

# Exercise: DOM Manipulations

## 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 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.

```
<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**, the value of a textbox, as one would expect, is **text**. In order 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 now is to append the result to the **div**. We use the same function to get the **result** element by id and change its **text content** to the result of the **subtraction**.

```
function subtract() {  
  let num1 = Number(document.getElementById('firstNumber').value);  
  let num2 = Number(document.getElementById('secondNumber').value);  
  document.getElementById('result').textContent = num1 - num2;  
}
```

## What to submit?

Zip file containing the following:

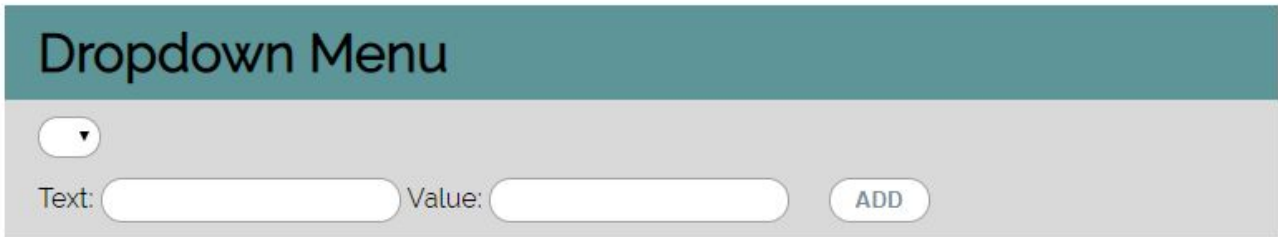
- subtract.js
- index.css
- index.html

File Name: SUBTRACTION.zip

## 2. 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



The screenshot shows a web interface with a teal header bar containing the text "Dropdown Menu". Below the header is a light gray container. Inside this container, there is a dropdown menu with a downward arrow, followed by two input fields. The first input field is preceded by the label "Text:" and the second by "Value:". To the right of these input fields is a rounded button labeled "ADD".

### 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.

## What to submit?

Zip file containing the following:

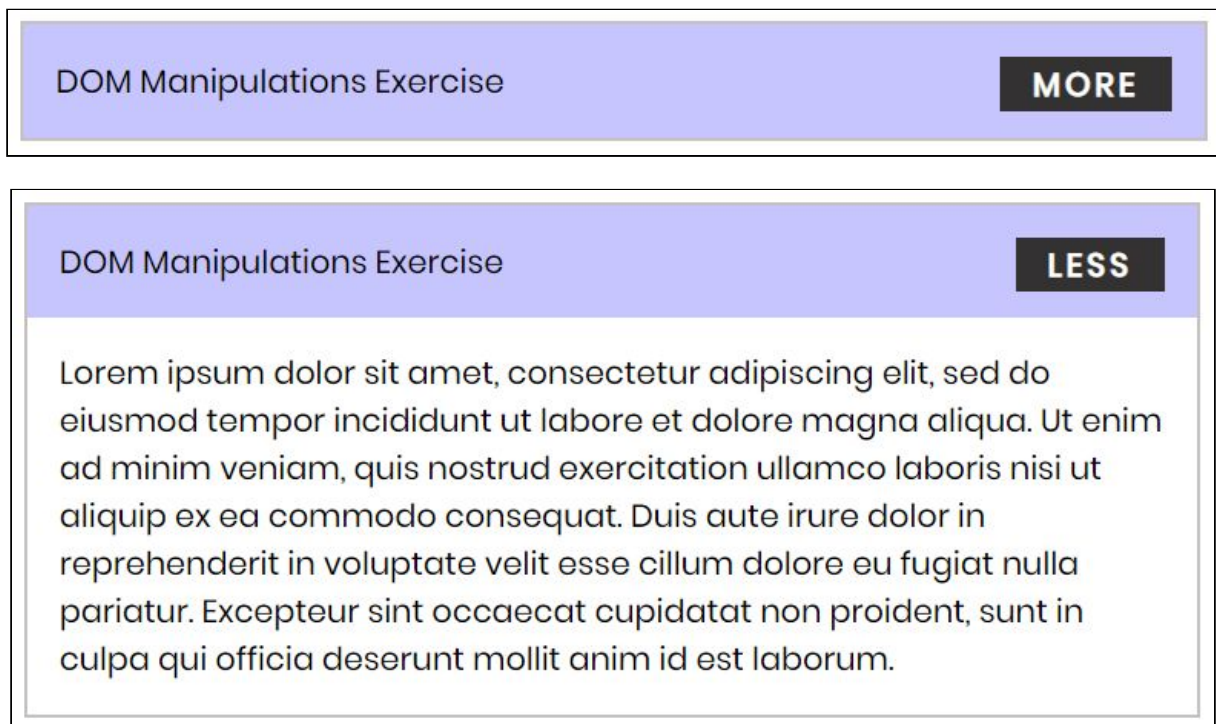
- dropdown.js
- filldropdown.css
- filldropdown.html

File Name: FILL-DROPDOWN.zip

## 3. Accordion

An **html** file is given and your task is to show **more/less** information by clicking a **[ADD] button** (it is not an actual button, but a **span** that has an **onclick** event attached to it). When **[More] button** is clicked, it **reveals** 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 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]**.

## What to submit?

Zip file containing the following:

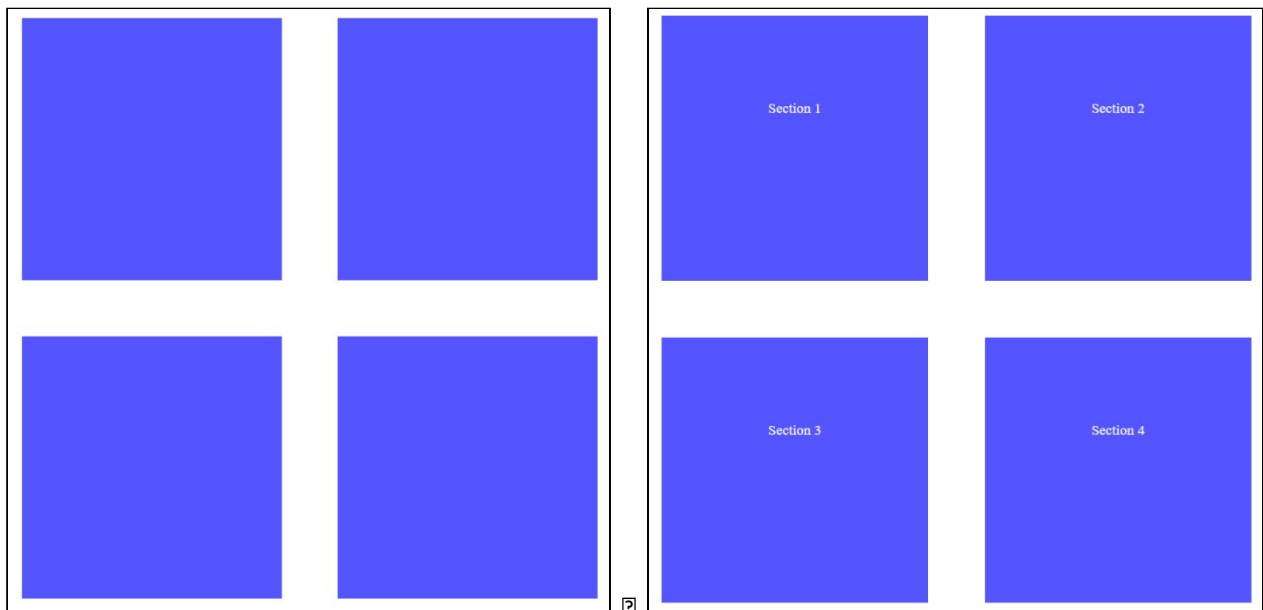
- accordion.js
- accordion.css
- accordion.html

File Name: ACCORDION.zip

## 4. 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



## What to submit?

Zip file containing the following:

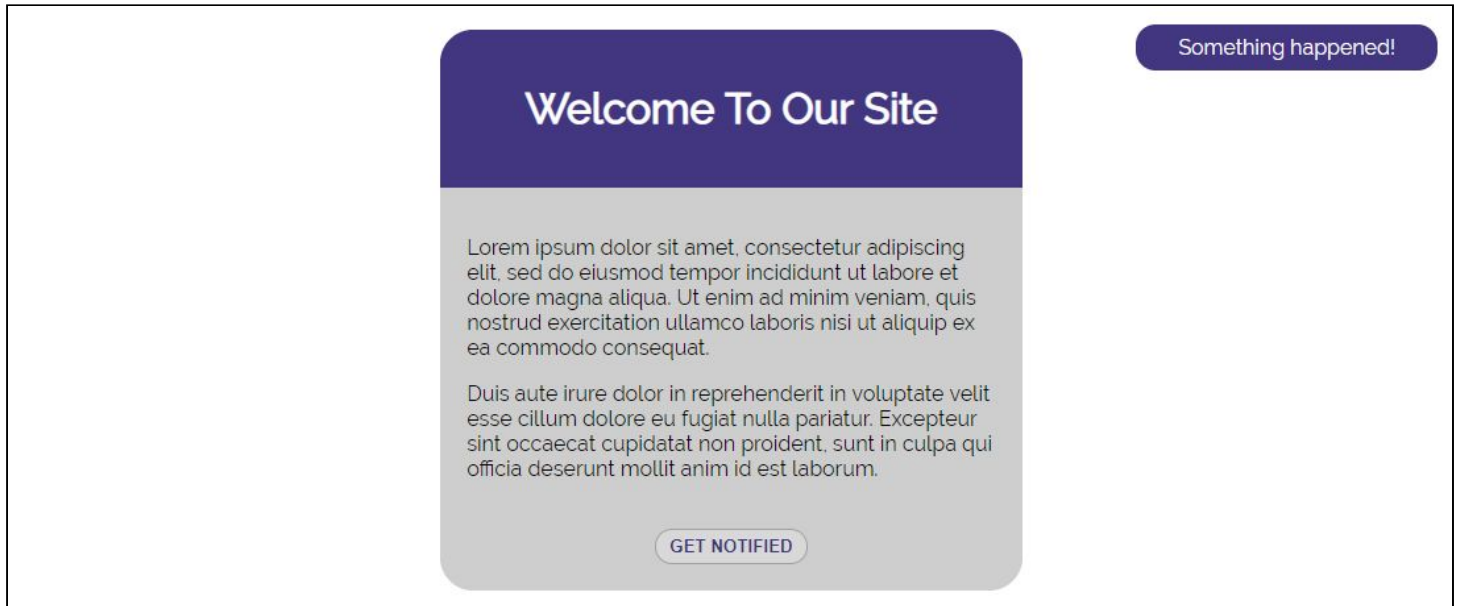
- script.js
- style.css
- index.html

File Name: SECTIONS.zip

## 5. Notification

Write a **function** that receives a string **message** and **displays** it inside a div with an id "**notification**" for 2 seconds. The div is initially **hidden** and when the function is called, it must be **shown**. After 2 seconds, **hide** the div. In the example below, a notification is shown when you **click** the button.

### Example



When we click the [GET NOTIFIED] button, a **div** appears in our upper-right corner. It should **disappear** in 2 seconds.

### What to submit?

Zip file containing the following:

- scripts.js
- style.css
- index.html

File Name: NOTIFICATION.zip