Chapter

10

Document Object Model

At the end of this chapter, you will be able to:

- Add Table Cells using the jQuery append and find function
- Add HTML elements

Instruction 1: Add Table Cells using the jQuery Append and find function

1. The HTML DOM (Document Object Model) allows you to write JavaScript codes to control all HTML elements on your page. For example, you can write DOM codes to add a new cell to a table or change the "href" element of an existing <a>.

2. To illustrate the functionalities of DOM, we will write a program to display a multiplication table in a HTML table. We will modify the multiplication table program that you wrote in Chapter 9.

3. Create a new project, C10.

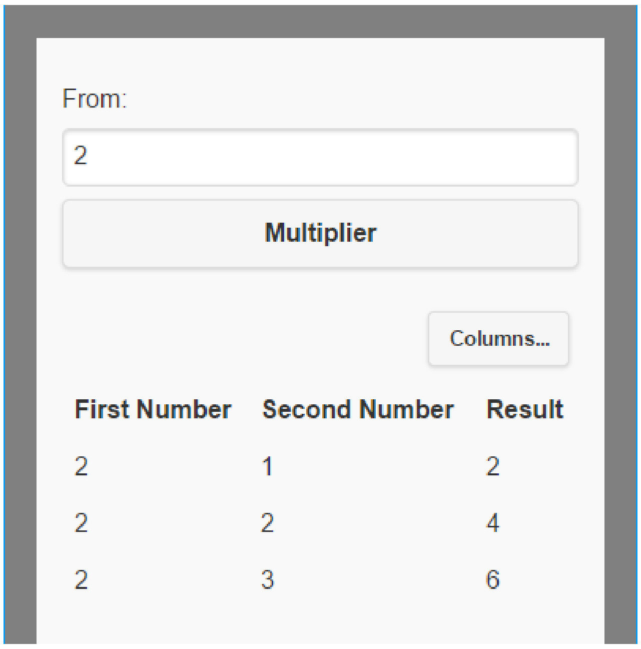
4. Follow Instruction1 in Chapter 4 to include jQuery and jQuery Mobile libraries into your project.

5. Amend the program that you wrote in Chapter 9, Instruction 1, Point 22. Edit index.html:

```html
<div id="result">
    <table data-role="table" data-mode="columntoggle" class="ui-responsive" id="myTable">
        <thead>
            <tr>
                <th>First Number</th>
                <th>Second Number</th>
                <th>Result</th>
            </tr>
        </thead>
        <tbody id="mybody">
            <tr>
                <td>2</td>
                <td>1</td>
                <td>2</td>
            </tr>
            <tr>
                <td>2</td>
                <td>2</td>
                <td>4</td>
            </tr>
            <tr>
                <td>2</td>
                <td>3</td>
                <td>6</td>
            </tr>
        </tbody>
    </table>
</div>
```

   a. We have added a table between <div id="result">. Some dummy data have been added between line 50 and 64 to show how the program will look like after it has been completed.

6. Run the program and note the output:



From:

2

Multiplier

Columns...

First Number	Second Number	Result
2	1	2
2	2	4
2	3	6

7. In this program, the user will enter the number that he wishes to display a multiplication for. The program then creates a table to display the multiplication table for this number. In the screen shot above, the multiplication table of 2 (i.e. 2 x 1 = 2, 2 x 2 = 4, 2 x 3 = 6) is shown.

8. Remove the dummy data from lines 11 to 25.

9. Change the multiply() function in index.js the following:

```
 1  function multiply(){
 2     var from, result;
 3     from = parseInt($("#txtFrom").val());
 4
 5
 6     for (i=1; i<=10; i++){
 7         result = from * i;
 8         $("#mybody").append("<tr><td>" + from.toString() + "</td><td>" +
  i.toString() + "</td><td>" + result.toString() + "</td></tr>");
 9      }
10  }
```

a. Line 2 declares variables from and result.

b. Line 3 copies the value that the user enters in the text field "from" to the JavaScript variable "from".

c. Line 6 executes lines 7 and 8 continuously for i is equal to 1 to 10.

d. Line 7 multiply "from" with "i" to produce "result".

e. Line 8 calls the jQuery function to append new cells to the <div> "mybody" which is the id of the table body declared in line 49 of point 10 above.

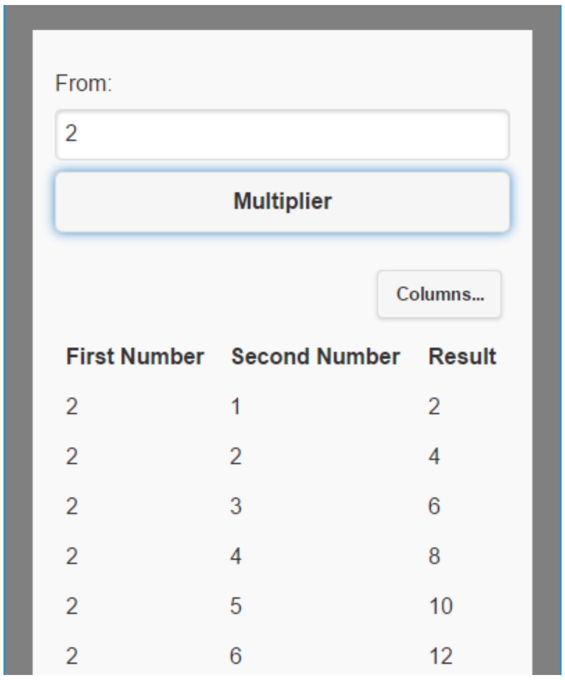
```
$("#mybody").append("<tr><td>" + from.toString() + "</td><td>" + i.toString()
+ "</td><td>" + result.toString() + "</td></tr>");
```

Notice that the codes above appends the value of "from", "i" and "result" to a series of HTML tags <tr><td></td></tr> and uses the jQuery append function to add these HTML codes to the HTML element "mybody".

10. The table below illustrates the values of the variables on each round in the loop.

Iteration	from	i	result	HTML codes appended to mybody
1	2	1	2	<tr><td>2</td><td>1</td><td>2</td></tr>
2	2	2	4	<tr><td>2</td><td>2</td><td>4</td></tr>
3	2	3	6	<tr><td>2</td><td>3</td><td>6</td></tr>
4	2	4	8	<tr><td>2</td><td>4</td><td>8</td></tr>
5	2	5	10	<tr><td>2</td><td>5</td><td>10</td></tr>
6	2	6	12	<tr><td>2</td><td>6</td><td>12</td></tr>
7	2	7	14	<tr><td>2</td><td>7</td><td>14</td></tr>
8	2	8	16	<tr><td>2</td><td>8</td><td>16</td></tr>
9	2	9	18	<tr><td>2</td><td>9</td><td>18</td></tr>
10	2	10	20	<tr><td>2</td><td>10</td><td>20</td></tr>

11. Run the program and note the output.



From:

2

Multiplier

Columns...

First Number	Second Number	Result
2	1	2
2	2	4
2	3	6
2	4	8
2	5	10
2	6	12

12. You would have noticed that when you run the program the second time, JavaScript simply appends your new multiplication table below the previous one. We will need to tell JavaScript to clear the table every time you click [Multiplier]. To do so, amend your multiply() function to the following. Note the new line of code that has been underlined below:

```
 1 function multiply() {
 2     var from, result;
 3     $("#mybody").find("tr").remove();
 4     from = parseInt($("#txtFrom").val());
 5
 6     for (i = 1; i <= 10; i++) {
 7         result = from * i;
 8         $("#mybody").append("<tr><td>" + from.toString() + "</td><td>" +
 9 i.toString() + "</td><td>" + result.toString() + "</td></tr>");
       }
10 }
```

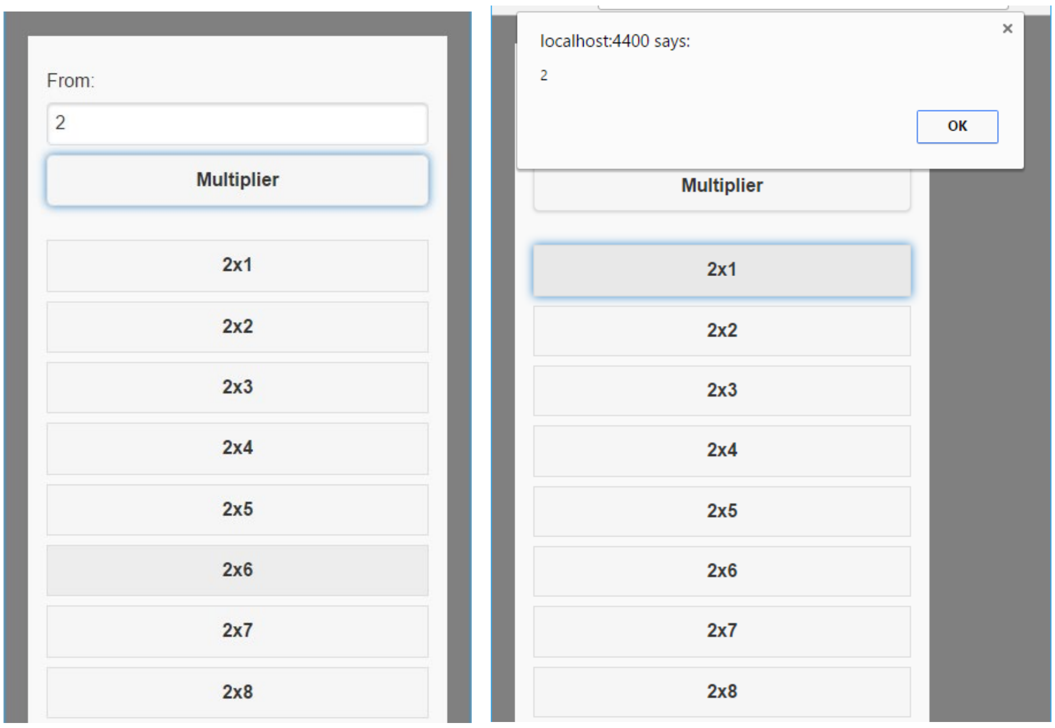
13. The code in line 3 tells JavaScript to remove all occurrences of <tr></tr> in mybody. Doing so removes all the rows and columns that were added when you ran the program previously.

```
$("#mybody").find("tr").remove();
```

Instruction 2: Add HTML elements

1. To illustrate how a HTML element such as <img> and <a> with its attributes such as <img src=…> and <a href=…> can be added, edited and removed in JavaScript using DOM, we will change our multiplication table program to use the <a href…> tag to display the multiplication table.

2. In this program, the multiplication table is displayed in a series of buttons. When the corresponding button is clicked, the result is displayed in an alert as follows:



localhost:4400 says:

2

3. The HTML codes for an <a> tag is as follows:

```
<a href="#" class="ui-btn" id="btn_1">2 x 1</a>
```

   a. The code above displays the text "2 x 1" in the form of a button (using class ui-btn).

   b. When the button is clicked, it pops an alert that displays the number 2.

4. Edit index.html:

```
<div data-role="main" class="ui-content">
    <form name="calculateform" id="calculateform">
        <div class="ui-field-contain">

            <div data-role="fieldcontainer">
                <label for="txtFrom">From:</label>
                <input type="text" name="txtFrom" id="txtFrom"
value="2">
```

```
            </div>

            <input type="button" value="Multiplier" id="btnMultiply">
        </div>
    </form>

    <div id="result">

    </div>
</div>
```

5. Edit index.js:

```
 1 function multiply() {
 2     var from, result;
 3     $("#mybody").find("tr").remove();
 4     from = parseInt($("#txtFrom").val());
 5
 6     var htmlstring;
 7
 8     for (i = 1; i <= 10; i++) {
 9         result = from * i;
10         htmlstring = "<a href='#' class='ui-btn' id='btn_" + i + "'>" +
from.toString() + "x" + i.toString() + "</a>";
11
12         $("#result").append(htmlstring);
13
14         $("#btn_" + i).bind("click", {id: result}, function (event) {
15             var data = event.data;
16             alert(data.id);
17         });
18     }
19 }
```

a. Line 10 creates a new HTML element <a> of the following format:

```
<a href="#" class='ui-btn' id='btn_i">from x to</a>
```

The variables i, from and to will be replaced with the values changed in each round of the for loop.

b. Line 12 appends the new html string to the "result" <div> division.

c. Line 14 is an event handler. It binds this button to the actions in line 15 and 16. The method to bind a set of actions an event such as a "click" is as follows:

```
$("#btn_i").bind("click", {id: result}, function(event) {…} );
```

Here, the value of "result" is passed into the binding as an id.

d. Line 15 finds out what is the value that has been passed into this event handler, and assigns it to the variable, data. In line 14, the variable "result" has been passed into the event as an "id".

e. Line 16 pops this id as an alert.

6. The table below illustrates the values of the variables on each round in the loop.

Iteration	from	i	result	HTML codes appended to result	data.id
1	2	1	2	<a href="#" class="ui-btn" id="btn_1">2 x 1</a>	2
2	2	2	4	<a href="#" class="ui-btn" id="btn_2">2 x 2</a>	4
3	2	3	6	<a href="#" class="ui-btn" id="btn_3">2 x 3</a>	6
4	2	4	8	<a href="#" class="ui-btn" id="btn_4">2 x 4</a>	8
5	2	5	10	<a href="#" class="ui-btn" id="btn_5">2 x 5</a>	10
6	2	6	12	<a href="#" class="ui-btn" id="btn_6">2 x 6</a>	12
7	2	7	14	<a href="#" class="ui-btn" id="btn_7">2 x 7</a>	14
8	2	8	16	<a href="#" class="ui-btn" id="btn_8">2 x 8</a>	16
9	2	9	18	<a href="#" class="ui-btn" id="btn_9">2 x 9</a>	18
10	2	10	20	<a href="#" class="ui-btn" id="btn_10">2 x 10</a>	20

14. You would have noticed that when you run the program the second time with a new number in the "Multiply From" text field, JavaScript simply appended your new multiplication table below the previous one. We will need to tell JavaScript to clear the table every time you click [Multiplier]. To do so, amend your multiply() function to the following. Note the new line of code underlined below:

```
 1 function multiply() {
 2     var from, result;
 3     $("#mybody").find("tr").remove();
 4     from = parseInt($("#txtFrom").val());
 5 
 6     $("#result").empty();
 7 
 8     var htmlstring;
 9 
10     for (i = 1; i <= 10; i++) {
11         result = from * i;
12         htmlstring = "<a href='#' class='ui-btn' id='btn_" + i + "'>" +
   from.toString() + "x" + i.toString() + "</a>";
13 
```

```
        $("#result").append(htmlstring);

        $("#btn_" + i).bind("click", {id: result}, function (event) {
            var data = event.data;
            alert(data.id);
        });
    }
}
```

15. The code below tells JavaScript to remove every child element that belongs to a parent. Thus, this function will remove every child element belonging to the "result" division. Doing so removes all the <a> elements added when you ran the program previously.

```
$("#result").empty();
```

Exercise

There are no exercises for this chapter.

Codes

Codes for this chapter can be found by searching VC10-DOM on Github.com.