

1 - Basic HTML/CSS and 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
- Created DOM elements using the native JS API

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 a minimum of 10 labs in the semester in order to obtain full CA credit.

You may work on this worksheet during lab 1 and lab 2 with instructor assistance. You may also be requested to demonstrate your submission to the lab instructor in order to receive credit.

4. Submission

The deadline for submission is Sunday Sep 30, 2016 @23:59 through Webcourses.

5. 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>) or a similar tool such as Plunkr, Codepen or Thimble.
- 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.

6. 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/Guide/HTML>
- <https://developer.mozilla.org/en-US/docs/Web/CSS>
- 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>

7. Problem Sets

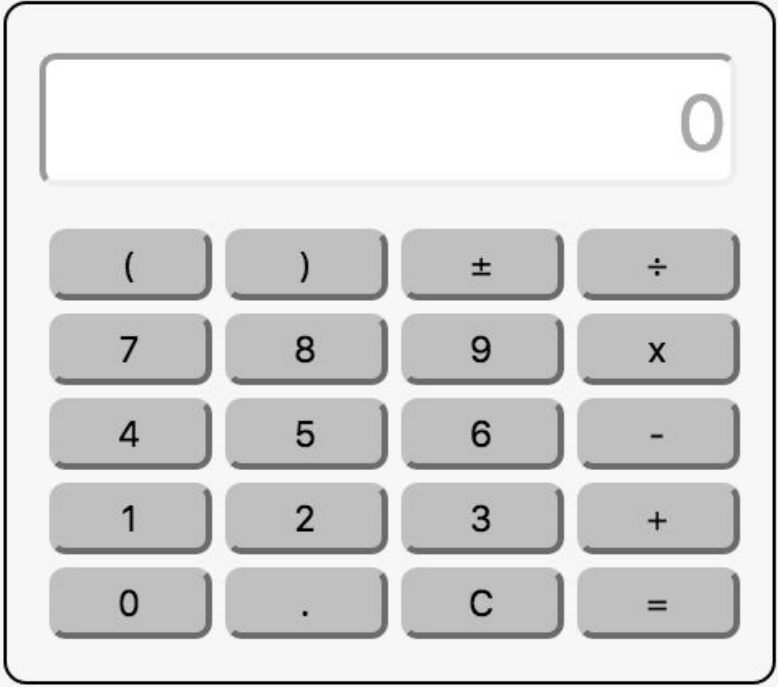
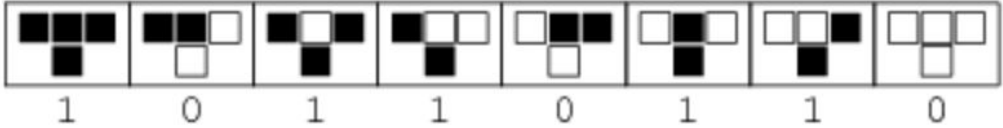
Provide code for the following problems in JS Bin or similar. Put each solution into a separate work area and share the URLs of your individual solutions.

Credit will be given for well-organised and maintainable code, so keep it [DRY](#)

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

1	<p>Given the following HTML5 code, provide the CSS code to make it layout as shown in the figure below. It doesn't have to be perfect but try to get as close as you can.</p> <p>You are <u>not allowed</u> to change the HTML5 code given, for example by adding attributes to the provided elements or by adding new elements</p> <pre> <header> <h1> Title </h1> </header> <section id="content"> <header> Article title </header> <section> <p>Article para1 ...</p> </section> </pre>	15 Marks
---	--	----------

	<pre> </section> <aside id="sidebar"> <header> Aside </header> Item 1 Item 2 </aside> <footer> <address> Kevin St, Dublin 8 </address> </footer> </pre>	
	<div> <h1>Title</h1> <div> <div> <p>Article title</p> <p>Article para1 ...</p> </div> <div> <p>Aside</p> <p>Item 1 item 2</p> </div> </div> <p>Kevin St, Dublin 8</p> </div>	
2	Provide the HTML and CSS for the this calculator front-end:	30 Marks

	 <p>Your solution should look as close as possible to the above but you are <u>not permitted</u> to use a <code><table></code> element. You are free to include embellishments of your own</p>	
3	<p>Provide the Javascript ES6 to implement a 1D, 2-colour cellular automaton based on the following description and steps:</p> <ol style="list-style-type: none"> Generate a row having 101 cells across made from <code><div></code> 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 Generate the next row of cells based on the state of the ancestors in the previous row according to the following rule  <p>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.</p> <p>The first cell in a row doesn't have a left sibling to use the row's last cell's state in this case</p>	35 Marks

	<p>The last cell in a row doesn't have a right sibling to use the row's first cell's state in this case</p> <p>c. Repeat the process until you have created 50 generations (rows) of cells, with each row being created from the previous row according to the rule above.</p> <p>When complete you should have a 101 x 50 cell lattice of different colours. Do not use HTML or CSS code in your solution. This should only be built using Javascript.</p> <p>Please work on this yourself and do not be tempted to plagiarise a solution, even in part, from a colleague or from the Internet.</p>	
--	---	--

8. Lecture Review Questions

Provide brief (1-2 paragraphs/bullets, etc) answers for the following. It is important that you have reviewed the module material in advance

1	Explain what is meant by Rich Web Application Development. Distinguish it from traditional web development	10 Marks
2	What is the Document Object Model? Explain, giving a couple of examples, how to interact with the DOM in Javascript	10 Marks