2017 March 16

EECS 1530 Computer Use: Programming
Report 4 Specification

Due: Wednesday, April 5, 5:00 pm
Where and how: see Section 1.3

1 Main points

Be sure to read and follow all the guidelines in the **On Academic Honesty** web page; link in the www home page for the course.

1.1 Learning objectives

- Use Javascript to modify the DOM for web pages.
- When modifying the DOM model for the page, you may only use the DOM functions in the exam-reference sheet (see *Detail* for week 12) – do not use **innerHTML** or equivalent.

1.2 To hand in

There is nothing to hand in. All your work is to be electronically submitted.

1.3 Electronic submission

Before the deadline, submit the following files that you develop for this report. An automatic program will close the submission directory at the due time by the clock in the computer system.

- **report_4.js**

To submit your files use the submit web app (link *File submission* on the course home page). The app requires you to login with your Passport York account.

`https://webapp.eecs.yorku.ca/submit/`

It is strongly recommended that you periodically submit files before the deadline to be sure that something has been submitted; computers and networks can fail.

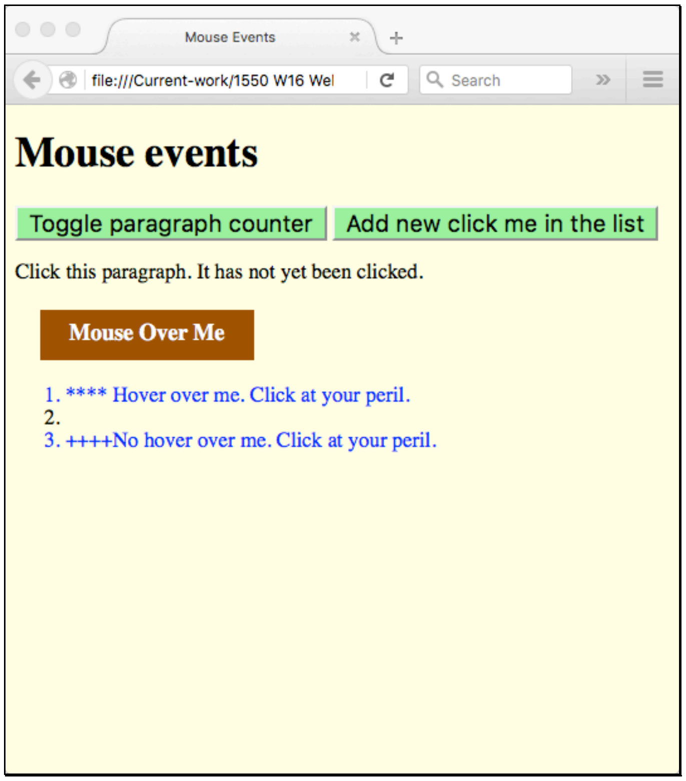
1.4 To get started

Download the archive file **events_mouse.html** from the announcement forum in Moodle – you do not modify nor submit this file.

Create the empty file **report_4.js.** When you load the file **events_mouse.html** into your browser, it automatically loads the file **report_4.js** file.

2 The mouse events

To start load the file **events_mouse.html**; it is a variation of the **Events_mouse_1.html** example – it has two buttons added. Figure 1 shows the page. At this point the page works as in the example, and as discussed in class.



Mouse Events

file:///Current-work/1550 W16 Wel

Search

Mouse events

Toggle paragraph counter

Add new click me in the list

Click this paragraph. It has not yet been clicked.

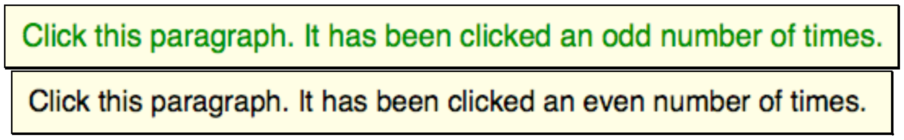
Mouse Over Me

1. **** Hover over me. Click at your peril.
2.
3. ++++No hover over me. Click at your peril.

Figure 1: The mouse events starting page

2.1 Mouse click count

In Figure 1, if you click the paragraph under the buttons the paragraph alternates in style and content as shown in the two lines in Figure 2.

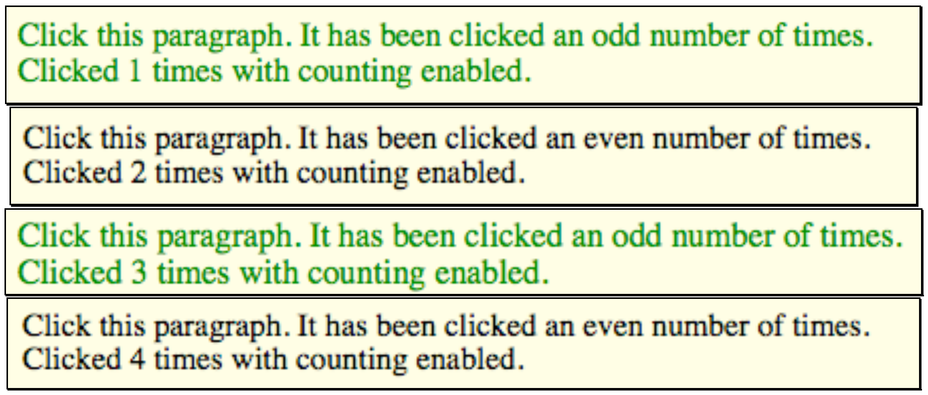


Click this paragraph. It has been clicked an odd number of times.

Click this paragraph. It has been clicked an even number of times.

Figure 2: Clicking the paragraph alternates between the two lines.

Create the function **toggleParagraphCounter** to add a counter to the paragraph so that when you click the button *Toggle paragraph counter* and then click on the paragraph, you will get a count of the number of times the paragraph has been clicked as a second line in the paragraph, as shown in Figure 3.



Click this paragraph. It has been clicked an odd number of times.
Clicked 1 times with counting enabled.

Click this paragraph. It has been clicked an even number of times.
Clicked 2 times with counting enabled.

Click this paragraph. It has been clicked an odd number of times.
Clicked 3 times with counting enabled.

Click this paragraph. It has been clicked an even number of times.
Clicked 4 times with counting enabled.

Figure 3: The paragraph changing for the first four clicks.

To achieve this result you need to do the following.

- Add a global variable *doclickCount* that is true or false depending upon whether the *Toggle* button has been pressed an even or odd number of times
- Add a global variable *clickCounter*.
- If the *Toggle* button has been pressed an even number of times then set the *onclick* attribute for the paragraph to the original function, and toggle *doClickCount* to false.
- If the *Toggle* button has been pressed an odd number of times then set the *onclick* attribute for the paragraph to a new function that you create, and toggle *doClickCount* to true.
- The new function has to do three things
  - Increment the click counter
  - Call the old *clickMe* function
  - Append the new line to the paragraph

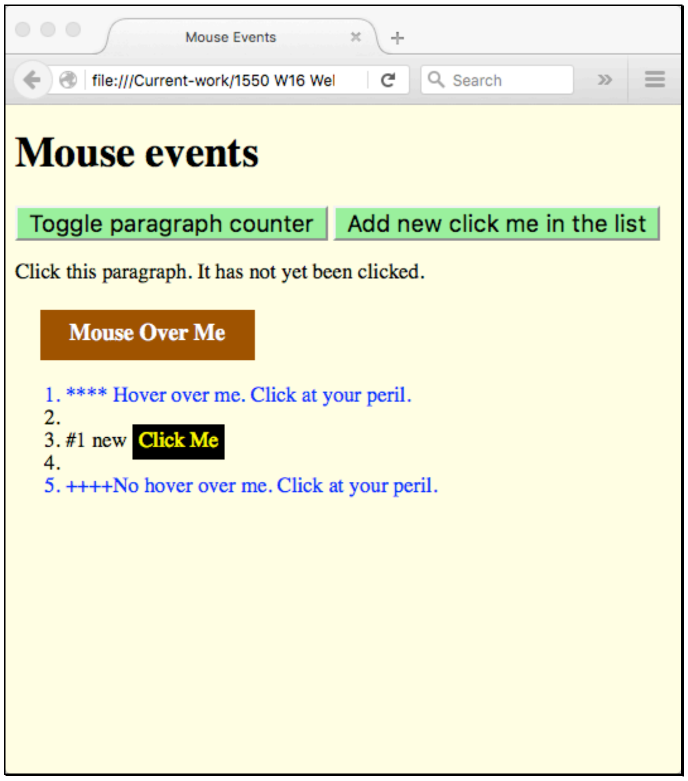
2.2 New click me

You are going to modify the list by adding a new "click me" list item, which when clicked, does a variation of what happens when you click on list items 1 and 3 in Figure 1.

2.2.1 Insert a new clickable item to the list

Create the function **addNewClickMeToList** to modify the list as seen in Figure 1, to look like the list as seen in Figure 4 when the button *Add new click me in the list* is clicked. Notice that the new item has empty list items above and below it. Figures 5 shows what the list looks like after the next two clicks on the *Add* button. The exercise *To Do List* done in class is a similar example what needs to be done.

If the task is done correctly each new *Click* Me list element has two children (1) the text "#1 new ", and (2) the SPAN element "Click Me".



Mouse events

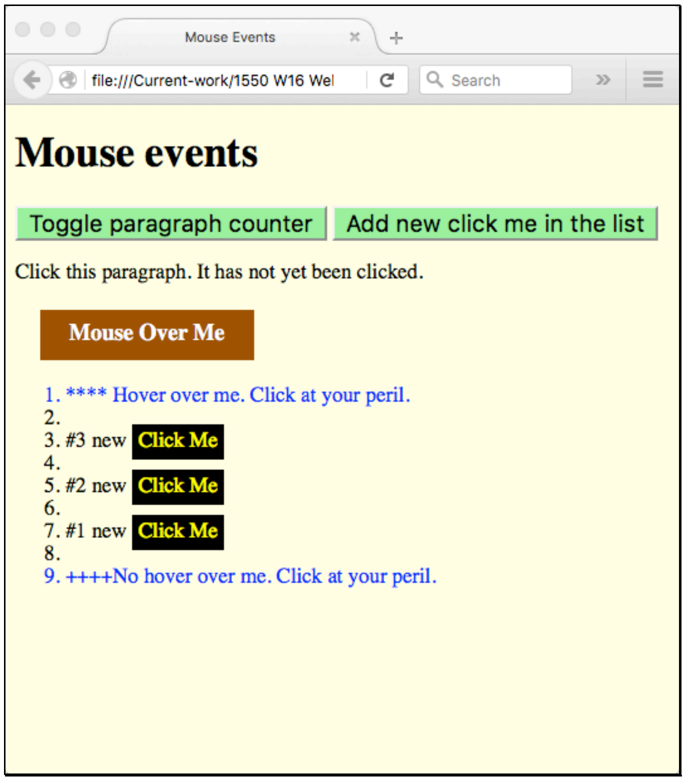
Toggle paragraph counter | Add new click me in the list

Click this paragraph. It has not yet been clicked.

Mouse Over Me

1. **** Hover over me. Click at your peril.
2.
3. #1 new Click Me
4.
5. ++++No hover over me. Click at your peril.

Figure 4: The first new “click me” item added to the list.



Mouse events

Toggle paragraph counter | Add new click me in the list

Click this paragraph. It has not yet been clicked.

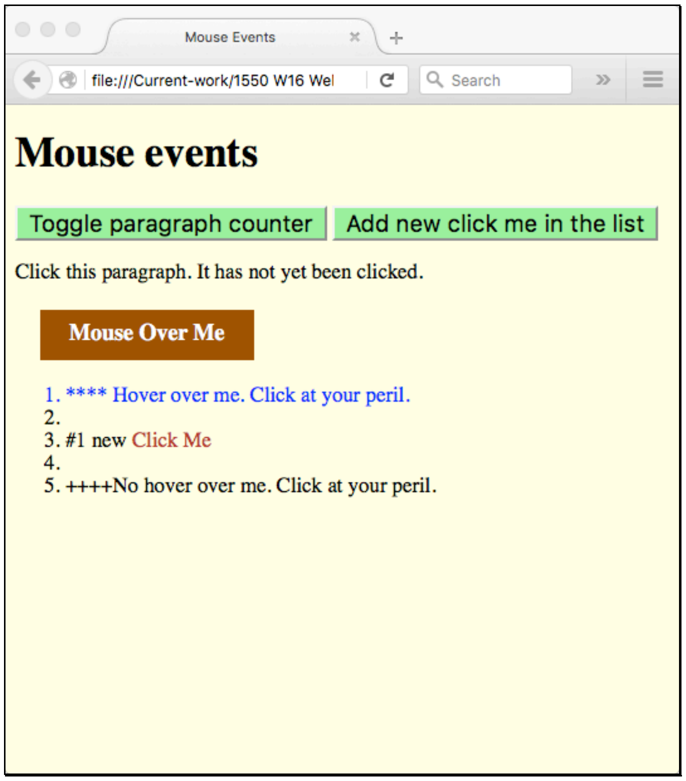
Mouse Over Me

1. **** Hover over me. Click at your peril.
2.
3. #3 new Click Me
4.
5. #2 new Click Me
6.
7. #1 new Click Me
8.
9. ++++No hover over me. Click at your peril.

Figure 5: The page after the next two new “click me” items have been added to the list.

2.2.2 Change “Click Me” style

Add the attribute *onMouseOver* to the *Click Me* so when the mouse hovers over it, it changes as shown in Figure 6; which is same thing as when the mouse hovers over list elements 1 and 5 in Figure 6.


Mouse Events
file:///Current-work/1550 W16 Wel
Search
Toggle paragraph counter
Add new click me in the list
Mouse Over Me

Mouse events

Click this paragraph. It has not yet been clicked.

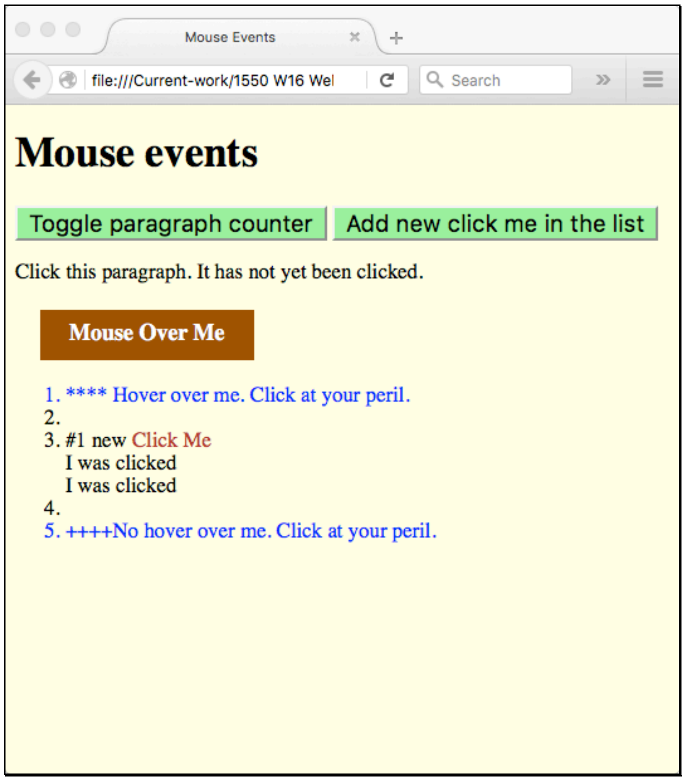
1. **** Hover over me. Click at your peril.
2.
3. #1 new Click Me
4.
5. ++++No hover over me. Click at your peril.

Figure 6: “Click Me” style with mouse hovering over it.

2.2.3 Make “Click Me” add HTML to the page

Set the attribute *onClick* of the “Click Me” element to the new function *newLIclick(this)*, and create the new function *newLIclick(whoWasClicked)*. The function body is a variation of the function *iWasClicked* in the file **events_mouse.html**, where the *newLIclick* function inserts the new HTML elements at the end of the “Click Me” list element, and not at the end of the document.

Figure 7 shows the result of clicking twice on “Click Me” with the mouse still hovering over “Click Me”.



Mouse Events
file:///Current-work/1550 W16 Wel
Search

Mouse events

Toggle paragraph counter
Add new click me in the list

Click this paragraph. It has not yet been clicked.

1. **** Hover over me. Click at your peril.
2.
3. #1 new Click Me
   I was clicked
   I was clicked
4.
5. ++++No hover over me. Click at your peril.

Figure 7: “Click Me” clicked twice with the mouse still hovering over it.

3.2.4 Make “Click Me” remove the added HTML

When you click on Items 1 and 5 in Figure 7, new elements are added to the end of the document but when the mouse stops hovering over those list elements the items are removed from the page. Do the same with “Click Me”, so that when the mouse moves off “Click Me” the added HTML disappears. The result takes you back to Figure 4. This is a variation of function *mouseOff* in the file **events_mouse.html** in that you point to the parent of “Click Me” and then remove all children at the end of the list of children until you have only original children left.

5 Grading Scheme

- Toggle paragraph counter – 15%
- Insert clickable item – 15%
- Change *Click Me* style – 10%
- *Click Me* add to HTML page – 15%
- *ClickMe* remove from HTML page – 15%
- JSLint – up to 15% – Depends upon how much of the tasks you have done.
- Comments – up to 15% – Depends upon how much of the tasks you have done.

5 Suggestions for doing the tasks

1. Develop your Javascript incrementally.
2. Look at the Console when loading the page, as many syntactic errors are detected.
3. Use console.log(…) to display the contents of variables and expressions, you will see whether or not you have the object you want or think you have. **However**, remove all your test console.log program statements from the file that you submit.
4. Use the Inspector to see the structure that you are developing. It gives a lot of information.
5. Use jsLink periodically to ensure you are using the recommended syntax. It is best to develop habits that will keep you out of trouble because the non-recommended syntax can lead to problems.