Exercises: DOM Manipulations

1. List of Items

Write a JS function that **reads** the text inside an input field and **appends** the specified text to a list inside an HTML page.

Submit **only** the **addItem()** function in judge.

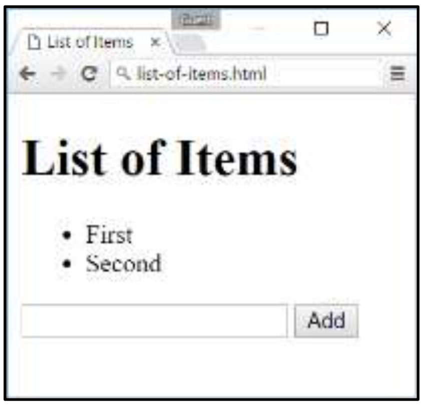
Input/Output

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

Sample HTML

`<h1>List of Items</h1>`
`<ul id="items"><li>First</li><li>Second</li></ul>`
`<input type="text" id="newItemText" />`
`<input type="button" value="Add" onclick="addItem()">`
`<script>`
`  function addItem() {`
`    // TODO: add new item to the list`
`  }`
`</script>`

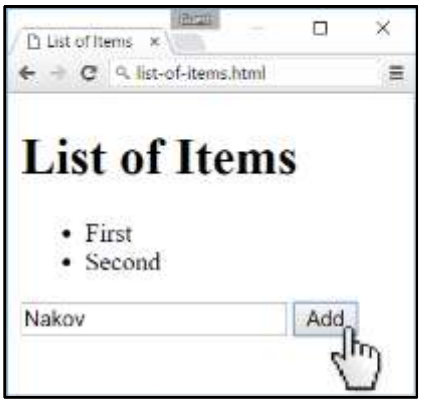
Examples



List of Items

- First
- Second

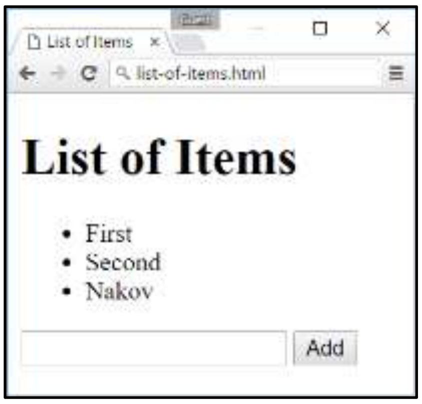
→



List of Items

- First
- Second

→



List of Items

- First
- Second
- Nakov

2. Add / Delete

Extend the previous problem to display a **[Delete] link** after each list item. **Clicking** it should **delete** the item with no confirmation.

Submit **only** the **addItem()** function in judge.

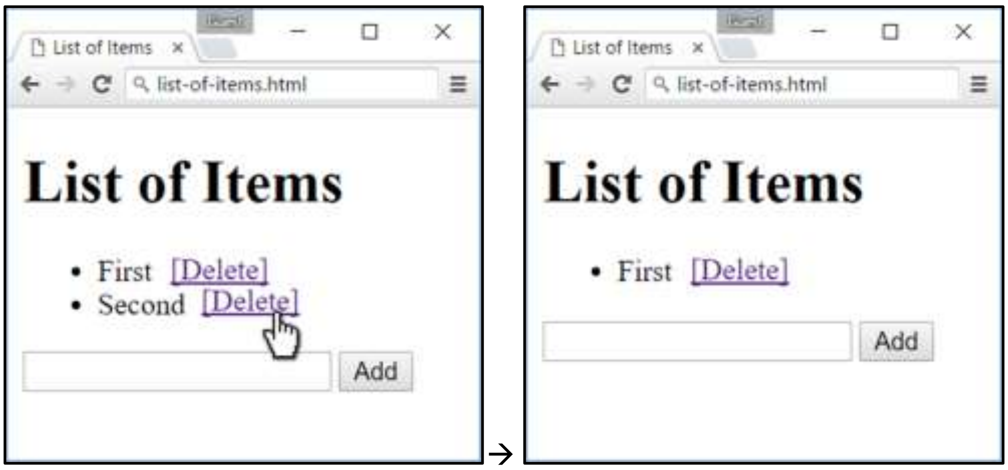
Input/Output

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

Sample HTML

`<h1>List of Items</h1>`
`<ul id="items"></ul>`
`<input type="text" id="newText" />`
`<input type="button" value="Add"`
`  onclick="addItem()">`
`<script>`
`  function addItem() {`
`       //TODO`
`     function deleteItem() {`
`          //TODO`
`     }`
`  }`
`</script>`

Examples



3. Delete from Table

Write a JS program that **takes** an e-mail from an **input field** and **deletes** the matching row from a table. If no entry is found, an **error** should be displayed in a **<div>** with ID "**results**". The error should be "**Not found**."

Submit **only** the **deleteByEmail()** function in judge.

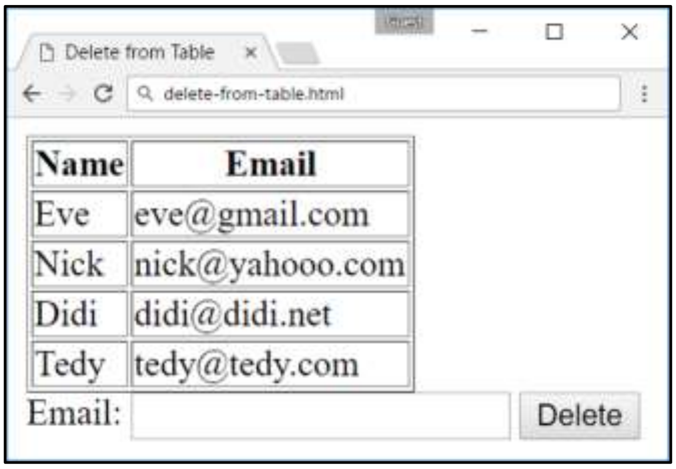
Input/Output

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

Sample HTML

```
<table border="1" id="customers">
 <tr><th>Name</th><th>Email</th></tr>
 <tr><td>Eve</td><td>eve@gmail.com</td></tr>
 <tr><td>Nick</td><td>nick@yahooo.com</td></tr>
 <tr><td>Didi</td><td>didi@didi.net</td></tr>
 <tr><td>Tedy</td><td>tedy@tedy.com</td></tr>
</table>
Email: <input type="text" name="email" />
<button onclick="deleteByEmail()">Delete</button>
<div id="result" />
<script>
     function deleteByEmail() {
          //TODO
     }
</script>
```

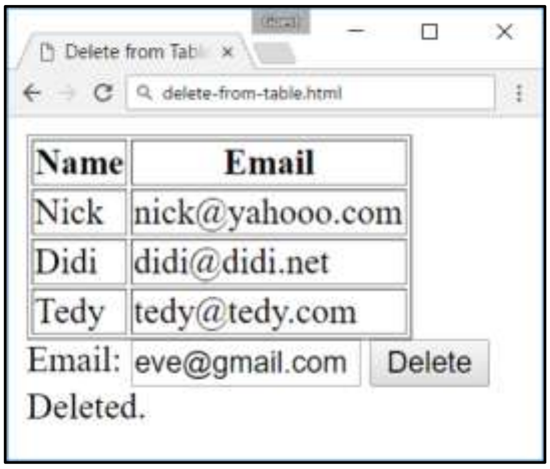
Examples



Name	Email
Eve	eve@gmail.com
Nick	nick@yahooo.com
Didi	didi@didi.net
Tedy	tedy@tedy.com

Email: [ ] Delete

→



Name	Email
Nick	nick@yahooo.com
Didi	didi@didi.net
Tedy	tedy@tedy.com

Email: eve@gmail.com Delete

Deleted.

4. Stopwatch

Write a **timer** that counts **minutes** and **seconds**. The user should be able to control it with **buttons**. Clicking **[Start] resets** the timer back to zero. Only **one** of the buttons should be enabled at a time (you cannot stop the timer, if it is not running).

Submit only the **stopwatch()** function in judge.

Input/Output

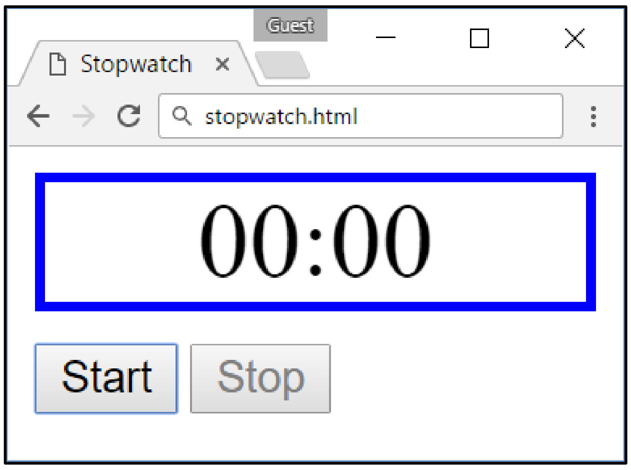
There will be no input/output, your program should instead **modify** the DOM of the given HTML document.

Sample HTML

```
<div id="time" style="border:3px solid blue; text-align:center; font-size:2em;
margin-bottom:10px">00:00</div>
```

```
<button id="startBtn">Start</button>
<button id="stopBtn" disabled="true">Stop</button>
<script>
     window.onload = function stopWatch() {
         //TODO
     }
</script>
```

Examples



5. Mouse Gradient

Write a JS program that **detects** and **displays** how far along a gradient the user has **moved** their **mouse**. Use the provided HTML and stylesheet (CSS) to test locally. The result should be **rounded down** and displayed as a **percentage** inside the **<div>** with ID "**result**".

Submit **only** the **attachGradientEvents()** function in judge. Make sure you write it in a **separate** file, called **gradient.js**.

Input/Output

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

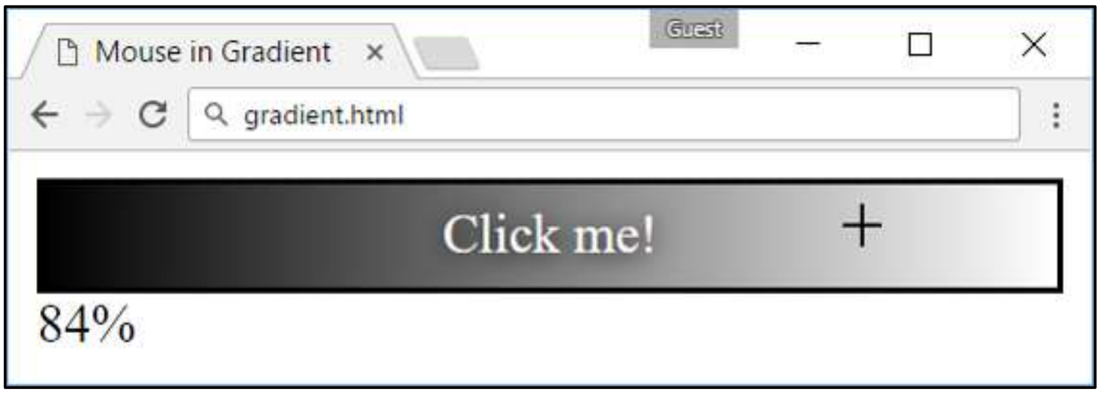
Sample HTML

```
<html>
<head>
  <title>Mouse in Gradient</title>
  <link rel="stylesheet" href="gradient.css" />
  <script src="gradient.js"></script>
</head>
<body onload="attachGradientEvents()">
  <div id="gradient-box">
    <div id="gradient">Click me!</div>
  </div>
  <div id="result"></div>
</body>
</html>
```

gradient.css

```
#gradient-box {
  width: 300px;
  border: 2px solid lightgrey;
}
#gradient-box:hover {
  border: 2px solid black;
}
#gradient {
  height: 30px;
  color: white;
  text-shadow: 1px 1px 10px black;
  text-align: center;
  line-height: 30px;
  background: linear-gradient(to right, black, white);
  cursor: crosshair;
}
```

Examples



6. Highlight Active

Write a JS function that **highlights** the **currently active** section of a document. There will be **multiple** divs with **input fields** inside them. Set the class of the div that contains the **currently focused** input field to "**focus**". When focus is lost (**blurred**), **remove the class** from the element.

Submit only the **focus()** function in judge.

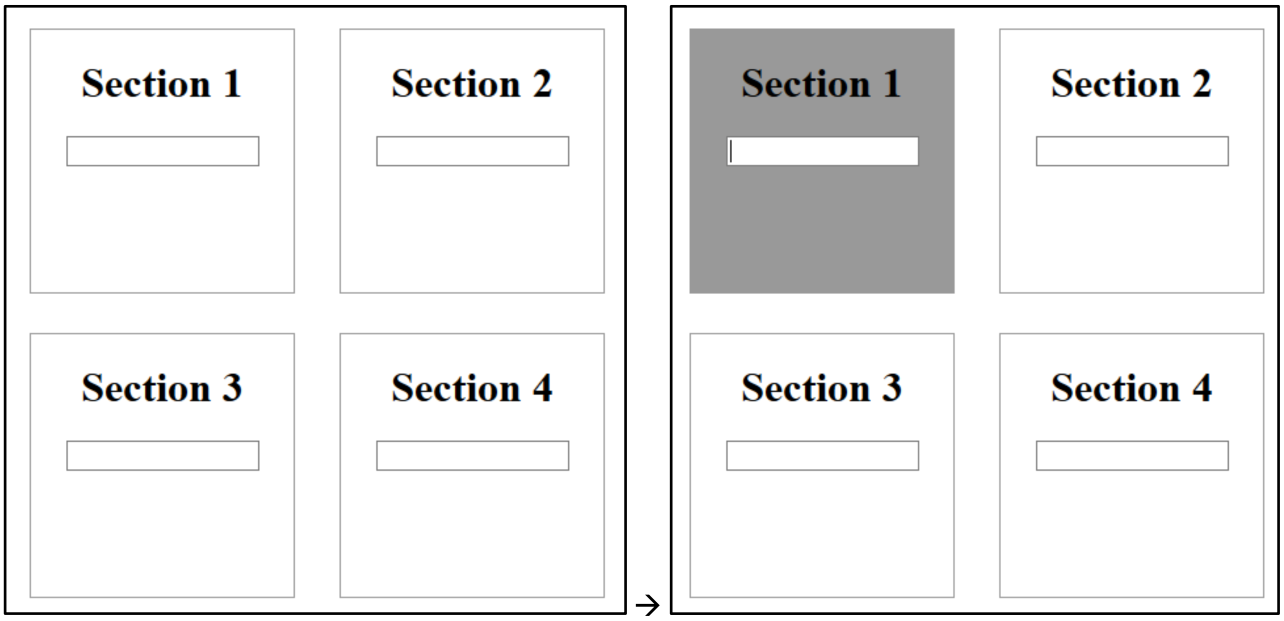
Input/Output

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

Sample HTML

```html
<!DOCTYPE html><html lang="en">
<head>
  <meta charset="UTF-8"><title>Focus</title>
  <style>
    div { width: 470px; }
    div div {
      text-align: center;
      display: inline-block;
      width: 200px;
      height: 200px;
      margin: 15px;
      border: 1px solid #999;
    }
    .focused { background: #999999; }
  </style>
</head>
<body onload="focus()">
  <div>
    <div><h1>Section 1</h1><input type="text"/></div>
    <div><h1>Section 2</h1><input type="text"/></div>
    <div><h1>Section 3</h1><input type="text"/></div>
    <div><h1>Section 4</h1><input type="text"/></div>
  </div>
  <script>
    function focus() {
      // TODO
    }
  </script>
</body>
</html>
```

Example



7. Dynamic Validation

Write a JS function that **dynamically validates** an email input field when it is **changed**. If the input is **invalid**, apply the style "**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.

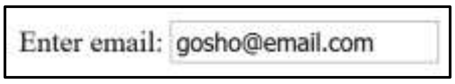
Input/Output

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

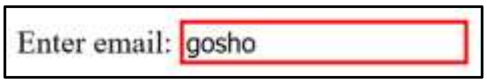
Sample HTML
```
<!DOCTYPE html><html lang="en">
<head>
  <meta charset="UTF-8"><title>Focus</title>
  <style>.error { border: 2px solid red; }</style>
</head>
<body onload="validate()">
  <label for="email">Enter email:</label>
  <input id="email" type="text"/>
  <script>
    function validate() {
      // TODO
```

```
    }
  </script>
</body>
</html>
```

Example


Enter email: gosho@email.com

→


Enter email: gosho