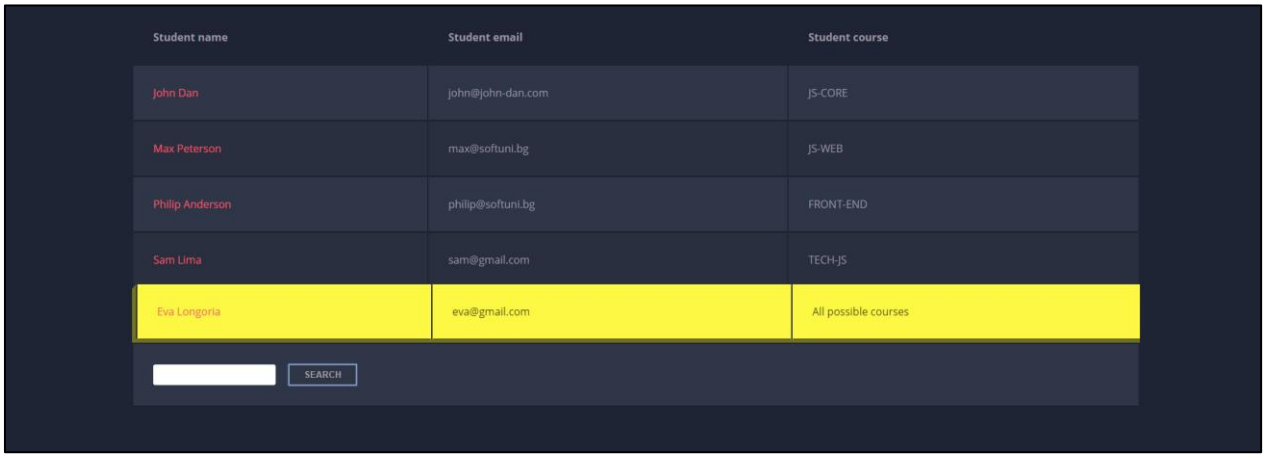
Otherwise, if there is no match, **<u>nothing should happen</u>**.

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**.

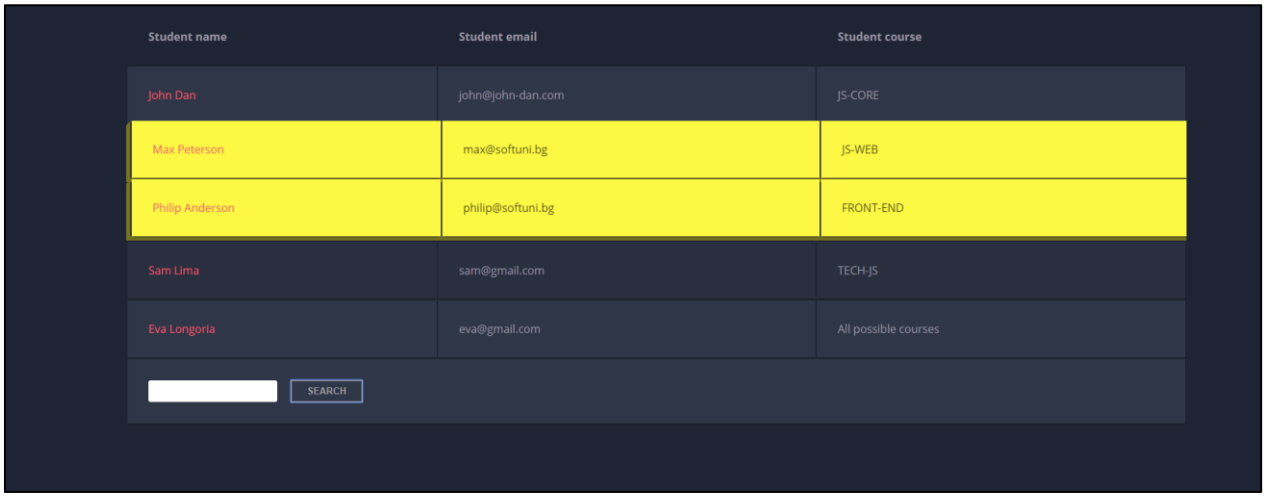
For instance, if we try to find **eva:**



The result should be:



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



7. 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.**

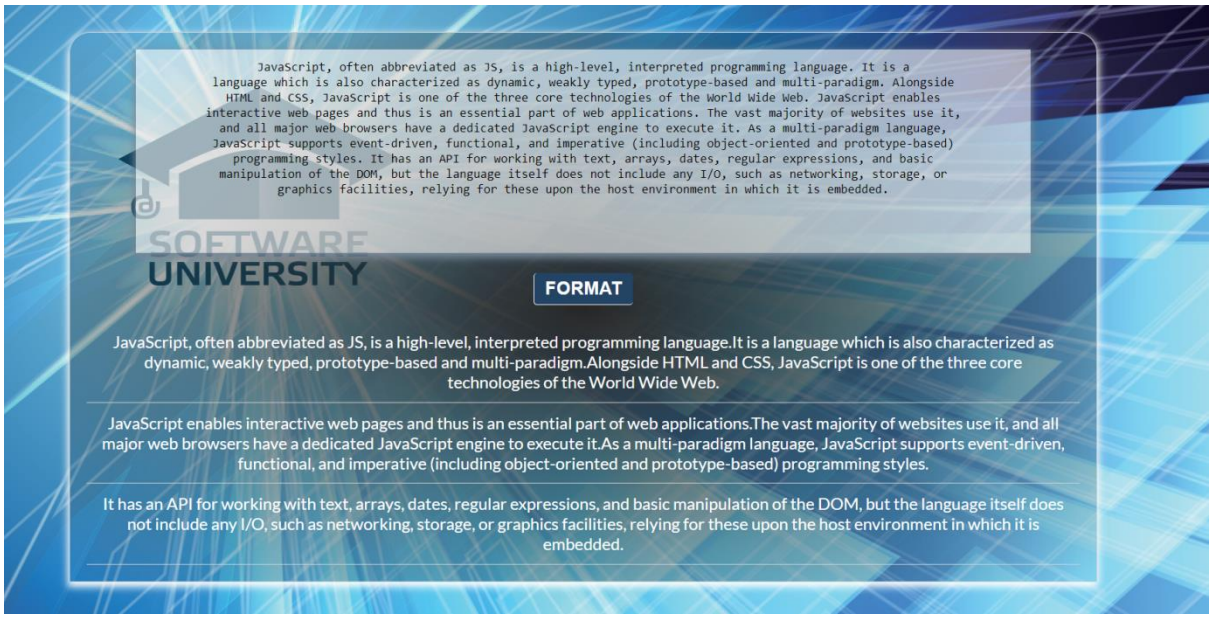
Just remember to **restrict** the **sentences** in **each paragraph to 3.**

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>

8. Furniture

You will be given some furniture as an **array of objects**. Each object will have a **name**, a **price,** and a **decoration factor**.

When the [**"Generate"**] **button is clicked**, add a **new row to the table** for each piece of furniture with **image, name, price,** and **decoration factor** (code example below).

When the [**"Buy"**] button is clicked, get all **checkboxes that are marked** and show in the **result textbox** the **names** of the piece of furniture that **were checked**, separated by a **comma** and **single space** (**", "**) in the following format: **"Bought furniture: {furniture1}, {furniture2}…"**.

On the next line, print the total price in the format: **"Total price: {totalPrice}"** (formatted to the second decimal point). Finally, print the average decoration factor in the format: **"Average decoration factor: {decFactor}"**

Input Example

```
[{"name": "Sofa", "img":
"https://res.cloudinary.com/maisonsdumonde/image/upload/q_auto,f_auto/w_200/img/
grey-3-seater-sofa-bed-200-13-0-175521_9.jpg", "price": 150, "decFactor": 1.2}]
```

Examples