2 - The DOM API

1. Learning Outcomes

On completion of this lab you will have reviewed basic HTML and CSS language features and implemented code fragments for creating and styling common web page elements in the browser.

2. Organisation

Please complete the exercises individually.

3. Grading

This worksheet is worth up to 10% of your overall module grade. You must attend and sign in at the labs in order to obtain the credit for the associated worksheets. You may work on this worksheet during lab 1 and lab 2 with instructor assistance. You must also demonstrate your submission in order to receive credit - see below.

4. Submission

The deadline for submission is Wednesday Oct 13, 2016 @23:59 through Webcourses.

5. Demonstration

You will give a brief demonstration of your submission to the lab instructor in lab 5.

6. Requirements

For this lab you will need to

- Use your own laptop with local tools or,
- Sign up for a free account with JS Bin (http://jsbin.com). It is **strongly recommended** to also have free account with Github (http://github.com) which you can use to authenticate to JS Bin. This allows you to publish Gists directly from your JS Bin workspaces which can be useful if you want to post questions or have a discussion regarding a problem or exercise in your module labs.

7. Resources

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

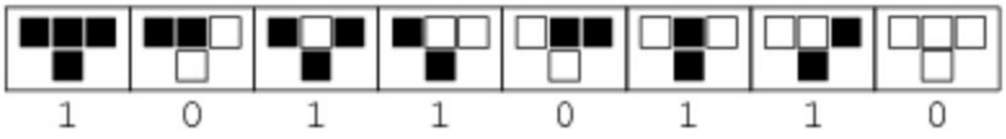
- https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model
- https://developer.mozilla.org/en-US/docs/Web/JavaScript
- https://www.codecademy.com/learn/javascript

8. Problem Sets

Provide Javascript ES6 code for the following problems using JS Bin or your own development environment as you prefer.

When submitting, clearly identify what code solutions are associated with which problems.

Note: Keep your JS Bin URLs private to you. Do not share these with class colleagues.

1	Implement the calculator front-end given in the following JS Bin in pure Javascript ES6. http://jsbin.com/gataru/edit Your solution should <u>not</u> use any HTML or CSS code in the conventional sense.	
2	Provide the Javascript ES6 to implement a 1D, 2-colour cellular automaton based on the following description and steps: a. Generate a row having 101 cells across made from `<div>` tags of size 8px by 8px. Each cell will have a randomly initialised state of 'active' or 'inactive'. If active, give it one fill color of your choice. If inactive give it another fill color of your choice b. Generate the next row of cells based on the state of the ancestors in the previous row according to the following rule  1 0 1 1 0 1 1 0 For example, **if** the direct ancestor is active **and** the direct ancestor's left sibling is active **and** the direct ancestor's right sibling is active, **then** make the corresponding cell in the new row active. The first cell in a row doesn't have a left sibling to use the row's last cell's state in this case The last cell in a row doesn't have a right sibling to use the row's first cell's state in this case	

	c. Repeat the process until you have created 50 generations (rows) of cells, with each row being created from the previous role according to the rule above. When complete you should have a 101 x 50 cell lattice of different colours. Do not use HTML or CSS code in your solution. **Please work on this yourself and do not be tempted to plagiarise a solution, even in part, from a colleague or from the Internet.**	
3		
4		