

COMP3006 – Full-Stack Development

Workshop 4 – Model Answer

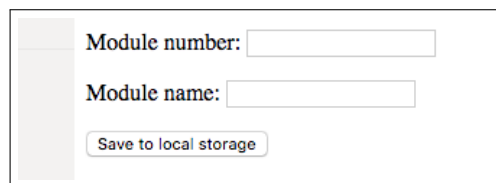
Autumn 2020

This set of model answers are intended to show possible ways of solving the unit test exercises in Workshop 4 – there are others. As stated in the workshop sheet, I have developed these answers using jQuery. If you would like feedback on a vanilla JavaScript solution, please ask during the lab session. To remind you, I have not provided a solution for exercises where you had the full code in the lab sheet, and I am not providing a model answer for the mini project or extension activities. Please attempt the exercises before looking at the answers.

1. The first exercise mostly requires you to replicate the local storage examples given in the lecture. You need to extract values from a form and save them into the local storage object, so that when you refresh the page they are automatically loaded. Here is the HTML you might use to do this:

```
<html>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="lctest.js"></script>
  </head>
  <body>
    <p>Module number: <input type="text" id="mod_num" / ></p>
    <p>Module name: <input type="text" id="mod_name" /></p>
    <button id="save">Save to local storage</button>
  </body>
</html>
```

When you load that in Chrome it will look something like this:



You now need to implement the JavaScript behaviour for the page. There are two things to do. You need to implement an event handler that fires when the button is clicked, as well as code that checks for the data in local storage and populates the form with it if it's found.

Here is the JavaScript code:

```
$(function() {
  if (localStorage.moduleNum !== undefined) {
    $("#mod_num").val(localStorage.moduleNum);
  }
  if (localStorage.moduleName !== undefined) {
    $("#mod_name").val(localStorage.moduleName);
  }

  $("#save").click(function() {
    console.log("Clicked button");
    // Extract the values from the fields.
  });
});
```

```

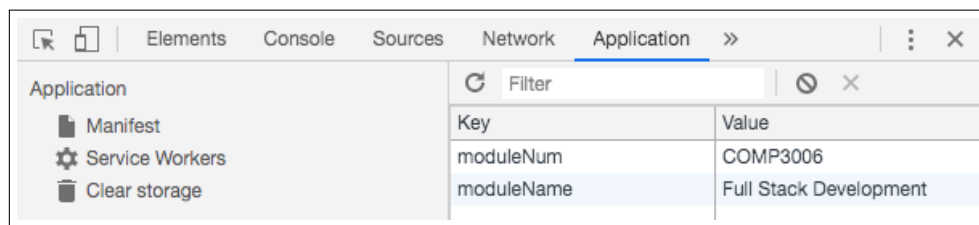
    let moduleNum = $("#mod_num").val();
    let moduleName = $("#mod_name").val();

    // Save the values into local storage.
    localStorage.moduleNum = moduleNum;
    localStorage.moduleName = moduleName;
  });
})

```

The first time you load the page the form will be empty. If you enter some data, click the button and refresh the page, you should see that the data automatically populates in the form fields.

If you want to check that this is working you can use Chrome's Dev Tools – have a look at the *Application* tab, and under storage you should see a local storage option. If you select it, your data should appear like this:



2. Most of the code for this has been provided elsewhere, but for completeness here it is all together. First, here is the HTML:

```

<html>
  <head>
    <title>Web Workers demo</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

    <script>

    </script>
  </head>
  <body>
    <p id="messagePar">Result: </p>
  </body>
</html>

```

Here is the application JS code (place this between the empty script elements:

```

$(function() {
  // Define the worker and point at the helloWorker.js file.
  let worker = new Worker("/helloWorker.js");

  // Process the response message sent by the worker.
  worker.addEventListener("message", function(evt) {
    // Extract the data from the event - this is the message,
    // put it into the page using jQuery.
    let msg = evt.data;
    $("#messagePar").append(msg);
  });
});

```

```
// Invoke the worker.
worker.postMessage("David");
});
```

Finally, here is the code for **helloWorker.js**:

```
self.addEventListener("message", function(evt) {
  // Extract the data from the event, store in a variable called
  // name, and use it to construct a message.
  let name = evt.data;
  self.postMessage("Hello " + name + " from a web worker");
});
```

3. Here is the HTML for the booking page:

```
<html>
  <head>
    <title>Seating Planner</title>
    <style>
      background-color: #00CC00; }
    .booked { background-color: #CC0000; }
  </style>

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script>

  </script>
</head>
<body>
  <h1>Seating Planner</h1>
  <div id="room">
    <div class="seat" id="seat01"></div>
    <div class="seat" id="seat02"></div>
    <div class="seat" id="seat03"></div>
    <div class="seat" id="seat04"></div>
    <div class="seat" id="seat05"></div>
    <div class="seat" id="seat06"></div>
    <div class="seat" id="seat07"></div>
    <div class="seat" id="seat08"></div>
    <div class="seat" id="seat09"></div>
  </div>
</body>
</html>
```

Here is the CSS:

```
#room { background-color: #CFCFCF; border: 1px #000 solid; width: 360px;
        overflow: hidden; padding-right: auto; margin: auto; }
.seat { width: 100px; min-height: 100px; margin: 10px; float: left;
        background-color: #00CC00; }
.booked { background-color: #CC0000; }
```

Here is the JS:

```
$(function() {
  $(".seat").click(function() {
    console.log("Clicked a seat: " + this.id);
    $(this).toggleClass("booked");

    if ($(this).hasClass("booked")) {
      localStorage.setItem(this.id, "booked");
    } else {
      localStorage.removeItem(this.id);
    }
  });
});
```

```

    }
  });

  for (i=0; i<9; i++) {
    if (localStorage.getItem("seat0" + i) !== null) {
      $("#seat0" + i).addClass("booked");
    }
  }
});

```

This code has two parts – the on click event handler activated when an element of the **seat** class is clicked, and the loop that iterates over the local storage object and identifies any booked seats.

4. This exercise requires you to build a simple UI and use a web worker to complete a computationally expensive task (bubble sorting a large array) so that you can observe the effect of the worker – the UI remains usable while sorting takes place.

Begin by implementing the UI as follows:

```

<html>
  <head>
    <title>Web Worker Example</title>

    <style>

    </style>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script>
    </script>
  </head>
  <body>
    <p id="number">Enter the number of values to generate and sort:
      <input type="text" id="numberInput" /></p>
    <button id="genAndSort">Generate and Sort</button>
    <button id="displayTime">Display the Time</button>
    <p class="status" id="status_current">Current time: </p>
    <p class="status" id="status_started">Generating numbers: </p>
    <p class="status" id="status_generated">Numbers generated: </p>
    <p class="status" id="status_sorted">Numbers sorted: </p>
    <p id="result"></p>
  </body>
</html>

```

I've included some CSS for the status paragraphs so that they show up nicely. Here it is:

```

.status {
  color: #969696;
  font-style: italic;
}

```

The model answer I've provided is for the version using the worker. Here is the JS application code:

```

$(function() {
  // Add an event handler to the button.
  $("#genAndSort").click(function() {
    // Generate the appropriate number of values.
    let number = $("#numberInput").val();
    let arr = [];
    $("#status_started").append(new Date());
    for (i=number; i>-1; i--) {
      arr.push(i);
    }
  }

```

```

$("#status_generated").append(new Date());

// Initialise the worker and point it at the sortWorker.
let worker = new Worker("/sortWorker.js");

// Process the result of sorting the numbers.
worker.addEventListener("message", function(evt) {
    // Add the result to the page.
    $("#result").html(evt.data.join(", "));
    $("#status_sorted").append(new Date());
});

// Invoke the worker.
worker.postMessage(arr);
});

$("#displayTime").click(function() {
    $("#status_current").append(new Date());
});
});

```

This instantiates the worker, defines the event handler callback function, and then posts a message to it. In this case, the message payload is the array. Here is the worker file [sortWorker.js](#):

```

// Standard implementation of the bubble sort algorithm. Note,
// this is an in-place sort.
function bubble(arr) {
    for (i=0; i<arr.length; i++) {
        for (j=i+1; j<arr.length; j++) {
            if (arr[i] > arr[j]) {
                let tmp = arr[i];
                arr[i] = arr[j];
                arr[j] = tmp;
            }
        }
    }
}

self.addEventListener("message", function(evt) {
    // Call the bubble function and pass the array, which is
    // accessed using the evt.data attribute.
    let numbers = evt.data;
    bubble(numbers);

    // Pass the result back to the the main application with
    // another message.
    self.postMessage(numbers);
});

```