JavaScript To Do List

This project assumes that you have completed the HTML5 static to-do list project. We will be building directly on that base. You should have a static web page that looks a bit like this.

In this project we will be adding the ability to add items to the list using a small form underneath the list. The items will then be created by the user when the load up the page. This should make the application much more useful. It will look like this.

So the first thing we shall do is to add the new form to the HTML page, then we can start writing JavaScript.

Write the Add New Item Form

The form will be in a new fieldset, to keep it apart from the list itself. So add a new <fieldset> tag underneath our existing one. This screenshot shows how our new fieldset goes just after the existing one, but still inside the </form> and </main> sections.

```
36          </fieldset>
37
38          <fieldset>
39
40          </fieldset>
41      </form>
42  </main>
```

Once again I am going to use the <legend> to instruct the user. Add a <legend> tag with the content ‘Add New Item’.

```
38          <fieldset>
39              <legend>Add New Item</legend>
40          </fieldset>
```

Next we will add an input field for the user to type the new item. An input text field looks like this. Note that I have given the text field an **id** of **newItem**. We will need this **id** value later in our JavaScript.

```
39          <legend>Add New Item</legend>
40          <input id="newItem" placeholder="type in a new item" type="text">
41      </fieldset>
```

Continue on next page...

Next we need to add a button using the following HTML.

```
40          <input id="newItem" placeholder="type in a new item" type="text">
41          <button id="cmdAdd" type="button">Add</button>
42      </fieldset>
```

Lastly for this section, we shall remove all the to-do items that are baked into our HTML. Delete the lines in the first <fieldset> that are <input><label>
 that make up our items. A single item looked like this.

```
21      <input id="item1" type="checkbox">
22      <label class="strikethrough" for="item1">Wash Clothes</label>
23      <br>
```

Once you have deleted all the hard coded items, the first <fieldset> will look like this.

```
16  <main>
17      <form onsubmit="return false">
18          <fieldset id="toDoFields">
19              <legend>Tick off as complete</legend>
20
21          </fieldset>
```

And the page will look like this.

Start Writing JavaScript

In this section we will import jQuery into our page, and then write the first bit of JavaScript. To import jQuery add the following <script> tag to your page in the <head> section of the HTML.

```
<!-- jQuery -->
<script src="https://code.jquery.com/jquery-3.1.1.js"
        integrity="sha256-16cdPddA6VdVInumRGo6IbivbERE8p7CQR3HzTBuELA="
        crossorigin="anonymous"></script>
```

You can copy and paste it from here

```
<!-- jQuery -->
<script src="https://code.jquery.com/jquery-3.1.1.js "
integrity="sha256-16cdPddA6VdVInumRGo6IbivbERE8p7CQR3HzTBuELA="
crossorigin="anonymous"></script>
```

Add another <script> tag to contain our JavaScript. At first it will look like this.

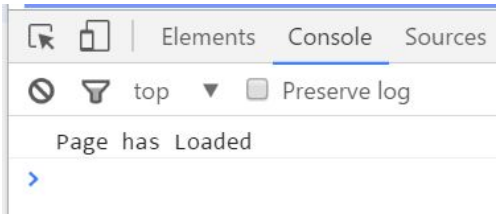
```
    <script type="text/javascript">

    </script>
</head>
```

jQuery has a special function that it calls when the page has completely loaded. This allows us to start interacting with the DOM knowing that the entire page is ready. Add the following code inside your <script> tag.

```
<script type="text/javascript">
    // Magic jQuery function
    $().ready(function() {
        console.log("Page has Loaded");
    });
</script>
```

Reload your page, open the developer console (in Chromium this is CTRL+SHIFT+I) and have a look at the console when the page first loads.



If this works, then jQuery is successfully imported and you are ready to start writing code.

Respond to Add Item Button

The first thing we need to do is get references to the correct DOM objects from our HTML page. To do this we use the **id** values from our HTML tags. We need a reference to three things.

- The to-do list <fieldset>
- The text input field
- The Add Item button

The following code will do this, make sure that the **id** fields in your HTML match the **id** values that follow the hash (#) in this code.

```
16      $().ready(function() {
17          console.log("Page has Loaded");
18          // Get hold of the UI items
19          var toDoFields = $("#toDoFields");
20          var txtNewItem = $("#newItem");
21          var cmdAdd = $("#cmdAdd");
22
23      });
```

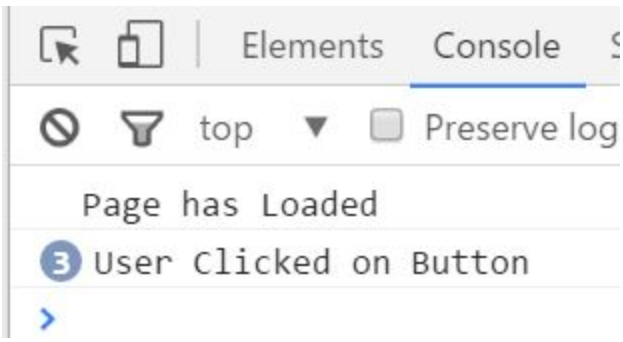
We will add a function to call when the user clicks on the button, for now just add this function.

```
23          function cmdAdd_click() {
24              console.log("User Clicked Button");
25          }
```

To register the click handler with the button, add the following code.

```
27          cmdAdd.click(cmdAdd_click);
```

Now reload the page and click the button. Observe the console in the developer window of Chromium and you should see the message appear when the button is clicked.



Add the New Item

When the user clicks the button, we should read the value of the text box into a variable. Add the following code inside our click handler.

```
23          var addItem = function() {
24              console.log("User Clicked on Button");
25
26              var newItem = txtNewItem.val();
```

We will then write another function to handle adding the item to the list. The reason I am doing this as a separate function is so that later we can add items from a server without modifying the code so much.

Add a new function **addItem** above our click handler like this. Note that this function takes an argument, which is the item to add.

```
23          function addItem(item) {
24              console.log("Adding Item " + item);
25          }
26
27          function cmdAdd_click() {
```

Now we need to build the new elements to add them to the page. We can use jQuery element creation to create a new label and a new checkbox like this:

```
23          function addItem(item) {
24              console.log("Adding Item " + item);
25
26              // Build a label
27              var newLabel = $("<label>")
28                  .text(item)
29                  .addClass("strikethrough");
30
31              // Build a new checkbox
32              var newCheckBox = $("<input>", {
33                  "type": "checkbox"
34              });
```

Now we simply add them to our page, using the reference to the **toDoFields** that we took in the last section. The code will go after the construction of the checkbox, shown here as lines 36 to 39:

```
31              // Build a new checkbox
32              var newCheckBox = $("<input>", {
33                  "type": "checkbox"
34              });
35
36              // Append the new label and a newline to our fieldset
37              toDoFields.append(newCheckBox);
38              toDoFields.append(newLabel);
39              toDoFields.append($("<br>"));
40          }
```

Finally we need to call our **addItem** function from our **cmdAdd_click** function. The **cmdAdd_click** function will look like this, line 46 shows the calling of our **addItem** function.

```
42          function cmdAdd_click() {
43              console.log("User Clicked Button");
44
45              var newItem = txtNewItem.val();
46              addItem(newItem);
47          }
```

Now reload your page and try adding items with the textbox and button. Watch the developer console for any errors. Check uppercase/lowercase on variable names and **id** values. Also watch out for punctuation during all the object creation, it can be particularly difficult in JavaScript.

If you want to put some default items on your todo list, you can hard code them at the end of the JavaScript by calling your **addItem** function a few times. Here is an example of this:

```
49          cmdAdd.click(cmdAdd_click);
50
51          addItem("Walk Dog");
52          addItem("Eat Lunch");
53          addItem("Do Washing Up");
54      });
```

Our next project will involve reading the todo items from a server.

CONGRATULATIONS - YOU HAVE COMPLETED THE INTERACTIVE TO DO LIST