

## 2 - Asynchronous Programming

### 1. Learning Outcomes

On completion of this lab you will have learned how to:

- Implement an interactive user interface in JS using event handling
- Read and process remote JSON documents using promises
- Implement a simple networked-UI using DOM API first principles

### 2. Organisation

Please complete the exercises individually.

### 3. Grading

This worksheet is worth up to 10% of your overall module grade.

You may work on this worksheet during weeks 4-6 with instructor assistance, however priority in week 4 will be given to Lab1.

### 4. Submission

The deadline for submission is Monday Oct 31, 2021 @23:59 through Brightspace.

.

- The work and submission workflow are as follows:
- Create a git repository or folder in your common git repo named lab3 or similar.
- Create a sub-folder for each problem below. problem-1, problem-2, problem-3 etc.
- Code your solution for each problem in their respective folders, committing regularly
- Include the repo link in your readme
- When you are finished developing your worksheet solution, compress and zip all problems into one zip file. Name this **<student-id>-lab-1.zip**
- **<student-id>** is something like C12345678
- Upload to Brightspace -> Week 4 -> Lab 2 -> Lab 2 - Upload

## 5. Resources

You are free to research whatever you need to solve the problems in this lab. Some recommended resources include:

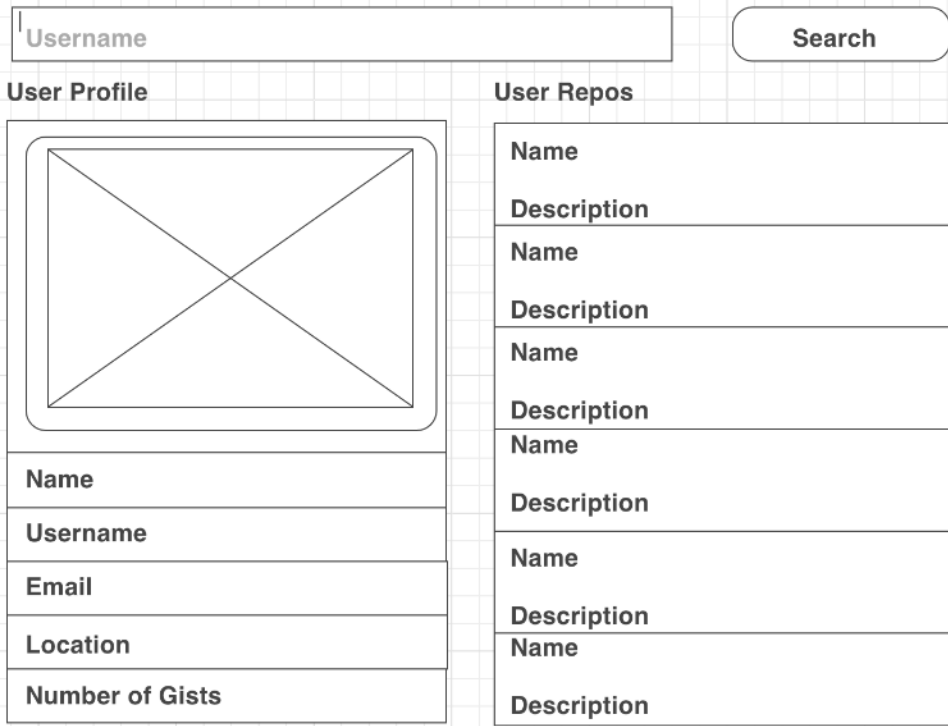
- [https://developer.mozilla.org/en-US/docs/Web/API/Document\\_Object\\_Model](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model)
- <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- <https://www.codecademy.com/learn/javascript>
- <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous>

## 6. Problem Sets

Provide Javascript code for the following problems. When submitting, clearly identify what code solutions are associated with which problems.

1	<p>Phone Directory:</p> <p>Phone Directory is a web application that allows a user to manage contacts. Once a contact is saved, it appears in a table. The list can be sorted by name. The list can be filtered by phone number.</p> <p><b>Functionality:</b></p> <p>It has three fields Name, Mobile and Email. All 3 are required fields. Clicking on the Add Contact button should add the contact to the table. Before adding a contact, the following validations should occur:</p> <p>Name - Should contain only Alphabets and Space. Should be less than or equal to 20 characters in length.</p> <p>Mobile - Should contain only Numbers. Should be equal to 10 characters in length.</p> <p>Email - Should have a proper validation and should be less than 40 characters in length.</p> <p>Show an error div with id 'error' if there is any error in input format or if there is any empty field.</p>	35 Marks
---	--	-------------

	<p>Valid contacts should get added sequentially in the table. After adding a valid contact, all fields should be reset to empty.</p> <p>Clicking on the Name heading in the table should sort it by ascending order of the contact name. Further clicks should alternately sort descending then ascending.</p> <p>The search should begin as soon as an input is typed by the user in the search field. It should filter rows based on the mobile number given in the search field.</p> <p>If there is no matching row for the search term, then the div with id 'noResult' should be made visible. It should be hidden otherwise. Odd numbered data rows should have #f2f2f2 as the background color.</p> <div style="text-align: center;"> <p><b>Phone Directory</b></p> <div style="border: 1px solid #ccc; border-radius: 10px; padding: 10px; width: fit-content; margin: 0 auto;"> <input type="text" value="Contact Name"/> <input type="text" value="Mobile Number"/> <input type="text" value="Email"/> <input type="button" value="Add Contact"/> </div> <p style="margin-top: 10px;">Search contact by Mobile No: <input type="text"/></p> <div style="margin-top: 20px;"> <p><b>Contacts Summary</b></p> <table border="1"> <thead> <tr> <th>Name</th><th>Mobile</th><th>Email</th></tr> </thead> <tbody> <tr> <td>Admin</td><td>9999999999</td><td>admin@xyzcompany.com</td></tr> <tr> <td>Yahiko</td><td>8888888888</td><td>yahiko@gmao.com</td></tr> <tr> <td>Itachi</td><td>6666666666</td><td>Ita@eee.com</td></tr> </tbody> </table> </div> </div>	Name	Mobile	Email	Admin	9999999999	admin@xyzcompany.com	Yahiko	8888888888	yahiko@gmao.com	Itachi	6666666666	Ita@eee.com	
Name	Mobile	Email												
Admin	9999999999	admin@xyzcompany.com												
Yahiko	8888888888	yahiko@gmao.com												
Itachi	6666666666	Ita@eee.com												
2	<p>For this problem refer to the lecture on array operations and consult the documentation at:</p> <p><a href="https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array">https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array</a></p> <p><a href="https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API">https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API</a></p> <p>The API you will work with is here:</p> <p><a href="http://jsonplaceholder.typicode.com">http://jsonplaceholder.typicode.com</a></p>	15 Marks												

	<p>Using this API and the <u>functional programming style of array operations</u>, provide code to solve for the problems below. You <b>MUST USE</b> this FP style with functions such as map, reduce, filter, flatmap and so on. If you find yourself using loops, you're doing it wrong. Carefully think through which data transformations you need to apply in which order in your pipeline.</p> <p>For each answer, use <b>console.log()</b> to display the data (i.e. you do not have to build DOM objects):</p> <ol style="list-style-type: none"> <li>1. List all of the <b>post</b> titles having more than six words</li> <li>2. Show a word frequency map for all of the body contents of the <b>posts</b></li> </ol>	
3	<p>You will build a web application for querying user information from the Github API that looks like the following:</p>  <p>Notes:</p> <ul style="list-style-type: none"> <li>● Use only native APIs, Javascript, HTML and CSS (no external libraries permitted)</li> <li>● You may use whatever style of async programming primitives you like such as callbacks, promises or streams.</li> <li>● The API endpoint for retrieving user information is</li> </ul>	50 marks

	<p><a href="https://api.github.com/users">https://api.github.com/users</a>.</p> <ul style="list-style-type: none"><li>● The user of your application can input a username and search in Github for that user's information</li><li>● The information should be displayed as shown, including the avatar picture at the top left</li><li>● The repo information can be obtained by following the "repos_url". You should implement a scrollable list to display these if the number exceeds 5</li><li>● Marks will be awarded for code quality such as formatting and how <a href="#">DRY</a> your code is</li></ul>	
--	---	--