DOM and Events – More 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/3796/DOM-and-Events-More-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. Edit Element

Create function **edit()** that takes **three** parameters.

Input/Output

The first parameter is a **reference** to an **HTML** element, the other two parameters are string–**match** and **replacer.**

You have to **replace** all occurrences of the **match** inside the **text content** of the given element with a **replacer.**

Examples

2. Extract Parenthesis

Write a JS function that when **executed**, extracts all parenthesized text from a target paragraph by given element ID. The result is a string, joined by "; " (semicolon, space).

Input

Your function will receive a **string parameter**, representing the target element ID, from which text must be extracted. The text should be extracted from the DOM.

Output

Return a string with all matched text, separated by "; " (semicolon, space).

Examples

```
•••<!DOCTYPE html> == $0
  <html lang="en">
  ▶<head>…</head>
  ▼<body>
    ▼<p id="content">
      "
          The Rose Valley (Bulgaria) is located just south of the Balkan Mountains
      (Kazanlak).The most common oil-bearing rose found in the valley is the pink-
      petaled Damask rose (Rosa damascena Mill).
      "
     </p>
    </body>
  </html>
```

Sample call
`let text = extract("content");`
Result (stored in variable text)
`Bulgaria; Kazanlak; Rosa demascena Mill`

3. Mouse Gradient

Write a program that **detects** and **displays** how far along a gradient the user has **moved** their **mouse**. The result should be **rounded down** and displayed as a **percentage** inside the **`<div>`** with id "**`result`**".

Submit **only** the **`attachGradientEvents()`** function in Judge.

Input/Output

There will be no input/output, your program should instead **modify** the DOM of the given HTML document.

Examples

4. Dynamic Validation

Write a **`function`** that **dynamically validates** an **email** input field when it is **changed**. If the input is **invalid**, apply the class "**`error`**". Do **not** validate on every keystroke, as it is annoying for the user, consider only **change** events.

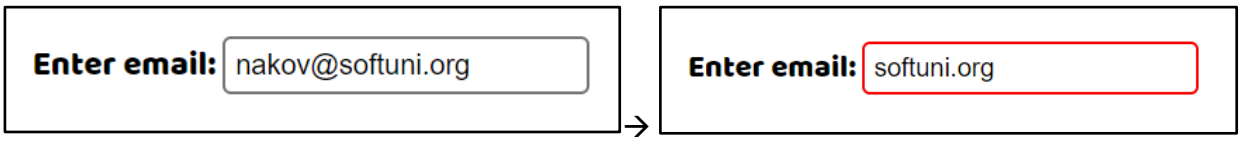
A valid email is considered to be in the format: **`<name>@<domain>.<extension>`**

Only **lowercase Latin characters** are allowed for any of the parts of the email. If the input is valid, **clear** the style. Submit **only** the **`validate()`** function in Judge.

Input/Output

There will be no input/output, your program should instead **modify** the DOM of the given HTML document.

Example



5. Shopping Cart

You will be given some products that you should be able to add to your cart. Each product will have a **`name`**, **`picture`**, and **`price`**.

When the **`"Add"`** button is clicked, append the current product to the **`textarea`** in the following format: **`"Added {name} for {money} to the cart.\n"`**. The price must be fixed to the second digit.

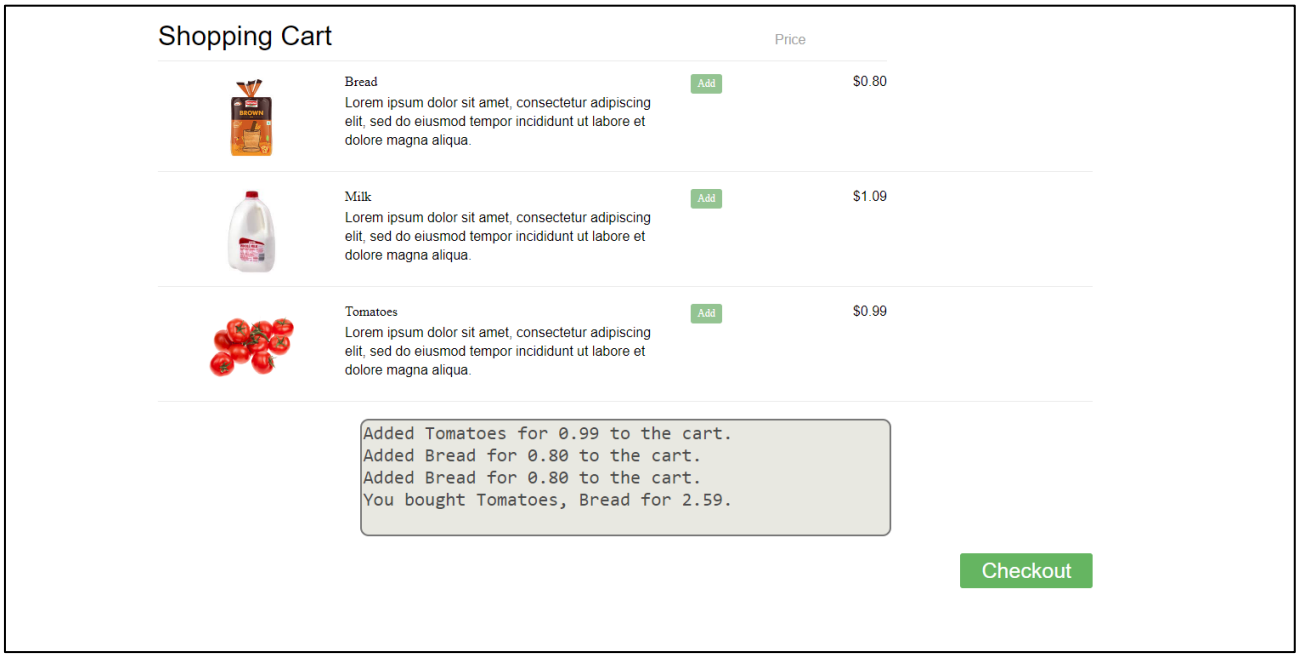
When the button **`"Checkout"`** is clicked, calculate the **total money** that you need to pay for the products that are currently in your cart. Append the result to the **`textarea`** in the following format:

`"You bought {list} for {totalPrice}."`

The list should contain only the **unique products**, separated by **`", "`**. The total price should be rounded to the second decimal point.

Also, after clicking over "**`Checkout`**" and every from above is done you should **disable all buttons**. (You **can't** add products or checkout again if once the checkout button is clicked).

Examples



6. 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

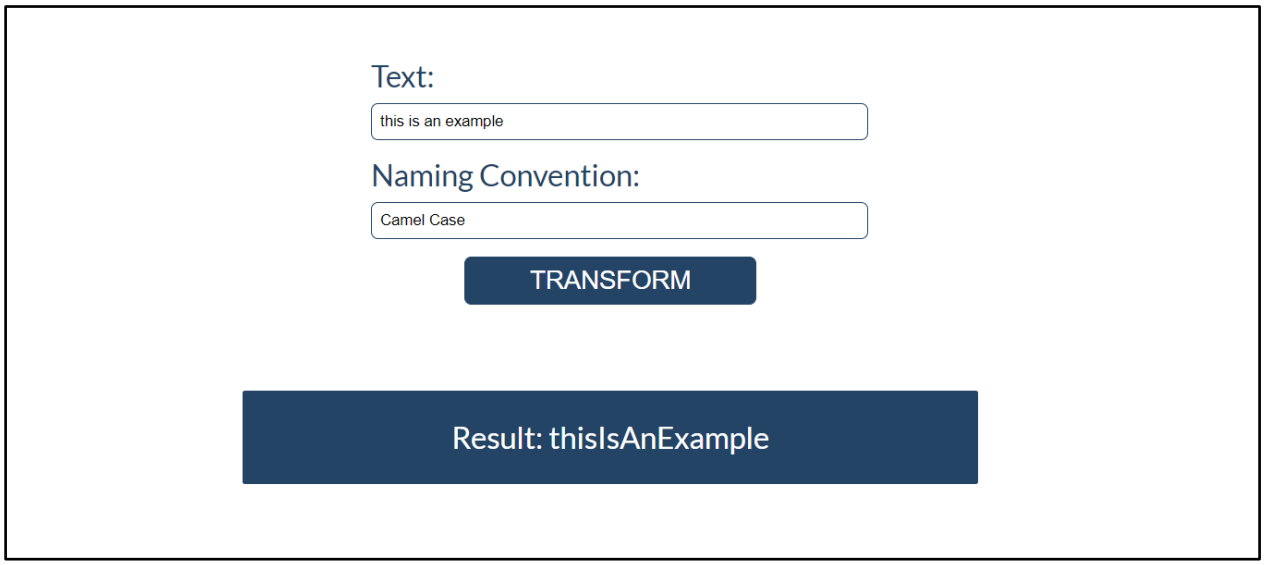
First, take the two values from the input fields:

```
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`**



7. 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` ``**.

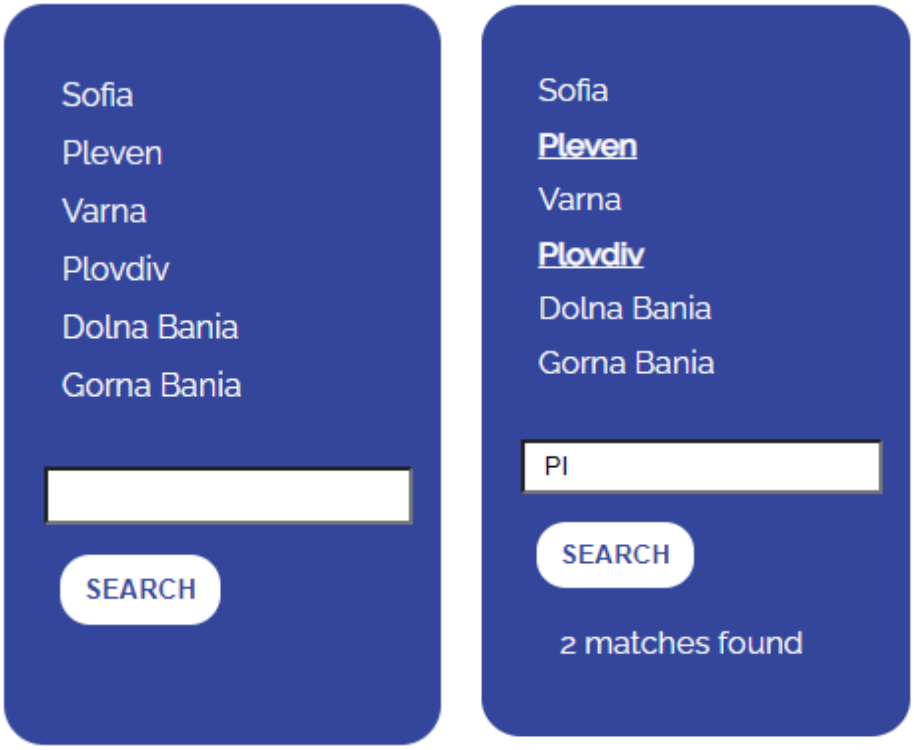
Note: It is necessary to clear the results of the previous search.

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

search.js

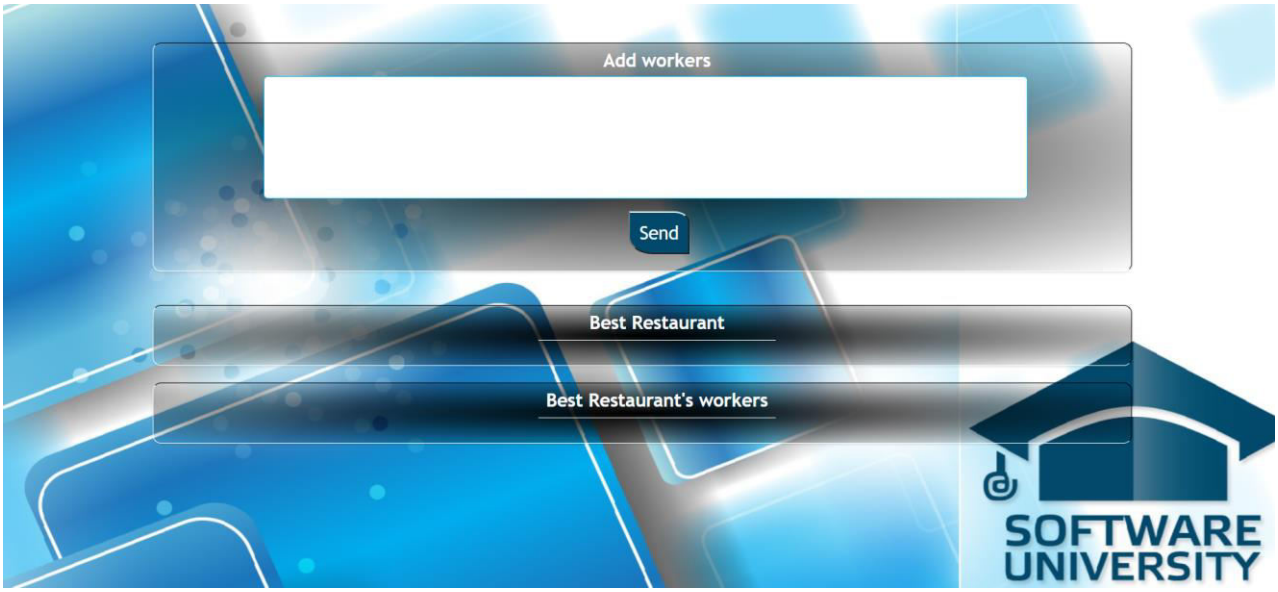
```
function search() {
    // TODO
}
```

Screenshots



8. Hell's Kitchen

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



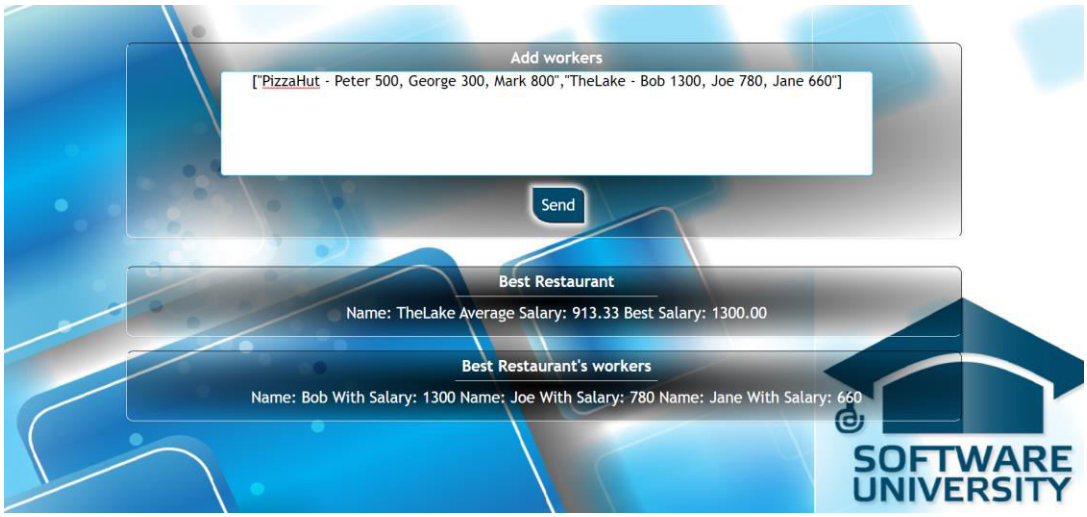
When the [**Send**] button is clicked:

- 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

- The 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:

```html
<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	

9. 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.

Generate Report


```
[
  {
    "employee": "Poole, Tracy",
    "deparment": "Facilities/Engineering"
  },
  {
    "employee": "Ramos, Jan",
    "deparment": "Human Resources"
  },
  {
```


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

Generate Report


```
[
  {
    "employee": "Poole, Tracy",
    "deparment": "Facilities/Engineering"
  },
  {
    "employee": "Ramos, Jan",
    "deparment": "Human Resources"
  },
  {
```


10. Number Convertor

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

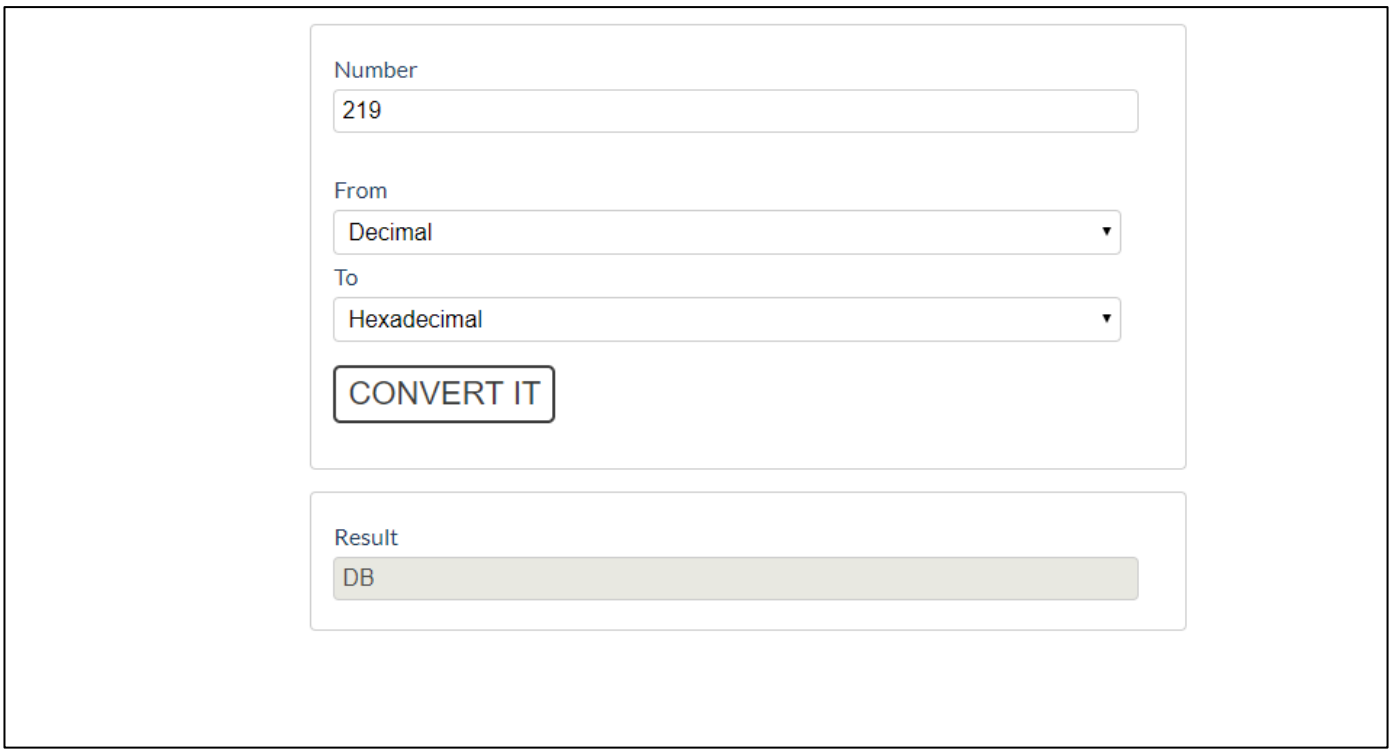
The given number will always be in **decimal format.** The "**From**" select menu will only have a **Decimal** option,

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

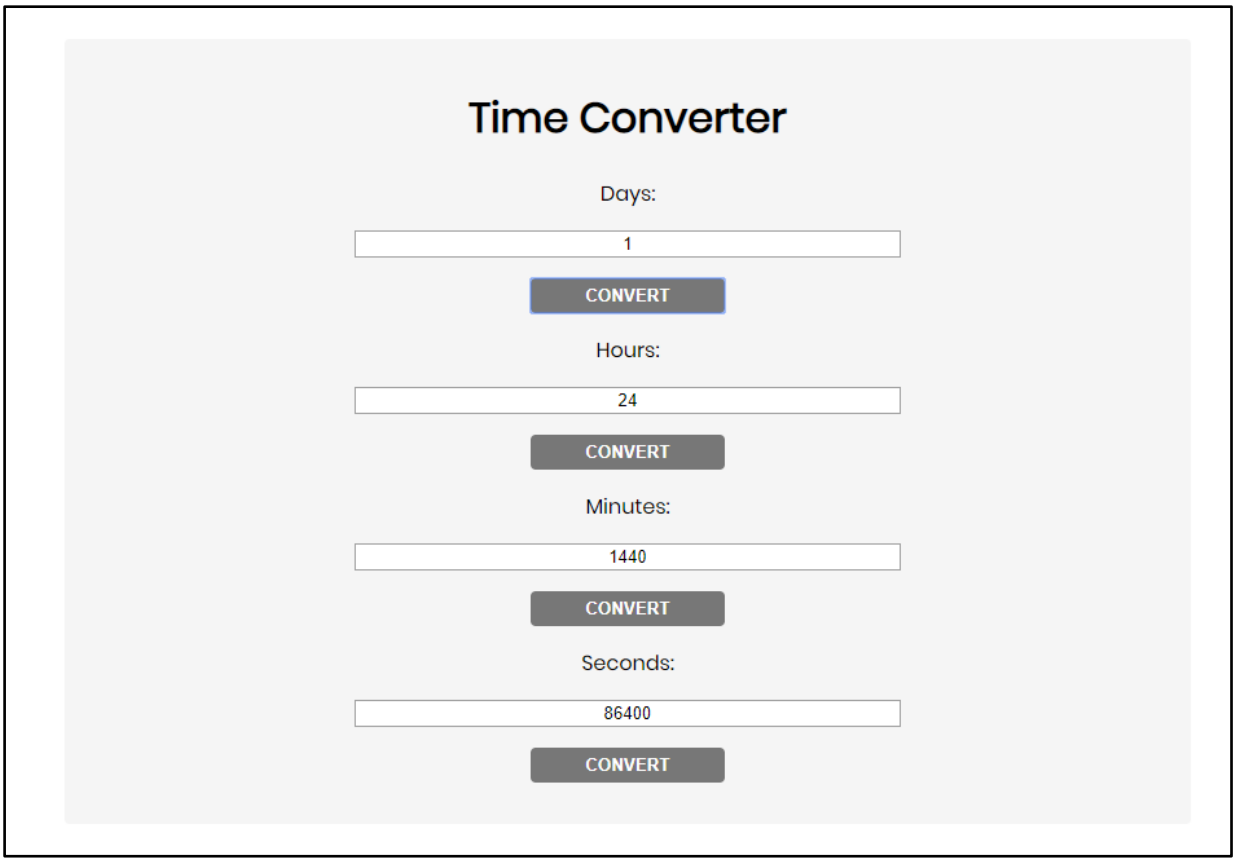
- When the [**`Convert it`**] button is **clicked**, the expected result should appear in the [**`Result`**] input field.

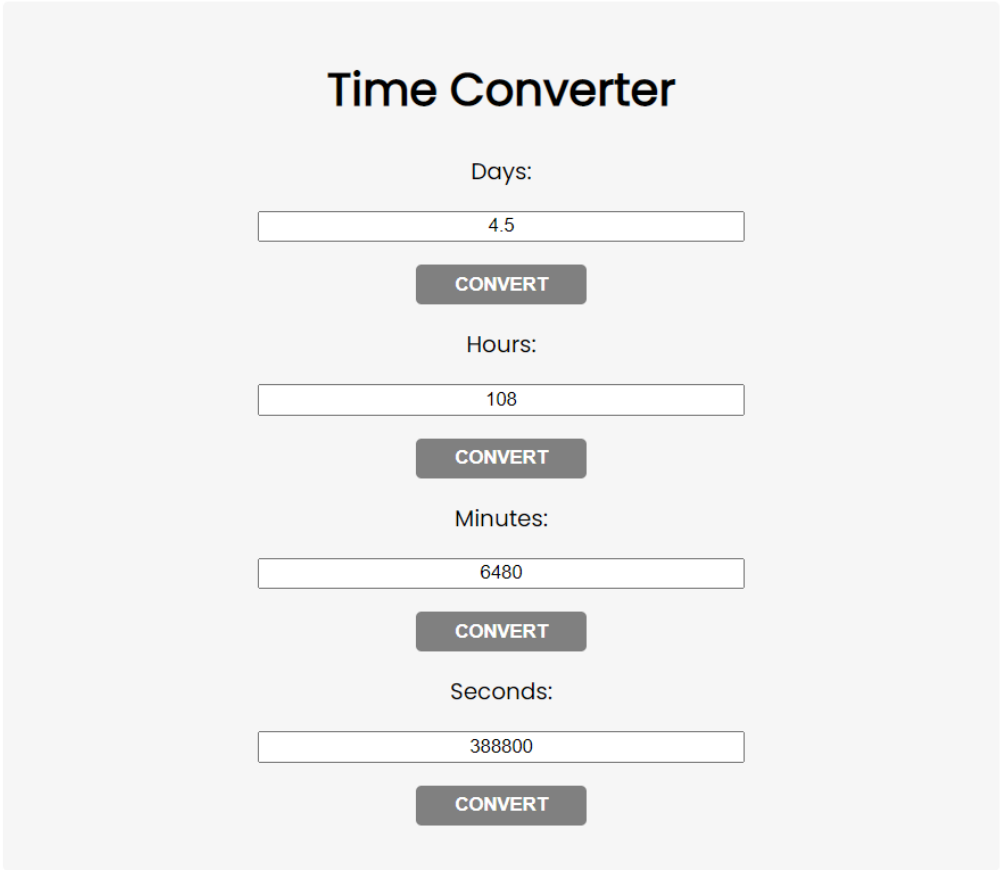


11. Time Converter

Create a program that **converts** different time units. Your task is to add a **click** event listener to **all** [**CONVERT**] **buttons**. When a button is **clicked**, read the **corresponding** input field, **convert** the value to the **three other** time units and **display** it in the input fields.

Example

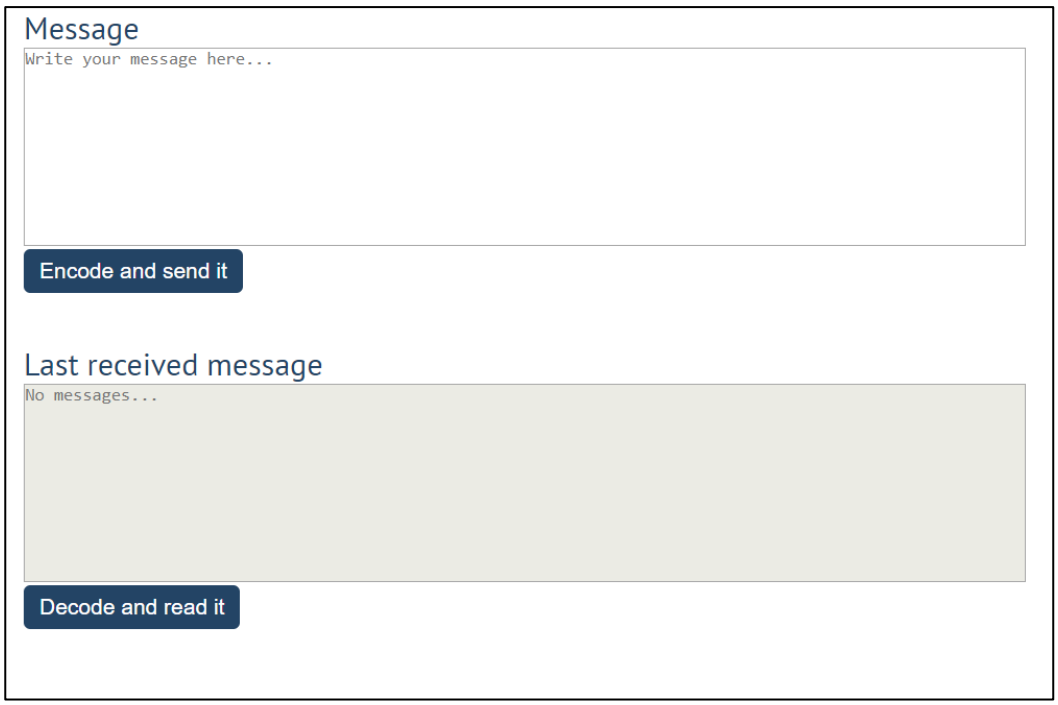




One day is equal to 24 hours/1440 minutes/86400 seconds. Whichever button we **click,** the input fields should **change** depending on the added value on the left. (For example, if we write 48 hours and click convert the days, the field value should change to 2).

12. Encode and Decode Messages

In this problem, you should **create a JS functionality** that **encodes and decodes some messages which travel to the network.**



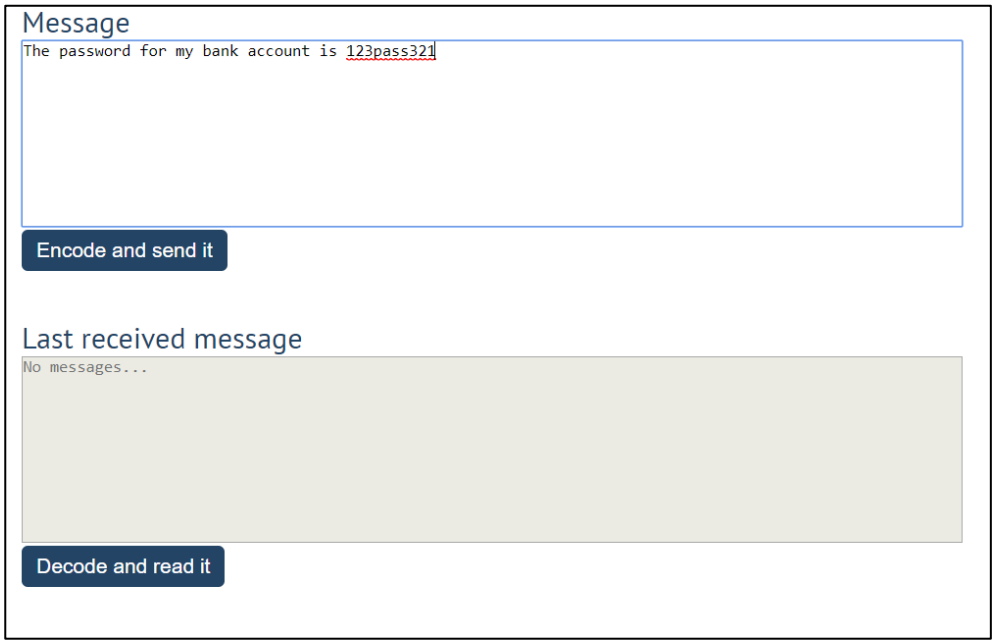
This program should contain **two functionalities**.

The first one is to **encode the given message** and **send it** to the **receiver**.

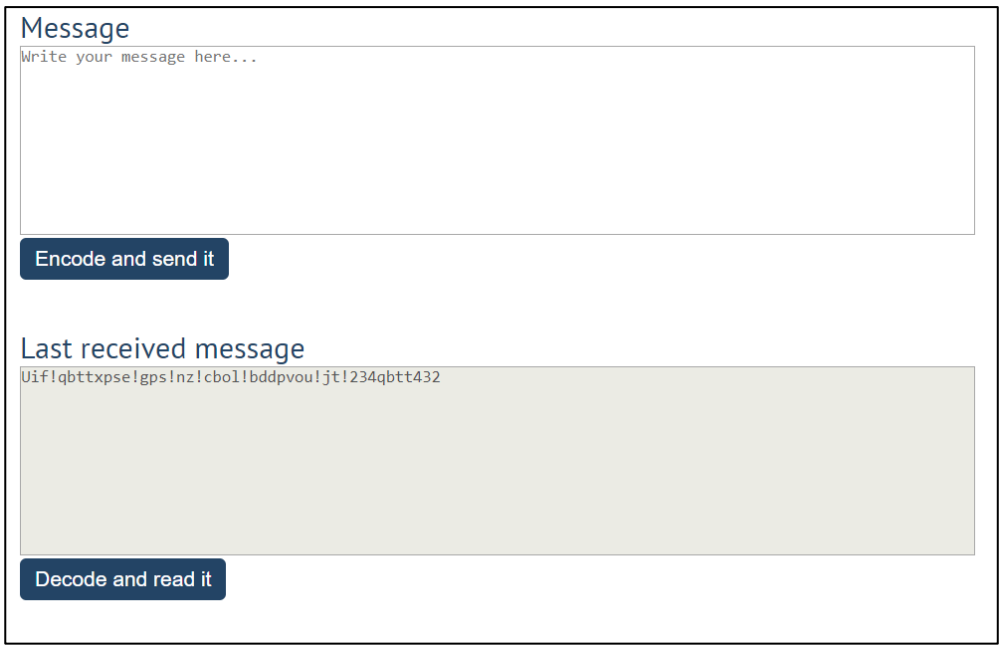
The second one is to **decode the received message** and **read it (display it)**.

When the [**`Encode and send it`**] **button** is clicked, you should get the given message from the first **textarea**. When you get the current message, you should encode it as follows:

- **Change** the **ASCII CODE** on **every single character** in that message when you **add 1** to the current **ASCII NUMBER**, that represents the current character in that message
- **Clear** the **sender textarea** and **add** the encoded message to the **receiver textarea**

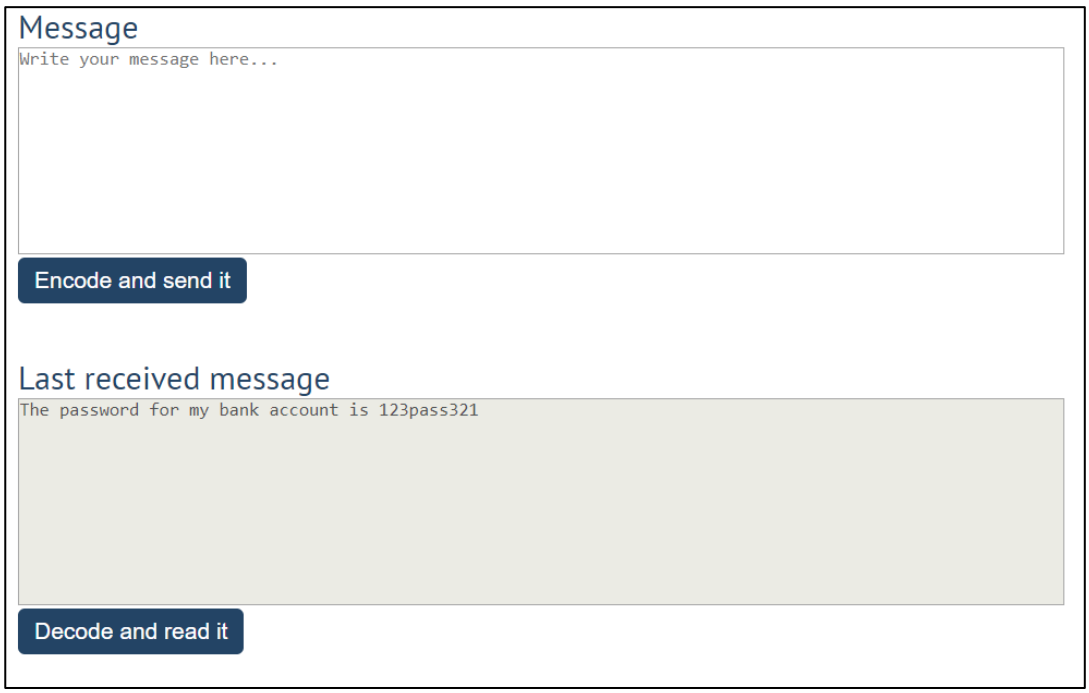


<u>After clicking the [**`Encode and send it`**] **button** the result should be:</u>



After that, when the [**`Decode and read it`**] **button** is clicked. You need to get the **encoded message** from **the receiver textarea** and do the **opposite logic** from encoding:

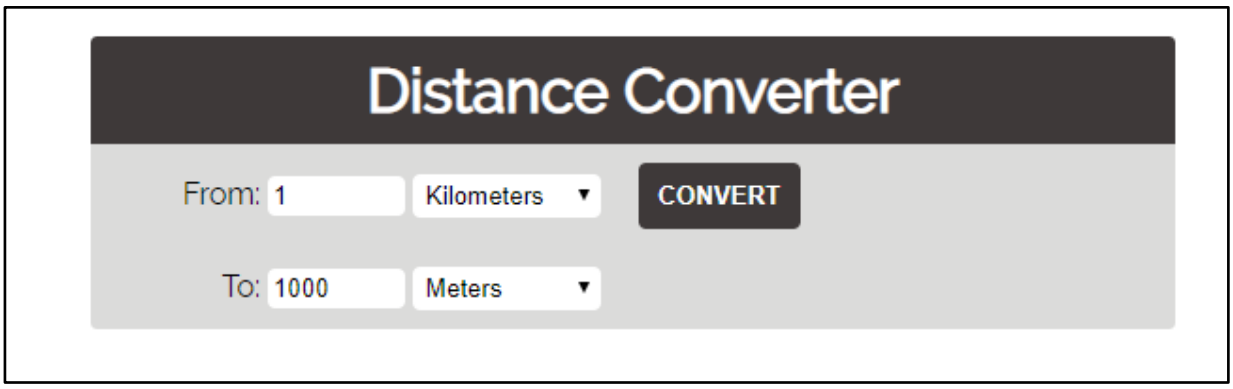
- **Subtract 1** from the current **ASCII NUMBER**, that represents the current character in that message
- Replace the **encoded message** with the already **decoded message** in the receiver **textarea**, to make it readable



13. Distance Converter

Your task is to convert from **one** distance unit to **another** by adding a **click** event listener to a button. When it is clicked, **read** the value from the input field and **get** the **selected** option from the **input** and **output** units dropdowns. Then **calculate** and **display** the converted value in the **disabled** output field.

Example



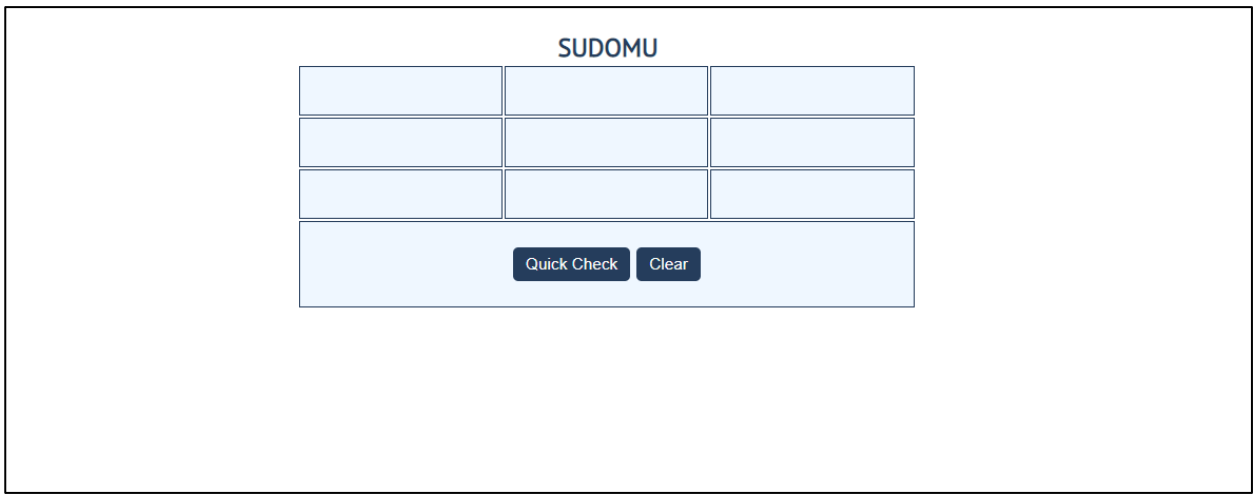
Hints

- Multiply the incoming distance by the following conversion rates to convert to meter
- Divide to convert from meters to the required output unit
- To see which option is selected, read the properties of its parent: **value** gives you the value of the selected option (as displayed in the HTML), **selectedIndex** gives you the 0-based index of the selected option. For example, if miles are selected, **inputUnits.value** is "**mi**", **inputUnits.selectedIndex** is **4**. Option text is irrelevant
- Use the following table information to do that:

1 km	1000 m
1 m	1 m
1 cm	0.01 m
1 mm	0.001 m
1 mi	1609.34 m
1 yrd	0.9144 m
1 ft	0.3048 m
1 in	0.0254 m

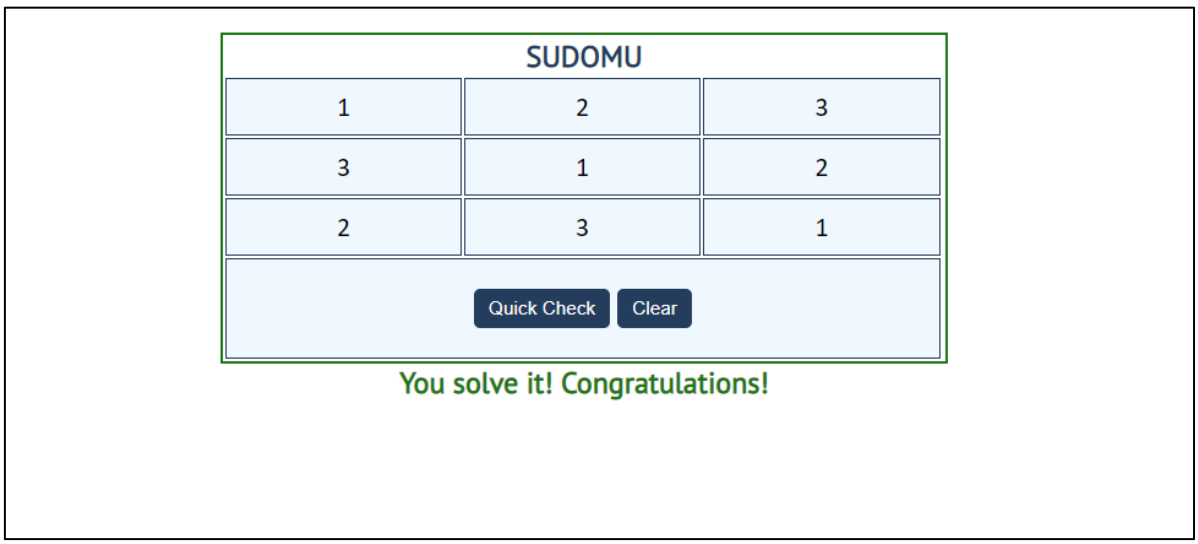
14. Sudomu

Write a function that implements **SUDOMU** (**Sudoku inside the DOM**).



The rules are simple and they are **the same** as the **typical sudoku game** (for more information, click here).

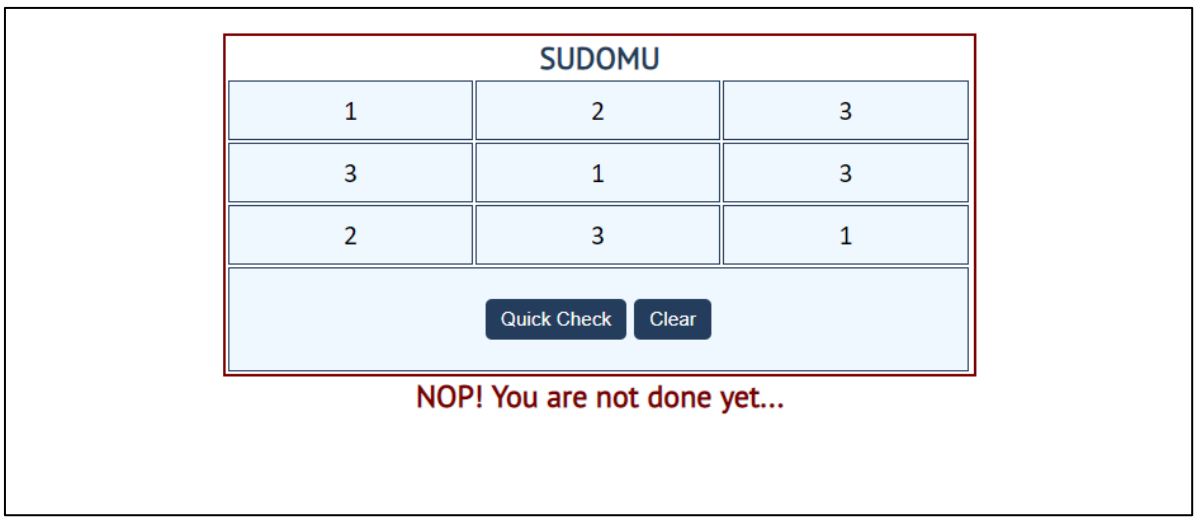
If the table is filled with the **right numbers**, and the ["**`Quick Check`**"] button is **clicked**, the expected result should be:



The table borer should be changed to: "**`2px solid green`**". The **`text content`** of the **`paragraph`** inside the **`div`** with an **id** "**`check`**" must be "**`You solve it! Congratulations!`**"

The text color of that paragraph must be **green.**

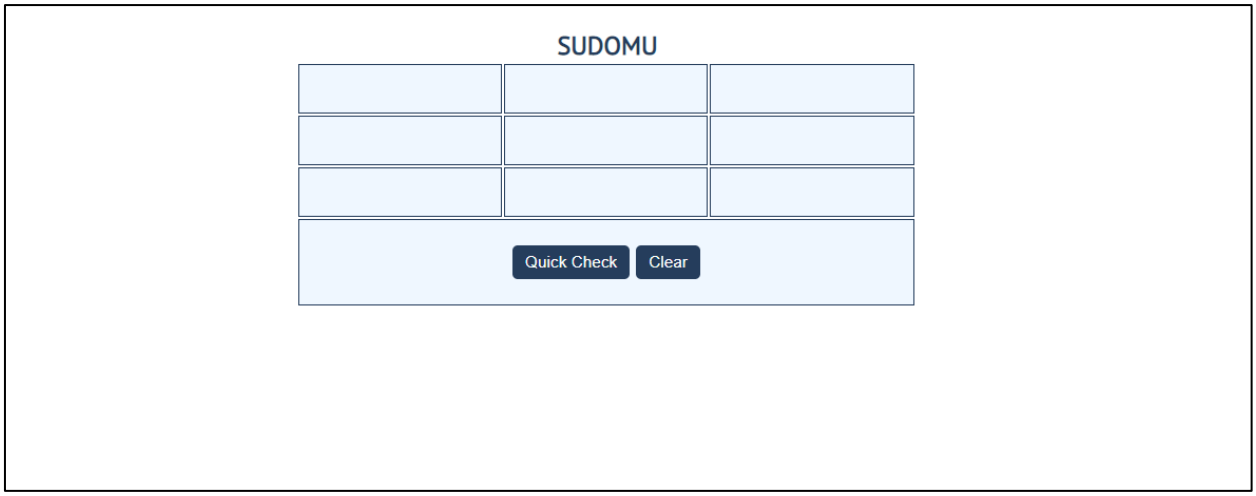
Otherwise, when the filled table **does not solve the sudomu,** the result should be:



The table border should be changed to: "**`2px solid red`**". The **`text content`** of the **paragraph** inside the **`div`** with an **id** "**`check`**" must be: "**`NOP! You are not done yet...`**"

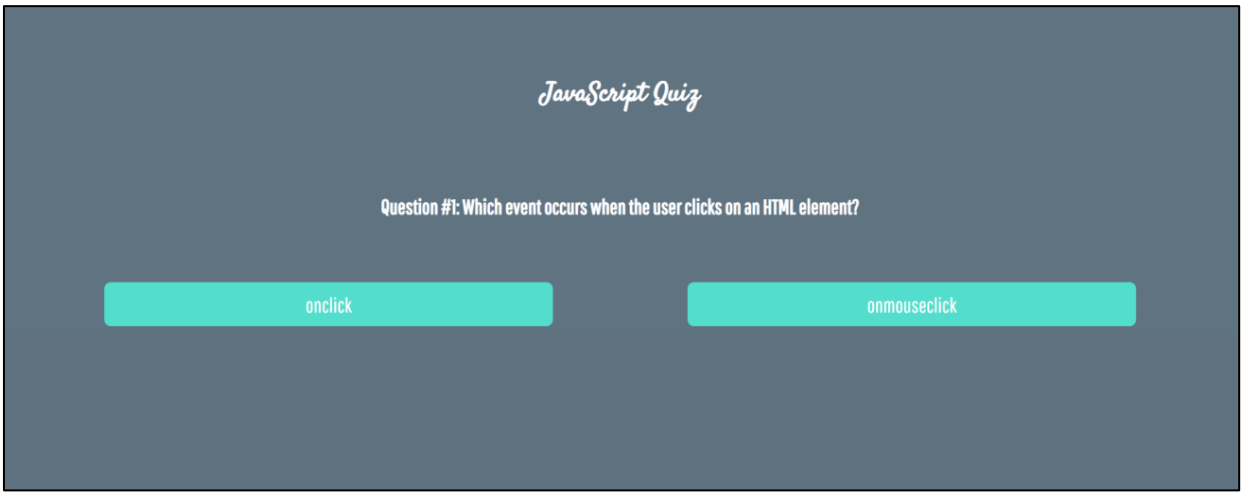
The text color of that paragraph must be **red!**

The ["**`Clear`**"] button **clears the whole SUDOMU (removes all numbers)** and the **paragraph which contains the messages. It also removes the table border.**



15. JavaScript Quizz

Write a function that has the functionality of a quiz.



Three sections contain **one question and 2 possible answers.**

The right answer is only one!

When one of the **list elements is clicked,** the next section **must appear (if any…)**.

After all three questions have been answered, the **`results ul`** must **appear,** (Use **`'none'`** and **`'block'`** to hide and show the question sections), and the **results** must be added in the **h1**.

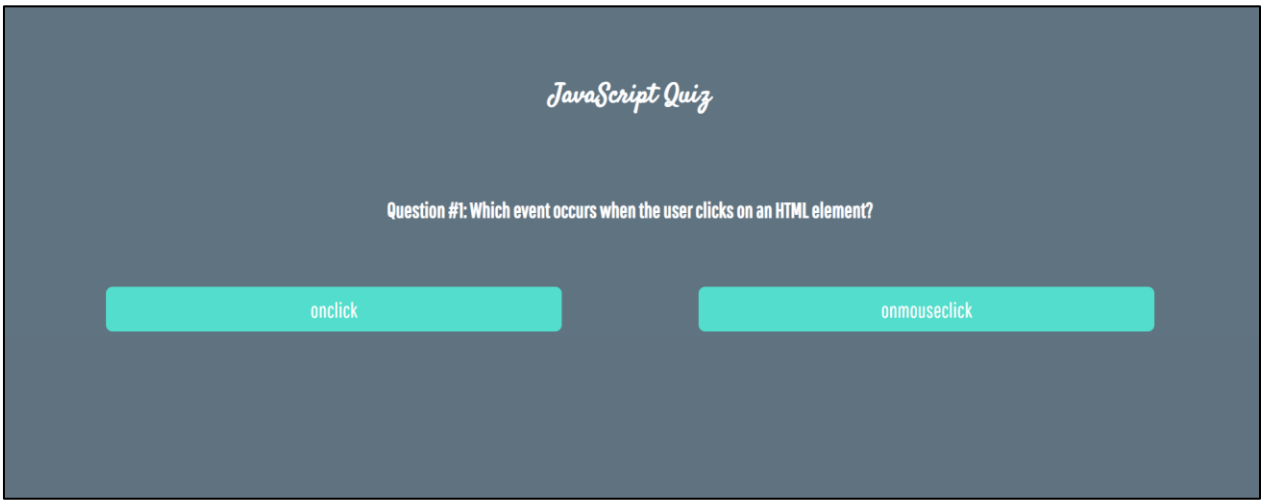
If all questions are answered correctly, you should print the following message:

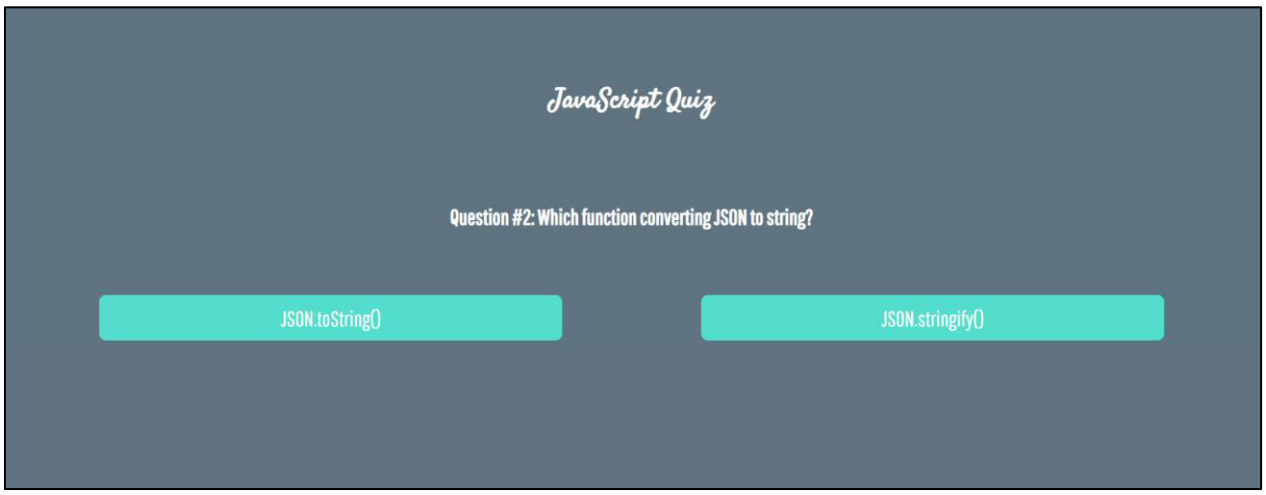
"**`You are recognized as top JavaScript fan!`**"

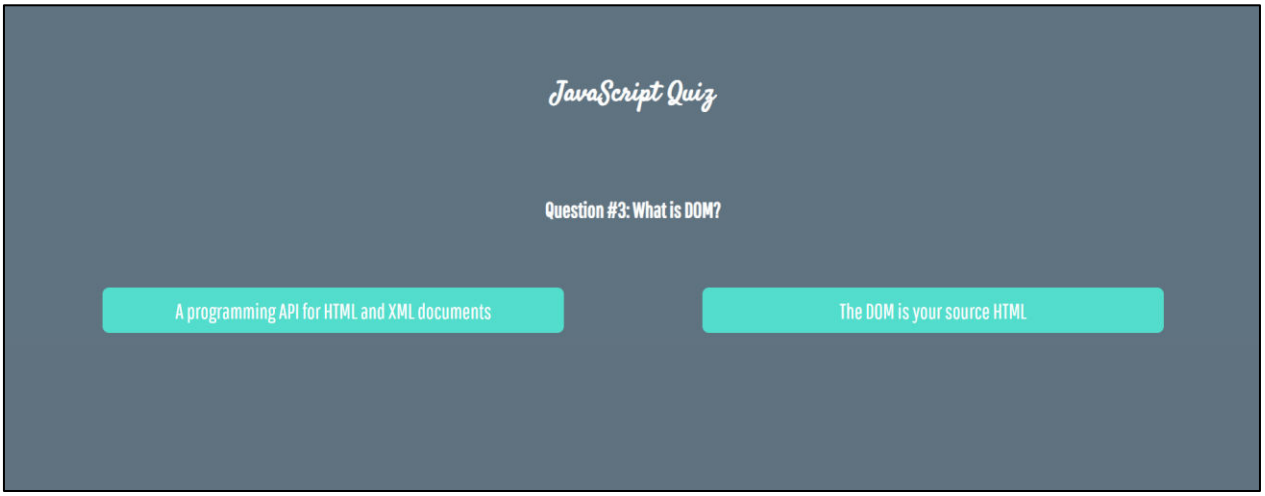
Otherwise, just print "**`You have {rightAnswers} right answers`**".

The right answers are:

- `onclick`
- `JSON.stringify()`
- `A programming API for HTML and XML documents`







JavaScript Quiz

You are recognized as top JavaScript fan!

JavaScript Quiz

You have 2 right answers