Lab 05: JavaScript Functions, Local Storage & DOM

Objective(s):

1. Learn Higher Order Function
2. Learn Closures
3. Learn Local Storage
4. Learn forEach, Map, Filter & Reduce
5. Learn Some, Every, Find, FindIndex
6. Combining Iterators
7. Learn Basics of Document Object Model

Lab Task(s):

Exercises

1. Write a function called **`countdown`** that accepts a number as a parameter and every 1000 milliseconds decrements the value and console.log it. Once the value is 0 it should log "DONE!" and stop.
2. Write a function called **`isEven`** which takes in a number and returns true if the number is even and returns false if it is not

```
isEven(2); //
true
isEven(3); //
false
```

3. Write a function called **`isOdd`** which takes in a number and returns true if the number is odd and returns false if it is not

```
isOdd(3); // true
isOdd(14); //
false
```

4. Write a function called **`isPrime`** which takes in a number and returns true if the number is a prime number (is greater than 1 and can only be divided in whole by itself and 1), otherwise returns false

```
isPrime(8); // false
isPrime(17); // true
```

5. Write a function called **`numberFact`** which takes in a number and a callback and returns the result of the callback with the number passed to it

```
numberFact(59,isEven); // false
numberFact(59,isOdd); // true
numberFact(59,isPrime); // true
```

6. Write a function called find. It should take in an array and a callback and return the first value found in the array that matches the condition.

```
find([8,11,4,27], function(val){return val >= 10}); // 11
find([8,11,4,27], function(val){return val === 5}); // undefined
```

7. Write a function called **`findIndex`**. It should take in an array and a callback and return the index of first value found in the array that matches the condition.

```
// returns 1 (index of the first value greater than or equal to 10)
findIndex([8,11,4,27], function(val){return val >= 10});
```

```
findIndex([8,11,4,27], function(val){return val === 7}); //
undefined
```

8. Write a function called **`specialMultiply`** which accepts two parameters. If the function is passed both parameters, it should return the product of the two. If the function is only passed one parameter - it should return a function which can later be passed another parameter to return the product. You will have to use closure and arguments to solve this.

```
specialMultiply(3,4); // 12
specialMultiply(3)(4); // 12
specialMultiply(3); // returns a function
```

9. For this task you will be combining your knowledge of DOM access and events to build a todo app!

   As a user, you should be able to:

   - Add a new todo (by submitting a form)
   - Mark a todo as completed (cross out the text of the todo)
   - Remove a todo

   Using localStorage, try to store your todos so that if you refresh the page you do not lose what you have added to the list! Try to also save todos that you have marked as complete!

10. Write a function called **`printFirstAndLast`** which accepts an array (of objects) and console.log a new string with the first character and the last character of each value.

```
printFirstAndLast(['awesome','example','of','forEach'])

// ae
// ee
// of
// fh
```

11. Write a function called **`addKeyAndValue`** which accepts three parameters, an array (of objects), a key and a value. This function should return the array of objects after each key and value have been added to each object in the array.

```
addKeyAndValue([{name: 'Elie'},{name: 'Tim'},{name: 'Elie'}],
"isInstructor", true)

/*
[
    {
        name: 'Elie',
        isInstructor: true
    },
    {
        name: 'Tim',
        isInstructor: true
    },
    {
        name: 'Elie',
        isInstructor: true
    }
]
*/
```

12. Write a function called **`valTimesIndex`** which accepts an array of numbers and returns a new array with each value multiplied by the index it is at in the array:

```
valTimesIndex([1,2,3]) // [0,2,6]
valTimesIndex([5,10,15]) // [0,10,30]
```

13. Write a function called **`extractKey`** which accepts two parameters, an array of objects, and the name of a key and returns an array with just the values for that key:

```
extractKey([{name: "Elie", isInstructor:true},{name: "Tim",
isInstructor:true},{name: "Matt", isInstructor:true}], "name")

// ["Elie", "Tim", "Matt"]
```

14. Write a function called **`filterLetters`** which accepts an array of letters and returns the array of occurrences of a specific letter. This function should be case insensitive.
15. Write a function called **`filterKey`** which accepts two parameters, an array of objects, and the name of a key and returns an array with only those objects which have truthy values for that key:

```
filterKey([{name: "Elie", isInstructor:true, isHilarious:
false},{name: "Tim", isInstructor:true, isHilarious: true},{name:
"Matt", isInstructor:true}], "isHilarious")

// [{name: "Tim", isInstructor:true, isHilarious:true}]
```

16. Write a function called **`addKeyAndValue`** which accepts three parameters, an array (of objects), a key and a value. This function should return the array of objects after each key and value has been added. You can do this a few ways, either by reducing starting with an empty array and making copies of the object or by starting with the actual array!

```
addKeyAndValue([{name: 'Elie'},{name: 'Tim'},{name: 'Elie'}],
"isInstructor", true);

/*
[
    {
        name: 'Elie',
        isInstructor: true
    },
    {
        name: 'Tim',
        isInstructor: true
    },
    {
        name: 'Elie',
        isInstructor: true
    }
]
*/
```

17. Use the following object for this set of questions:

```
let users = [
  {
    username: 'larry',
    email: 'larry@foo.com',
    yearsExperience: 22.1,
    favoriteLanguages: ['Perl', 'Java', 'C++'],
    favoriteEditor: 'Vim',
    hobbies: ['Fishing', 'Sailing', 'Hiking'],
    hometown: {
      city: 'San Francisco',
      state: 'CA'
    }
  },
```

```
  {
    username: 'jane',
    email: 'jane@test.com',
    yearsExperience: 33.9,
    favoriteLanguages: ['Haskell', 'Clojure', 'PHP'],
    favoriteEditor: 'Emacs',
    hobbies: ['Swimming', 'Biking', 'Hiking'],
    hometown: {
      city: 'New York',
      state: 'NY'
    }
  },
  {
    username: 'sam',
    email: 'sam@test.com',
    yearsExperience: 8.2,
    favoriteLanguages: ['JavaScript', 'Ruby', 'Python', 'Go'],
    favoriteEditor: 'Atom',
    hobbies: ['Golf', 'Cooking', 'Archery'],
    hometown: {
      city: 'Fargo',
      state: 'SD'
    }
  },
  {
    username: 'anne',
    email: 'anne@test.com',
    yearsExperience: 4,
    favoriteLanguages: ['C#', 'C++', 'F#'],
    favoriteEditor: 'Visual Studio Code',
    hobbies: ['Tennis', 'Biking', 'Archery'],
    hometown: {
      city: 'Albany',
      state: 'NY'
    }
  },
  {
    username: 'david',
    email: 'david@test.com',
    yearsExperience: 12.5,
    favoriteLanguages: ['JavaScript', 'C#', 'Swift'],
    favoriteEditor: 'VS Code',
    hobbies: ['Volunteering', 'Biking', 'Coding'],
    hometown: {
      city: 'Los Angeles',
      state: 'CA'
    }
  }
];
```

a. Write a function called printEmails which console.log's each email for the users.

b. Write a function called printHobbies which console.log's each hobby for each user.
c. Write a function called findHometownByState which returns the first user which has a hometown of the state that is passed in
d. Write a function called allLanguages which returns an array of all of the unique values
e. Write a function called hasFavoriteEditor which returns a boolean if any of the users have the editor passed in
f. Write a function called findByUsername which takes in a string and returns an object in the users array that has that username

18. Write a function called vowelCount that accepts a string and returns an object with each key being the vowel and the value being the number of times the vowel occurs in the string (the order of keys in the object does not matter).
19. Write a function called removeVowels that accepts a string and returns an array of each character that is not a vowel (y should not count as a vowel for this function).
20. Write a JavaScript function to get the values of First and Last name of the following form.

```
<!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>
```

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

```
<!DOCTYPE html>
<html>

<head>
```

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

22. 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>
```

23. 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>
```

24. 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>
```

25. Given the following HTML:

```
<!DOCTYPE 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`.

26. 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.

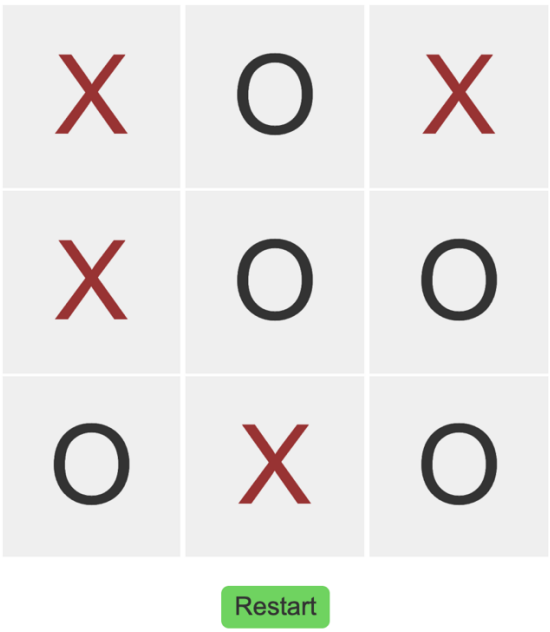
27. For this task you will be combining your knowledge of DOM access and events to build a todo app!

    As a user, you should be able to:

    - Add a new todo (by submitting a form)
    - Mark a todo as completed (cross out the text of the todo)
    - Remove a todo

    Now using localStorage, try to store your todos so that if you refresh the page you do not lose what you have added to the list! As a super bonus, try to also save todos that you have marked as complete!

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



END