CS 2033

Multimedia and Communications

Lab 06: Use JavaScript to change styles and content; add interactivity to a website.

- Website Development -

REMEMBER TO BRING YOUR MEMORY STICK TO EVERY LAB!

JavaScript

JavaScript is the third and final component of the trinity of web languages, along with HTML and CSS. The advantage of JavaScript is its ability to dynamically change website styles and content, and to perform actions in response to user input. These features allow us to make interactive and modern websites. HTML and CSS on their own provide very little user interactivity so JavaScript is an important language in bridging this gap.

In this lab you will learn how to add JavaScript (JS) to a website, access an HTML elements from JS, change the content of an element, change the CSS styles of an element, creating functions, conditionals, loops, and attaching functions to user inputs (like clicking a button).

EXERCISE 1: ADDING JAVASCRIPT

Alert Box

1. Navigate to your USB drive folder (F:) and into the ***cs2033*** subfolder (should have been created in a previous lab).
2. Create a folder called ***lab06*** inside ***cs2033***.
3. Open Brackets and create a new file (click File > New).
4. Start by saving the new file into your ***Lab06*** > ***exer1*** folder and name the file index.html
5. Add the basic webpage shell code (refer to Lab 4 for that code).
6. Now we want to begin adding simple JavaScript to our site. Much like CSS, JavaScript can be added in 3 different ways: external (in its own file), internal (within the head), and inline (within HTML tags). However, internal JavaScript can also be added within the body of the webpage instead or in addition to the head section.
7. For now we will just use internal styles. Within the body tag, add the following HTML lines:
   <script type="text/javascript">
   </script>
8. This script tag indicates where JavaScript code will go, so add a few empty lines between this opening and closing script tag to make room for the JS code you are about to add.
9. First we'll use a built-in JS function called **alert** which is used to make a little popup text window. Add the following line of JS code between the script tags:
   alert("Welcome to my site!");
10. Save the file and open it in Chrome. Notice that the alert box is in front of the site and has focus. This means it must be closed before the rest of the site can be loaded or accessed.
11. As you may have noticed already, the alert box can be annoying to users because of its focus in front of the website so they should only be used if necessary. Comment out this line of JS by typing // at the start of the line. Commenting it out will stop it from being executed

but it's still there for reference. Your code should now look like this:

```
<!DOCTYPE html>

<html>
<head>
</head>

<body>

    <script type="text/javascript">
        //alert("Welcome to my site");
    </script>

</body>

</html>
```

Variables

12. Variables are a fundamental part of programming as they allow us to store a value for later use (or immediate) use. To create a variable, use the statement **var** followed by a name of your choosing, then use an equal sign and assign a value to it (Note that text must be surrounded by quotation marks). Under the commented out alert line, add a new line and enter the following code, replacing "**Bryan**" with your own name.
    var name = "**Bryan**";
13. This line created a variable but if you were to save and reload Chrome now, nothing would happen because we're not using the variable in any way. Add another line after this variable line (it must be AFTER the variable has been created – order is important here!):
    alert("Welcome, " + name);
14. The above line should create an alert box that says "Welcome, **Bryan**" with the name you assigned to the variable *name*. Save the file and refresh the browser to confirm this step.
15. Comment out this alert box line like you did with the previous line in step 11, but leave the variable creation line untouched so that we can still use the name variable below.

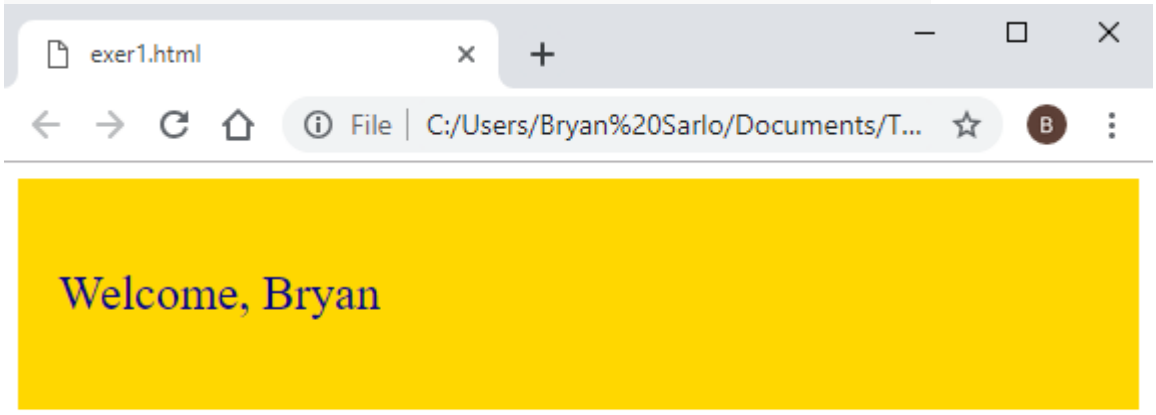
Accessing and changing HTML elements

16. First we want to create an HTML element so add some empty lines within the body tag but above the script tag (in between <body> and <script type="text/javascript">).
17. In the space you just created, add the following line to create an HTML div:
    Welcome guest
18. Notice that we set the ID of this div, and that's because we want to use JS to access it in the upcoming steps. The easiest way to access HTML elements from JS is from the ID. To do this, we will use the function **getElementById**. Add the following lines in the JS portion, AFTER the variable *name* is created:
    var bannerDiv = document.getElementById("banner");
    bannerDiv.innerHTML = " Welcome, " + name + " ";
19. The first of these two lines is getting/accessing the HTML element that we created above with the ID "banner" and storing it into a new variable called bannerDiv. The second line is using that variable to change the content within that banner div. We use the property *innerHTML* to change the content and, as its name suggests, it supports HTML directly in the content. We are setting the content to a paragraph that says "Welcome," followed by your name, as specified in the *name* variable created earlier. Save and reload to see this.
20. In addition to changing the content of an element, JavaScript also allows us to change the CSS styles of an element. This is done by accessing the *style* property of elements and then

individual styles within the styles. Let's start by changing the background colour:
bannerDiv.style.backgroundColor = "gold";

21. Save the file and refresh the browser to see the banner change to gold.
22. Now we'll change the font colour, font size, and padding on the same div:
bannerDiv.style.color = "darkblue";
bannerDiv.style.fontSize = "24px";
bannerDiv.style.padding = "20px";
23. Save the file and refresh the browser to see the changes. The images below show you what your code and website should look like.

```
<!DOCTYPE html>

<html>
<head></head>
<body>

    <div id="banner"><p>Welcome guest</p></div>

    <script type="text/javascript">
        //alert("Welcome to my site");
        var name = "Bryan";
        //alert("Welcome, " + name)

        var bannerDiv = document.getElementById("banner");
        bannerDiv.innerHTML = "<p>Welcome, " + name + "</p>";
        bannerDiv.style.backgroundColor = "gold";
        bannerDiv.style.color = "darkblue";
        bannerDiv.style.fontSize = "24px";
        bannerDiv.style.padding = "20px";
    </script>

</body>

</html>
```



EXERCISE 2: FUNCTIONS, CONDITIONALS, AND LOOPS

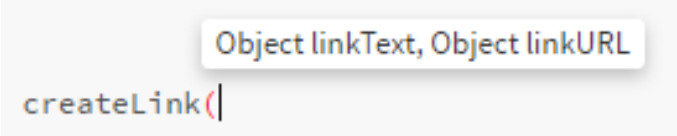
You probably noticed, most JavaScript lines end with a semicolon (;). This is mostly true, but there are lines in certain code structures that end with a curly bracket ({ or }). In this exercise you will learn about functions, conditionals, and loops.

Functions are routines or processes that can be executed any number of times. They come in handy when you want to perform certain actions from more than one spot in the code and to simplify the code by cutting out repetitive commands. **Conditionals** are structures in which certain lines of code only execute under specific conditions. We use if-else statements to perform such condition-based branching. **Loops** allow us to run the same line(s) of code repetitively in sequence which is very useful especially when you want to perform the same action on several elements.

Functions

1. Open http://cs2033.gaul.csd.uwo.ca/~bsarlo/Lab06/, download ***exer2_shell.html*** and copy it into ***lab06***.
2. Open exer2_shell.html in Brackets and immediately save it as exer2_complete.html. All work for this exercise should be done in exer2_complete.html.
3. Open exer2_complete.html in Chrome as well. You should see a dark red bar, a gold bar, and a light blue bar. We will use JS to add links and text into these three divs.
4. Click inside the JavaScript area (between the opening and closing script tags). Within this area, create a function definition using the following lines of JS code:
   function createLink(linkText, linkURL) {

   }
5. The function we just created is named createLink and takes in 2 parameters to execute. The function is just a shell for now though, so we need to add code within it (between the curly brackets { and }.
6. First, we will use JS to create a new anchor (link) element. Add the following line between the function's curly brackets:
   var newLink = document.createElement("a");
7. Now we will access the attributes of this new link to set its CSS class, link text, and href path. The following 3 lines should be added into the function after the newLink variable line:
   newLink.className = "link";
   newLink.innerHTML = linkText;
   newLink.href = linkURL;
8. Notice that the className attribute is always set to "link" so that all links are given the same CSS styles, but the innerHTML (link text) and href (link path) are given variable values which come from the input parameters in the function definition: linkText and linkURL. This means that whatever we pass in to the function will be used in the link we create. This parameterization gives functions a great amount of flexibility.
9. The last line to include within this createLink function is to actually add this newLink into the navbar div (the darkred bar along the top). The following line does this:
   document.getElementById("navBar").appendChild(newLink);
10. Now our function is complete. Save the file and refresh the browser. Nothing changes? That's ok! That's because we just created the function but haven't called (executed) it yet. To call the function, we type the name of the function followed by parentheses and include values for any input parameters; linkText and linkURL in this case. Brackets reminds you of the parameters as you start typing:



11. Below the end of the function (after the } that closes the function) but still within the script area, add the following line to call the function to add a Home link:
    createLink("Home", "index.html");
12. Save the file and refresh the browser and you should now see the Home link show up in the dark red navbar at the top.
13. Repeat step 11 but this time it should say "About" and link to "about.html".
14. Repeat step 11 but this time it should say "Portfolio" and link to "portfolio.html".
15. Repeat step 11 but this time it should say "Contact" and link to "contact.html".
16. Save the file and refresh the browser and you should see the 4 links side by side in the dark red navbar. Note that we did not create these pages so clicking the links won't work, but if the files existed, they would open right away just like links made directly in HTML. The JS code for this exercise should now look like this:

```
<script type="text/javascript">

    function createLink(linkText, linkURL) {
        var newLink = document.createElement("a");

        newLink.className = "link";
        newLink.innerHTML = linkText;
        newLink.href = linkURL;

        document.getElementById("navBar").appendChild(newLink);
    }

    createLink("Home", "index.html");
    createLink("About", "about.html");
    createLink("Portfolio", "portfolio.html");
    createLink("Contact", "contact.html");


</script>
```

Conditionals

Conditionals are useful tools in programming to change which portions of code are executed depending on certain conditions. In this part of the exercise, we will use JS to retrieve the current day of the week and then display a message depending on which day it is. Because the message depends on the current day, you will only see one message during your lab slot. To confirm that it works for other days, you should try opening this completed file other days on your own computer.

17. Beneath the function calls from the above portion, but still within the JS area, create a new function called displayGreeting which won't take in any input parameters:
    function displayGreeting() {

    }
18. We will start by getting the current day of the week using the built-in Date function. Note that the day of the week comes as an integer from 0 (Sunday) to 6 (Saturday). Add the following 2 lines to get the day integer within displayGreeting function.
    var today = new Date();
    var day = today.getDay();
19. Just below these lines, still inside the function, we will create a new variable called msg will start as an empty string (text). It will be changed in the upcoming conditionals.
    var msg = "";
20. Now we are ready to add the conditionals that check the value of the current day and change msg as a result. Note that you need the double equal signs == when checking if something equals something else.

21. Start with the first if-statement to check if the day is Sunday:
```
if (day == 0) {
    msg = "No school today… Sleep in!";
}
```
22. In this conditional, the msg text is changed only if the current day is Sunday (day == 0). Now we need to add follow-up conditionals. This is done with the **else if** statement. Put your cursor on the right side of the closing curly bracket } from above, and add the following code, shown in bold:
```
} else if (day == 1) {
    msg = "Ugh, I hate Mondays";
}
```
23. At this point, the displayGreeting function should look like this:
```
function displayGreeting() {

    var today = new Date();
    var day = today.getDay();

    if (day == 0) {
        msg = "No school today… Sleep in!";
    } else if (day == 1) {
        msg = "Ugh, I hate Mondays";
    }



}
```
24. Repeat step 22 but check instead for day == 2 (Tuesday) and change the msg to something new that is specific to Tuesday.
25. Continue repeating this process until you have checked each day from 0 to 6 and set a different, appropriate message for each day. After you have them all included, your code should look something like the below picture, but you don't have to use the same messages I used! Also, I added comments beside each if or else-if statement to note the day which you didn't have to do.
```
function displayGreeting() {

    var today = new Date();
    var day = today.getDay();

    if (day == 0) { // Sunday
        msg = "No school today… Sleep in!";
    } else if (day == 1) { // Monday
        msg = "Ugh, I hate Mondays";
    } else if (day == 2) { // Tuesday
        msg = "Tuesdays are too long";
    } else if (day == 3) { // Wednesday
        msg = "It's hump day";
    } else if (day == 4) { // Thursday
        msg = "Getting close to the weekend";
    } else if (day == 5) { // Friday
        msg = "TGIF!";
    } else if (day == 6) { // Saturday
        msg = "No school today... I love Saturdays";
    }



}
```

26. Now we are almost done this function, but we just need one more line of code to add this text to the content div. Below all the conditionals, but still within the function, add the line:
    document.getElementById("msgBox").innerHTML = msg;
27. Now the function is complete but once again nothing will show up from this if we save and refresh the browser now. We need to call this function to make it execute. Below the closing bracket of the function }, add the following function call line:
    displayGreeting();
28. The screenshot below shows what the bottom of your function should look like (the conditionals haven't been touched here, they are just shown again as a reference point)

```
        } else if (day == 5) { // Friday
            msg = "TGIF!";
        } else if (day == 6) { // Saturday
            msg = "No school today... I love Saturdays";
        }

        document.getElementById("msgBox").innerHTML = msg;

    }

    displayGreeting();
```

29. Save your file and refresh the browser and you should now see a message corresponding to the current day of the week. Again, I recommend re-opening this webpage on other days from your own computer to see it display a different message. Isn't this fun?!

Loops

Another handy structure is the loop which allows us to repeat lines of code numerous times. This also helps to run code on multiple elements in sequence. In this portion of the exercise we will create 2 functions with loops. One will repeat a line of text several times, and the other will make a slight CSS change to each of the 4 links we created earlier.

30. Below the displayGreeting code from above, but still within the JS area, create a new function called repeatText which will not take in any input parameters:
    function repeatText () {

    }
31. Within this function, create a variable called *text* and initialize it to an empty string:
    var text = "";
32. Next we will create a for-loop. For-loops consist of 3 parameters: variable initialization, condition, and increment. This means we have to indicate our starting value, ending value, and how much it should increment by in each loop iteration. For this one, we will start our variable *i* at 0, loop while it's less than 5, and increment by 1 each time. Add the following code for this loop into the repeatText function:
    for (var i = 0; i < 5; i++) {

    }
33. Between the curly brackets { and } of this loop, append the following line to add a short poem to the text variable:
    text += " No more worry, no more stress. I'm having fun, learning JS! ";
34. Notice that we used the += operator above which means the poem is being appended to the current variable. Using this within the loop means the line is added 5 times in a row.
35. After the loop, write the following line to add this text to the blue bar:
    document.getElementById("content").innerHTML = text;
36. Below the repeatText function, we need to include a call to this function so that it executes:
    repeatText();

37. Save the file and refresh the browser to see the poem repeated on the webpage. Your webpage should now look something like this:

38. Now we will create another function, called loopLinks:
    function loopLinks() {

    }
39. Within this function, we will start by gathering the links from their class name:
    var links = document.getElementsByClassName("link");
40. Because there are 4 such link elements, the links variable is a list of the 4 elements so we can now use a loop to iterate over them in sequence. We use links.length in our condition to indicate how long to loop for. Add these lines for the for-loop:
    for (var i = 0; i < links.length; i++) {

    }
41. Within this loop, access each of the link items using square brackets and our variable *i* to indicate which item to modify. Since *i* will go from 0 to 4, each link will be modified in that order. The following line of code must be added within the for-loop:
    links[i].className = "link differentLink";
42. The function is now complete so we just have to call it below. Add the following line below the end of the function closing bracket }:
    loopLinks();
43. Save the file and reload the browser to see the 4 links modified with a white bottom border.

44. The code for the two loop functions should look like this:

```
function repeatText () {
    var text = "";
    for (var i = 0; i < 5; i++) {
        text += "<p>No more worry, no more stress. I'm having fun, learning JS!</p>";
    }
    document.getElementById("content").innerHTML = text;

}

repeatText();


function loopLinks () {

    var links = document.getElementsByClassName("link");

    for (var i = 0; i < links.length; i++) {
        //links[i].style.backgroundColor = "darkblue";
        links[i].className = "link differentLink";

    }

}

loopLinks();
```

EXERCISE 3: HANDLING MOUSE EVENTS

JavaScript makes it easy to receive user input from mouse or keyboard actions (clicking, mouse movement, key presses, etc.). These actions are called **events** and we use **event handlers** to capture these events in the code. The event handlers are usually attached to individual HTML elements (like divs) or the entire body of the website.

In this exercise, we will apply JavaScript event handlers to look for mouseover (hovering), mouseout (un-hovering), and click (no clarification needed 😊) on a button. The mouseover and mouseout will change the button's background colour (this is similar to the CSS :hover pseudo-class you learned previously, but it uses JS instead of CSS). When the button is clicked, we will use a random number generator to change the website's background colour randomly. After that, we will apply another mouse event called mousemove (which is triggered every time the mouse moves even a tiny amount) on the middle circle and change the circle's colour from the cursor's position.

Mouseover, mouseout, and click

1. Open http://cs2033.gaul.csd.uwo.ca/~bsarlo/Lab06/, and download ***exer3_shell.html*** and copy it into ***lab06***.
2. Open exer3_shell.html in Brackets and immediately save it as exer3_complete.html. All work for this exercise should be done in exer3_complete.html.
3. Open exer3_complete.html in Chrome as well. You should see a button in the upper-left corner and a big, red circle in the middle of the screen. These shapes are static right now meaning nothing changes when you hover over them or click them. Let's start adding JS!

4. In Brackets, click into the JS area (between the opening and closing <script> tags).
5. Create a variable, called *btn*, to hold the element with the ID "btn":
   var btn = document.getElementById("btn");
6. Then we will add event listeners for mouseover, mouseout, and click, and provide function names to be run for each event. The functions don't exist yet but we will create them in upcoming steps. The event listeners are created with the following format:
   divVar.addEventListener("*eventType*", *functionName*);
   Using this format, we will add the following 3 lines to add the handlers to btn:
   btn.addEventListener("mouseover", hoverButton);
   btn.addEventListener("mouseout", leaveButton);
   btn.addEventListener("click", changeBackgroundColour);
7. Now our event handlers are created but they won't do anything until we create the 3 functions: hoverButton, leaveButton, and changeBackgroundColour.
8. Add the function hoverButton which will change the background colour of btn:
   function hoverButton() {
      document.getElementById("btn").style.backgroundColor = "#25E5EE";
   }
9. Now you need to create the function, leaveButton, which will turn the btn's background colour back to its original colour, #DDDDFF. Use the hoverButton as a base to help you create this leaveButton function.
10. After that, create a new function called changeBackgroundColour. Within this function we will generate 3 random numbers between 0 and 255 as our red, green, and blue values and then assign that random colour to the main website background. Use the following line to generate the red value randomly:
    var red = Math.floor(Math.random() * 256);
11. Use the same code but with different variable names to create two additional random numbers for green and blue.
12. Once the three values have been generated, we will use them within the RGB colour code for the body's background. Use the following line of code for this:
    document.body.style.backgroundColor = "rgb(" + red + "," + green + "," + blue + ")";
13. Your code should now look like this:

```
<script type="text/javascript">


    var btn = document.getElementById("btn");
    btn.addEventListener("mouseover", hoverButton);
    btn.addEventListener("mouseout", leaveButton);
    btn.addEventListener("click", changeBackgroundColour);


    function hoverButton() {
        document.getElementById("btn").style.backgroundColor = "#25E5EE";
    }

    function leaveButton(btn) {
        document.getElementById("btn").style.backgroundColor = "#DDDDFF";
    }

    function changeBackgroundColour () {
        var red = Math.floor(Math.random() * 256);
        var green = Math.floor(Math.random() * 256);
        var blue = Math.floor(Math.random() * 256);

        document.body.style.backgroundColor ="rgb(" + red + ", " + green + ", " + blue + ")";
    }

</script>
```

14. Save the file and refresh the browser. Double check that each portion works: hovering over the button changes its background colour, leaving (un-hovering) reverts its background colour, and clicking the button changes the website body's background to a random colour. If any of these does not work as expected, revisit the steps above corresponding to the component in question and double check your code (Hint: programming errors often come from forgetting a semicolon or bracket!)

Mouse move

In this last portion we will trigger a function on the mouse move event on the big circle. This means that as the mouse cursor moves around within the boundaries of the circle, the function we create will be executed and it will detect the location of the cursor within that circle. We will calculate a new background colour using the mouse coordinates so it will change constantly while the mouse moves around the circle.

15. First we will apply an event listener. This could be done like the way we added the other mouse event listeners previously, however there is an easier way to add event listeners that we will use here: inline listeners. Just like inline CSS, these JS event listeners are added as attributes within the HTML. Edit the line of HTML that creates the roundBtn div as follows:
    <div id="roundBtn" **onmousemove="moveCursor(event);"**>
16. Once again we have added an event listener but the function being called (moveCursor in this case) has not yet been created. Let's create it now. Start with the function definition:
    function moveCursor (e) {

    }
17. Now we need to begin filling in the code within this function. Start by retrieving the roundBtn element using its ID:
    var circle = document.getElementById("roundBtn");
18. Next we have to obtain the (x, y) coordinates of the cursor relative to the top left corner of the circle. There are two key components for these coordinates: the absolute coordinates on the screen (e.clientX, e.clientY) and the start position of the circle (circle.offsetLeft, circle.offsetTop). We'll use simple subtractions with these values to get the (x, y) we need:
    var x = e.clientX – circle.offsetLeft;
    var y = e.clientY – circle.offsetTop;
19. The circle's background colour will be based on how close the cursor is to the center. Thus we need to calculate the horizontal and vertical distances between the mouse cursor and the circle's center. We use the absolute value to get the distance as a positive number (as a refresher, abs(5) = abs(-5) = 5). Add the following lines to the function:
    var dx = Math.abs(x – 150);
    var dy = Math.abs(y – 150);
20. Almost done! We just need to determine the colour based on the dx and dy values and then assign the colour to the circle background colour style. We'll use interpolation to determine the dx and dy values on scales of 0-255, the same range as RGB values:
    var hColour = dx / 150.0 * 255.0;
    var vColour = dy / 150.0 * 255.0;
21. There's no easy way to assign an RGB colour from 2 values (like our horizontal and vertical distances) without converting to the HSV colour model. Alternately, we'll keep this simple and just use these values asymmetrically in the RGB model by using the horizontal value for both the R and G components, and the vertical value for the B component:
    circle.style.backgroundColor = "rgb(" + hColour + "," + hColour + "," + vColour + ")";
22. Remember that if R = G = B, the colour will be on the grayscale, which is the case here if hColour = vColour. Otherwise, if hColour is 0 (i.e. the cursor is horizontally centered), the colour will be on the blue scale – darker near the vertical center and brighter blue at the top and bottom. If vColour is 0 (i.e. the cursor is vertically centered), the colour will be on the

yellow scale (because yellow = red+green) – darker near the horizontal center and brighter yellow on the left and right edges.

23. Save this file and reload the browser. Ensure that each of the scenarios explained in step 22 is true. If anything does not work properly, double check the steps above and fix your code accordingly.

This concludes this lab session. Call your TA over to check your work and receive your mark for this lab.

REMEMBER TO REMOVE YOUR MEMORY STICK FROM YOUR MACHINE AND PUT IT IN YOUR BACKPACK! (don't forget it)! ☺