Lab 06: JavaScript Document Object Model (DOM)

Objective(s):

1. Learn Basics of Document Object Model

Lab Task(s):

Exercises

1. Write a JavaScript function to get the values of First and Last name of the following form (and show them in an alert dialog).

```
<!DOCTYPE html>
<html>
<head>
  <meta charset=utf-8 />
      <title>Return first and last name from a form </title>
   </head>
<body>
   <form id="form1" onsubmit="getFormvalue()">
     First name:
     <input type="text" name="fname" value="David"><br>
     Last name:
     <input type="text" name="lname"     value="Beckham"><br>
     <input type="submit" value="Submit">
   </form>
</body>
</html>
```

2. Write a JavaScript program to set the background color of a paragraph.

```
<!DOCTYPE html>
<html>

<head>
   <meta charset=utf-8 />
   <title>JS DOM</title>
```

```
</head>

<body>
   <input type="button" value="Click to set paragraph background color"
onclick="set_background()">
   <p>w3resource JavaScript Exercises</p>
   <p>w3resource PHP Exercises</p>
</body>

</html>
```

3. Here is a sample html file with a submit button. Write a JavaScript function to get the value of the href, hreflang, rel, target, and type attributes of the specified link.

```
<!DOCTYPE html>
<html>

<head>
   <meta charset=utf-8 />
</head>

<body>
   <p><a id="w3r" type="text/html" hreflang="en-us" rel="nofollow" target="_self"
         href="http://www.w3resource.com/">w3resource</a></p>
   <button onclick="getAttributes()">Click here to get attributes value</button>
</body>

</html>
```

4. Here is a sample html file with a submit button. Now modify the style of the paragraph text (such as fontSize, fontFamily, color, etc.) through javascript code.

```
<!DOCTYPE html>
<html><br>

<head>
   <meta charset=utf-8 />
   <title>JS DOM paragraph style</title>
</head>
```

```
<body>
   <p id='text'>JavaScript Exercises - w3resource</p>
   <div>
      <button id="jsstyle" onclick="js_style()">Style</button>
   </div>
</body>

</html>
```

5. Write a JavaScript function to add rows to a table.

```
<!DOCTYPE html>
<html>

<head><br>
   <meta charset=utf-8 />
   <title>Insert row in a table - w3resource</title>
</head>

<body>
   <table id="sampleTable" border="1">
      <tr>
         <td>Row1 cell1</td>
         <td>Row1 cell2</td>
      </tr>
      <tr>
         <td>Row2 cell1</td>
         <td>Row2 cell2</td>
      </tr>
   </table><br>
   <input type="button" onclick="insert_Row()" value="Insert row">
</body>

</html>
```

6. Given the following HTML:

```
<!DOCTYPE html>
```

```html
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <div class="header">
  </div>
  <section id="container">
    <ul>
      <li class="first">one</li>
      <li class="second">two</li>
      <li class="third">three</li>
    </ul>
    <ol>
      <li class="first">one</li>
      <li class="second">two</li>
      <li class="third">three</li>
    </ol>
  </section>
  <div class="footer">
  </div>
</body>
```

Write the code necessary to do the following:

1. Select the `section` with an id of `container` without using `querySelector`.
2. Select the `section` with an id of `container` using `querySelector`.
3. Select all of the list items with a class of "second".
4. Select a list item with a class of third, but only the list item inside of the `ol` tag.
5. Give the `section` with an id of `container` the text "Hello!".
6. Add the class `main` to the `div` with a class of `footer`.
7. Remove the class `main` on the `div` with a class of `footer`.
8. Create a new `li` element.
9. Give the `li` the text "four".
10. Append the `li` to the `ul` element.
11. Loop over all of the `li`s inside the `ol` tag and give them a background color of "green".

12. Remove the div with a class of `footer`.

7. Given the following HTML, create a `script.js` file to complete the first two parts.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>DOM Exercise</title>
    <style>
        div {
          width: 50px;
          height: 50px;
          display: inline-block;
        }
        .brown{
          background-color: brown;
        }
        .green{
          background-color: green;
        }
        .blue{
          background-color: blue;
        }
        .purple{
          background-color: purple;
        }
        .yellow{
          background-color: yellow;
        }
        .car1 {
         background-color: #8C9C12;
        }
        .car2 {
         background-color: #1DA788;
        }
        .car1, .car2 {
            margin-left: 0;
        }
    </style>
</head>
<body>
    <h1 id="change_heading">Change Me!</h1>
    SELECTED COLOR <span class="selected">None!</span>
    <section>
        <div class="brown"></div>
        <div class="green"></div>
        <div class="blue"></div>
        <div class="yellow"></div>
    </section>
    <h2>Race!</h2>
```

```
    <button>Start the race!</button>
    <br>
    <div class="car1"></div>
    <br>
    <div class="car2"></div>
    <script src="script.js"></script>
</body>
</html>
```

1. Add the necessary code to wait for the DOM to load to make sure that anything you manipulate in the DOM has loaded. You can do this either using window.onload or adding an event listener for DOMContentLoaded.

2. Replace the text "Change me" with "Hello World!".

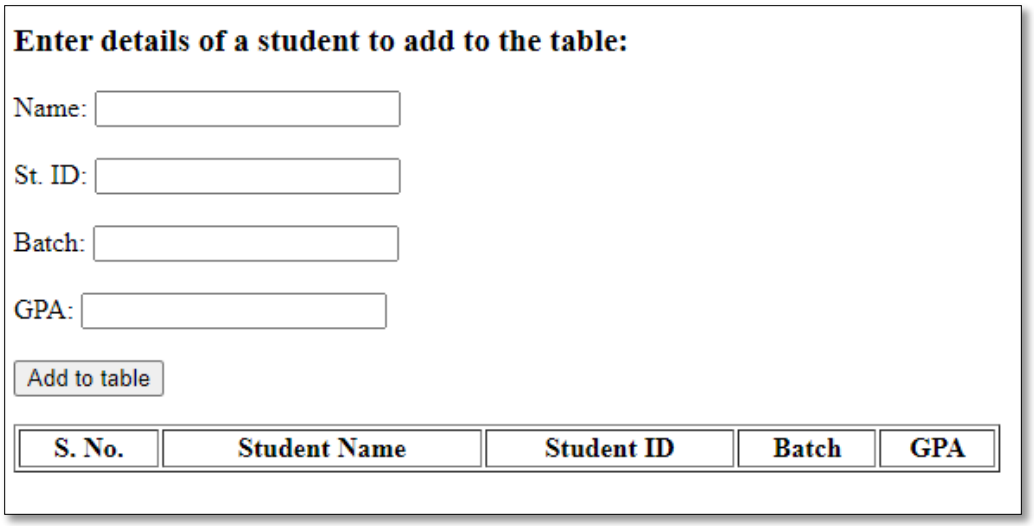
3. When a user hovers over one of the colored boxes change the text to display the color that is being hovered over.

4. Create a new div element.

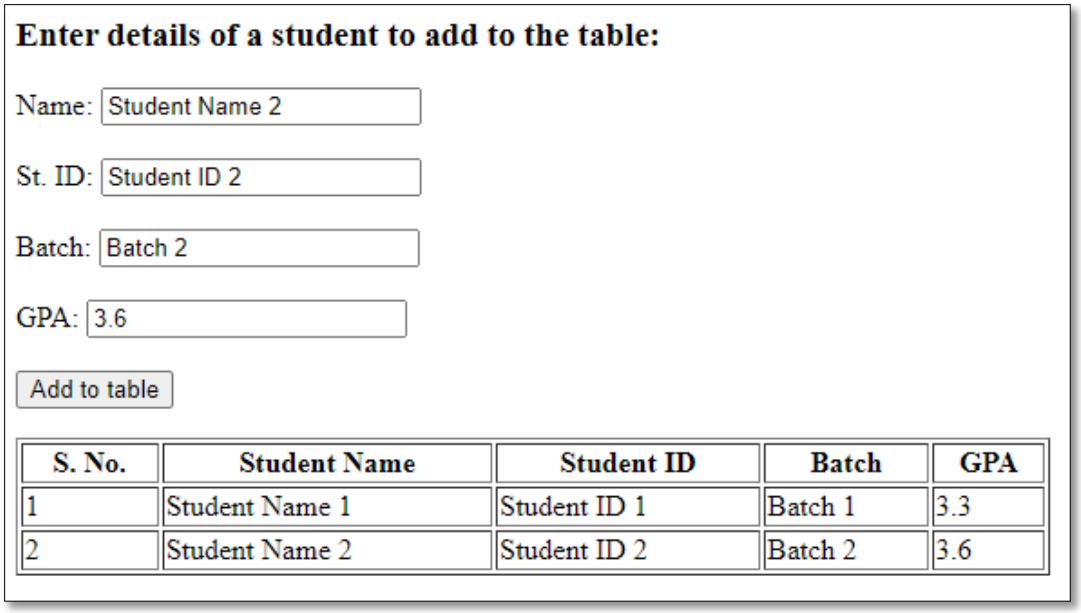
5. Give your new div a class of purple and style it so that it has a background color of purple.

6. Append your new div to the page to the section tag.

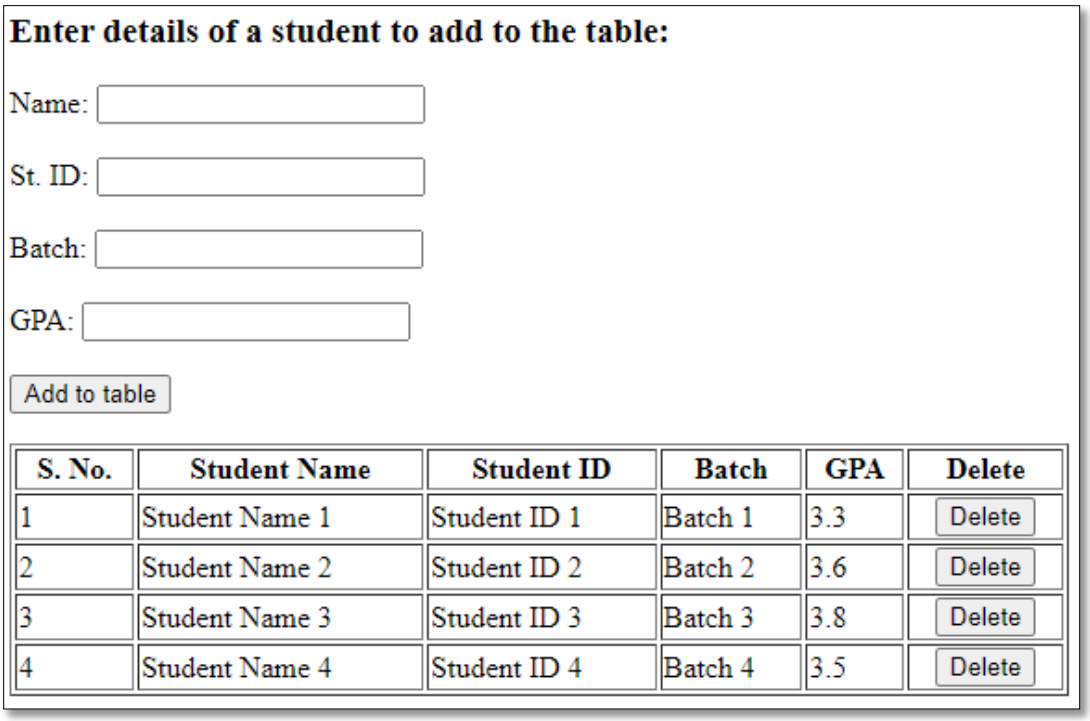
8. Create an HTML page that should contain a few text fields to get input from the user, a button captioned "**Add to table**", and a table. Initially the table should only contain a header row (no other data should be there) as shown below:



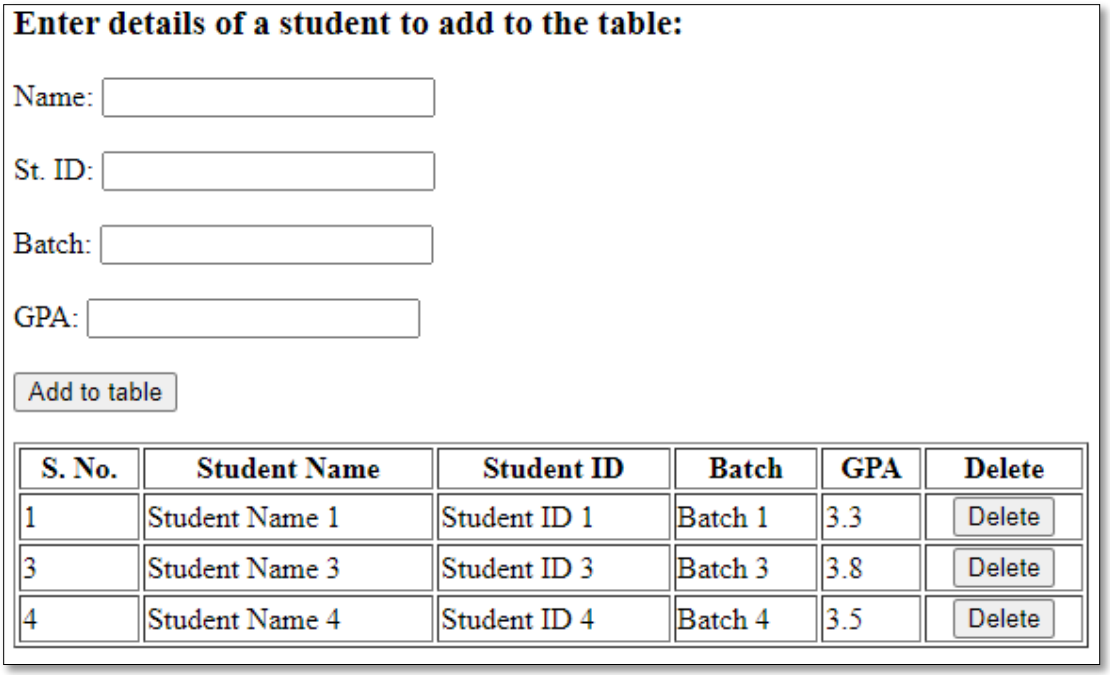
Then, whenever the user enters some data and presses the button, a new row/record comprising the entered data should be added to the table dynamically (using JavaScript) as shown in the sample below:



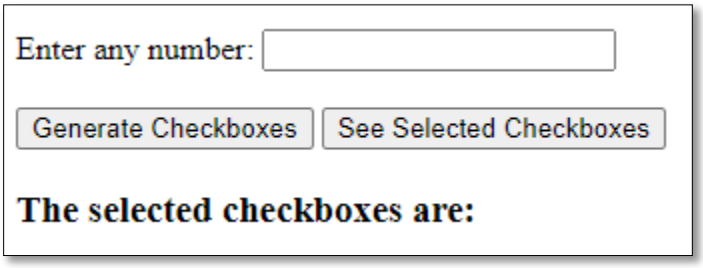
9. Extend the functionality of the previous task. Now, there should be an additional column in the table, where in every row, there should a button captioned **Delete**, as shown below:



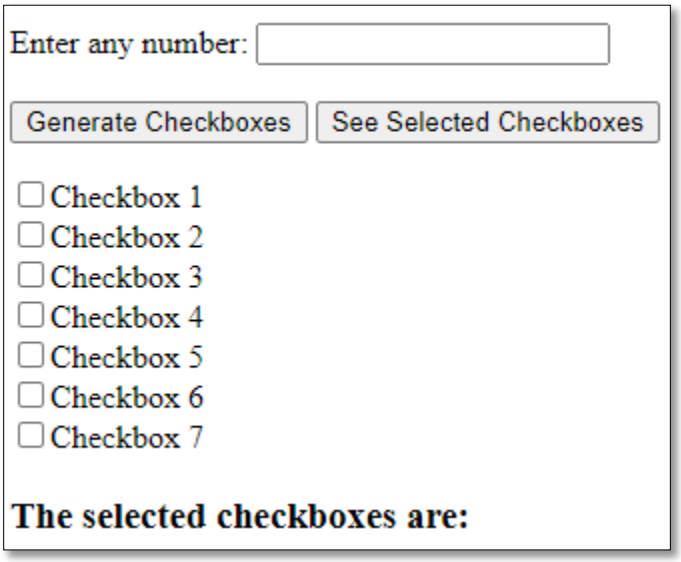
When the user clicks the **Delete** button, then that entire row should be deleted (in which row that button was present). For example, when the user clicks the delete button in the second row, then second row should be deleted and table should be updated, as shown below:



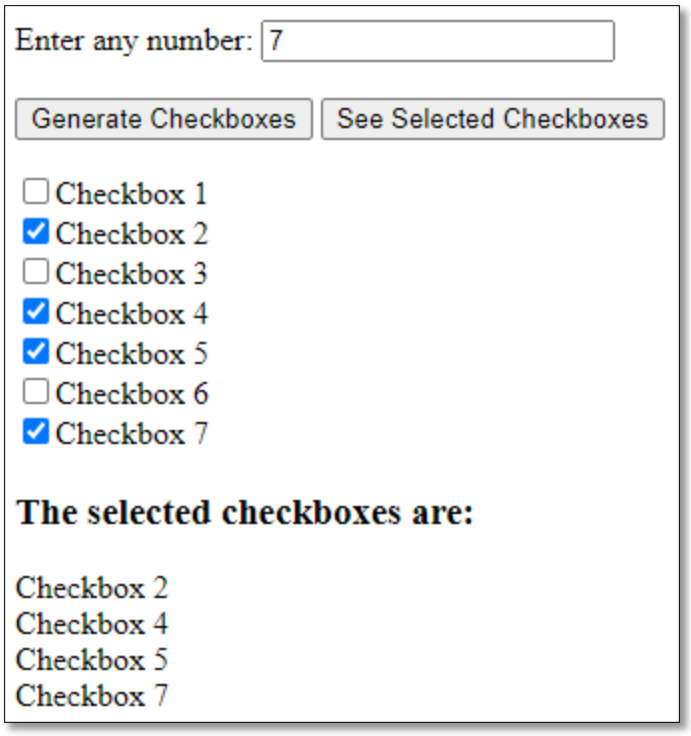
10. Create an HTML page that should initially contain a text field and two buttons as shown below:



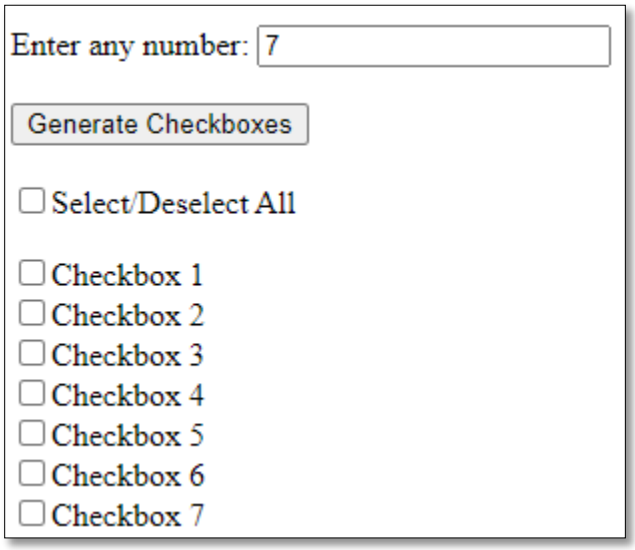
The user should enter a number indicating how many checkboxes he/she wants to create/generate. And then when the "**Generate Checkboxes**" button is clicked, that much number of checkboxes should be created dynamically at runtime (using JavaScript) as shown in the sample given below.



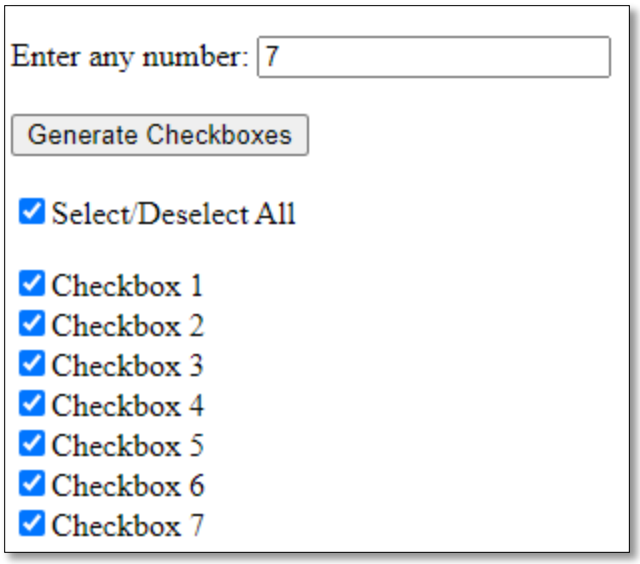
Besides, there should also be another button with the caption "**See Selected Checkboxes**", and when it is clicked, you should tell which checkboxes are checked/selected (using JavaScript).



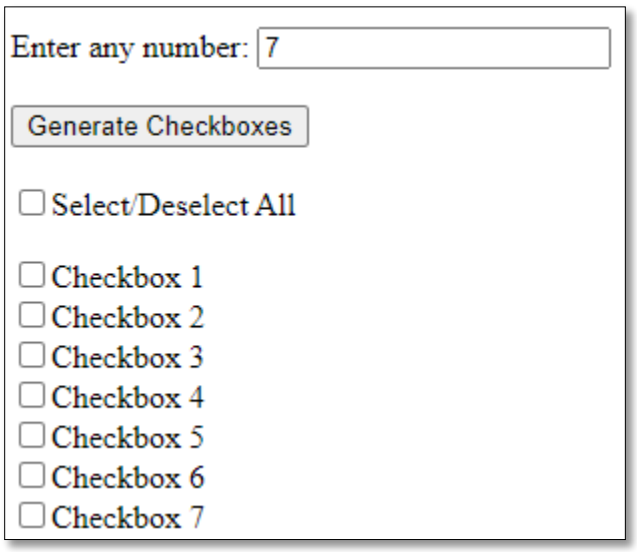
11. Create a task similar to the previous one, where user should enter a number and that much number of checkboxes should be created when the button is clicked, as shown in the sample below:



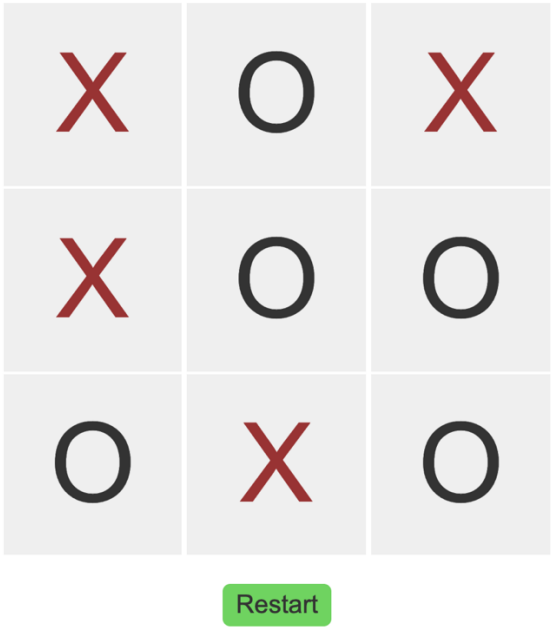
In addition, there should be a checkbox with the label/caption "**Select/Deselect All**", and when that checkbox is selected/checked, then all the other checkboxes should also be automatically selected/checked, as shown below:



In the same way, when that checkbox is deselected/unchecked, then all the other checkboxes should be automatically deselected/unchecked, as shown below:



12. Create a Tic-Tac-Toe game with two players. Following is a sample output.



END