# Exercise: DOM and Events

## Subtraction

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

### HTML and JavaScript Code

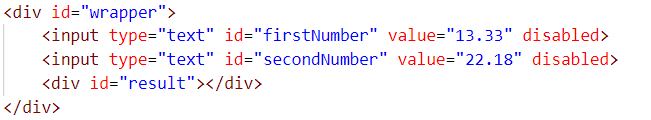
Implement the aboveto provide the following functionality:

* Your function should take the values of "**firstNumber**" and "**secondNumber**", **convert** them to numbers, **subtract** the second number from the first one and then append the result to the **<div>** 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.  


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**.



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

## 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!**"**

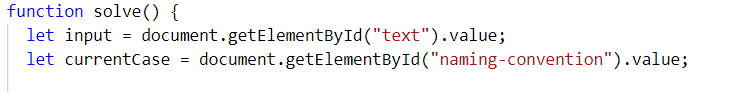
When the button is clicked, the function should convert the first string to either of the cases. The **output** should consist of only **one word** – the string you have modified. Once your **output** is done, you should set it as HTML to the **<span>** **element**. For more information, see the examples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



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**.

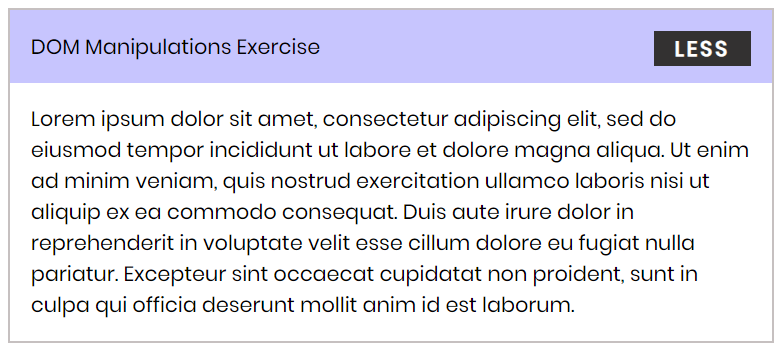


## Accordion

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]**.

## 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



## 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

 🡪 

## 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).

## 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.

## 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".**
* Finally, you should **clear** the value of the two **input** fields.

## 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**



After clicking the [**Encode and send it**] **button** the result should be:



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



## 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

