

EECS 1012: LAB #5 –JavaScript DOM (Oct 29-Nov 2, 2018)

1. Read the lab instructions in this document and take the pre-lab quiz for Lab #5 -- the TAs will not mark your lab if you do not show them you have not scored 100% on the pre-lab quiz.

Please have a look at the examples posted [here](#) for JavaScript DOM. These examples come directly from the lecture notes and will be useful for completing this lab.

2. GOALS & OUTCOMES FOR THIS LAB

- To learn and apply more JavaScript programming
- To learn how to perform unobtrusive JS
- To learn how to use the DOM

3. LAB 5– Four (4) Tasks

[Task 1] – Image src manipulation using JavaScript.

[Task 2] – Changing style using JavaScript. (JS code should be unobtrusive)

[Task 3] – Accessing multiple items using the DOM. (JS code should be unobtrusive)

[Task 4] – Creating and deleting elements. (JS code should be unobtrusive)

Further details to all tasks are provided below. Note, you are free to use the prototype library, but refrain from using JQuery (if you know it). On the 2nd in-lab test, we will give you access to the prototype library only, so learn to use it.

4. SUBMISSIONS

1) [Manual verification by a TA]

As with the previous labs, when you have completed all tasks, ask the TA to come and verify your code and output.

2) Moodle submission

You will see an assignment submission link for Lab5 on Moodle.


1) Create a **folder** named “**Lab5**” and copy all of your files into it. Compress this file and upload the compressed file to Moodle. To upload, please follow the instructions in the following video that we used for Lab 1:

<https://www.youtube.com/watch?v=stEOh6ntV5o>

TASK 1. Task 1 involves eight files: task1.html, task1.js, and six images: light_0-5.jpg. The image light_0.jpg shows 5 light bulbs all turned off. The images light_1.jpg through light_5.jpg corresponds to a light being turned on (e.g. light_2.jpg is an image of the 2nd light turned on).

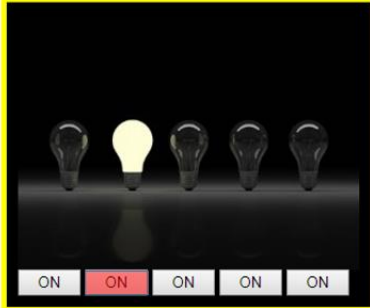
Your task it to modify task1.html to include 5 buttons as shown below. Each time a button is selected it, it should call a JS function that changes the image's src to be the corresponding image that the light "turned on". Please put your name where it says "Put your name".

Put your name



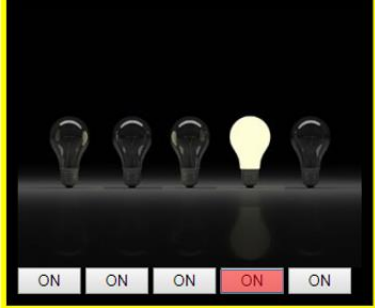
Initial image (light_0.jpg)
has no lights on.

Put your name



When this button is pressed,
the image is changed to (light_2.jpg)

Put your name



When this button is pressed,
the image is changed to (light_4.jpg)

The HTML file you are given has the following appearance (task1.html)

Put your name



Modify your HTML code to allow your JavaScript code to change the image.

You'll also need to add in the five buttons using the `<button>` `</button>` commands.

There is a CSS defined for these buttons, but you'll need to modify the width of the buttons to fit. The images width is 300 pixels.

You can solve this task anyway you like. For example, one option is to have a separate function per button. Another option would be to have a single function and pass a parameter.

For this task, it is OK if you do not use unobtrusive JS.

OPTIONAL TASK: The current task, each button's innerHTML has the text "ON". If you'd like an additional challenge (for no additional points, please don't ask) . . have your button's text change when a button is clicked. In this case, only have the text of the current light that is on say "ON", all other buttons will have the text "OFF". (*Please complete the other tasks before trying this optional functionality*).

TASK 2. You have given an HTML and JavaScript file: task2.html and task2.js. Write unobtrusive JavaScript code that adds the buttons shown below (Green, Blue, Mono, Sans Serif, Serif, Size++, Size--). Each time the button is pressed it changes the text in the box to have the style associated with the button, for example, clicking “Green” changes the text color green; clicking “Mono” changes the font to monospaced.

The default font size is 12pt. Each time “size++” is clicked, make add 1 to the current font size. Each time “Size—” is clicked, subtract one from the current font size.

Your code should be unobtrusive, i.e. you need to use the “window.onload” event to link the buttons to JS functions.

OPTIONAL: Add a “refresh” button that uses the location object to reload the page.

Put your name

Click buttons below to change this text style.

Green Blue Mono Sans Serif Serif Size++ Size--

Put your name

Click buttons below to change this text style.

Green Blue Mono Sans Serif Serif Size++ Size--

Put your name

Click buttons below to change this text style.

Green Blue Mono Sans Serif Serif Size++ Size--

Task 3: [Accessing the DOM] You are given an HTML file, task3.html that has a poem inside a `<div id="poem">`. Each line of the poem is a `<p>` tag. Modify the HTML page to add a button. When this button is clicked, you need to retrieve all the paragraphs within the div and make their background highlighted.

Write the corresponding JS code (**in an unobtrusive manner**) to link the button to a function that highlights the paragraphs when clicked. The button should act as a “toggle”, that is, if the paragraphs are already highlighted, then clicking the button unhighlight them. If the paragraphs aren’t highlighted, then clicking the button highlights them. **The button’s text should change to reflect this (see below).** You can introduce additional variables to make this work.

The diagram illustrates a toggle mechanism for highlighting text. It consists of two side-by-side panels, each representing a state of the web page. Both panels have a grey background and contain the same text: the title "How Many, How Much" in blue, the author "by Shel Silverstein" in blue, and eight lines of a poem. The poem lines are: "How many slams in an old screen door?", "Depends how loud you shut it.", "How many slices in a bread?", "Depends how thin you cut it.", "How much good inside a day?", "Depends how good you live 'em.", "How much love inside a friend?", and "Depends how much you give 'em.". In the left panel, all text is blue. In the right panel, all text is blue, but the background of each line of the poem is highlighted in yellow. Below each panel is a button. The left button is labeled "Click to highlight" and has an arrow pointing to the first line of the poem in the left panel. The right button is labeled "Click to unhighlight" and has an arrow pointing to the first line of the poem in the right panel. At the bottom center, a text box contains instructions for the button's behavior.

How Many, How Much
by Shel Silverstein

How many slams in an old screen door?
Depends how loud you shut it.
How many slices in a bread?
Depends how thin you cut it.
How much good inside a day?
Depends how good you live 'em.
How much love inside a friend?
Depends how much you give 'em.

Click to highlight

How Many, How Much
by Shel Silverstein

How many slams in an old screen door?
Depends how loud you shut it.
How many slices in a bread?
Depends how thin you cut it.
How much good inside a day?
Depends how good you live 'em.
How much love inside a friend?
Depends how much you give 'em.

Click to unhighlight

Add the following button.

When clicked, this button should call a function that retrieves all paragraphs in the `<div id="poem">` element and set their background color to yellow.

The button should act as a “toggle”. When the paragraphs are highlighted, clicking un-highlights them. Clicking again makes them highlighted again. **Note that the button’s text changes.**

Task 4: [Creating, Add, and Deleting DOM Nodes] You are given an files: task4.html and a task4.js.

An empty <div id="output"> has been added in the HTML page. Your code should do the following:

(1) Modify your JS code such that when the button "ADD" button is clicked it will create a new paragraph and add it to the <div> output. The contents of the paragraph should come from the text area that is below the [ADD] button. See example below. **Your JS code should be unobstrusive.**

(2) If the "delete" button is pressed you need to delete a paragraph. The easiest is to delete the first paragraph in the <div>, however, see if you can delete the last paragraph instead (there isn't an example this in the notes).

(3) If the user tries to delete when there are no paragraphs, create an "alert" (see example below).

TASK 3 - Creating, Appending and Deleting Nodes in the DOM Tree

Type in text below, click add to add as paragraph.

And one more.

Added Paragraphs

Here is a new paragraph.

Yet another paragraph.

And one more.

(1) When the user clicks "ADD", a paragraph with the inputted text is added to the end below.

TASK 3 - Creating, Appending and Deleting Nodes in the DOM Tree

Type in text below, click add to add as paragraph.

And one more.

Added Paragraphs

Yet another paragraph.

And one more.

(2) When the user clicked "Delete" remove the paragraph. Delete the "top" paragraph on DIV.

OPTIONAL: instead of removing the first paragraph, remove the most recent paragraph added (i.e. the last paragraph).

TASK 3 - Creating, Appending and Deleting

Type in text below, click add to add as paragraph.

Added Paragraphs

(3) If the user clicks "delete" but there are no paragraphs in the <div>, create an "alter" box that says "No paragraph to delete!"