CS 2033

Multimedia and Communications

Lab 07: Advanced JavaScript

- Website Development -

REMEMBER TO BRING YOUR MEMORY STICK TO EVERY LAB!

JAVASCRIPT

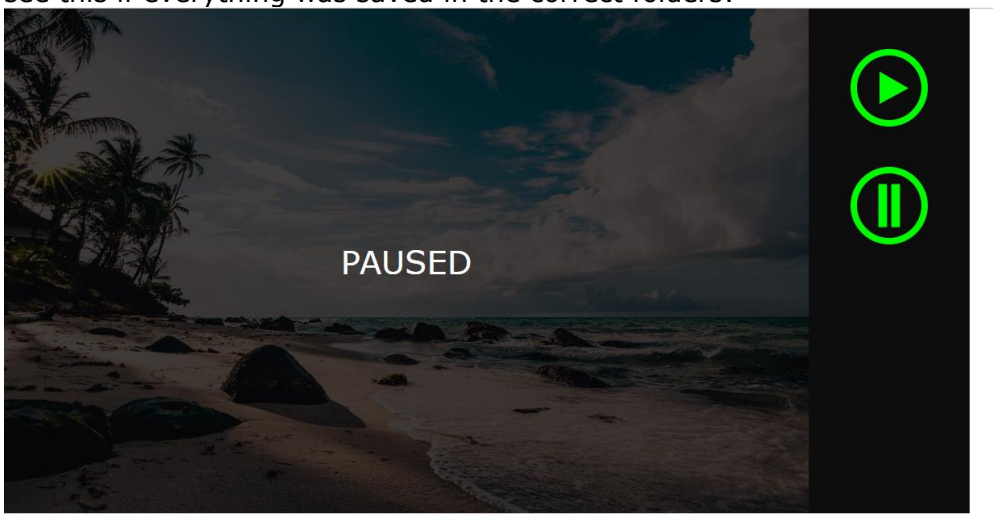
Last week you learned how to add internal and inline JavaScript into an HTML site. In that lab you created variables, conditionals (if-else statements), loops, and functions, and used these and other tools to change CSS styles and content dynamically. Mouse event handlers were used to provide user input for some of these functions.

We will build on that foundation and create fun and interactive web modules. In addition to mouse events, JavaScript supports keyboard events, timers, and other events so you will learn these in this lab as well.

EXERCISE 1: IMAGE CAROUSEL/SLIDESHOW

In this first exercise you will create an image slideshow, often called a carousel, in which the image changes every so often. One of the key tools to make this work is the JavaScript timer, specifically the interval function. This allows us to trigger a function on a regular timed interval basis. Using this, we will indicate that the image path (src) should be updated each time the function is triggered from the interval timer. In addition to the timer event, we will use mouse events to create rollovers on our pause and play buttons and to actually pause and play the slideshow when they are clicked respectively.

1. Navigate to your USB folder (F:) and into the ***cs2033*** subfolder (should have been created in a previous lab).
2. Create a folder called ***lab07*** inside ***cs2033***. Within ***lab07*** create another folder, called ***exer1***. Inside ***exer1***, create a subfolder called ***img***.
3. Open http://www.csd.uwo.ca/~bsarlo/cs2033b/labs/Lab07/exer1/, download ***exer1.html***, and save it into ***lab07*** > ***exer1***.
4. From the same site, click into ***img***, download all 7 images (5 JPG pictures and 2 PNG icons), and save them into your folder ***lab07*** > ***exer1*** > ***img***
5. Open exer1.html in Brackets and immediately save it in the same folder as ***exer1_complete.html***. All work in this exercise should be done in exer1_complete.html
6. Open exer1_complete.html in Chrome as well to see how it looks from the start. You should see this if everything was saved in the correct folders:



7. As you can see, the beach image is there with a tint (black translucent overlay) and the word "PAUSED". On the right, we have a controls panel with play and pause buttons. If you click or hover over the buttons, nothing happens because JS hasn't been added yet.
8. Click into the script tag in the head section. We need to start by adding 4 variables – these are called **global variables** because they are created outside of a function and are accessible within any function in the file. The variables we need are:
   images – this is an array of the 5 image filepaths
   index – this will keep track of which image we are currently showing, based on its position in the array (remember that list indexing begins at 0, not 1!)
   numImages – this is the number of images in our array, in our case it is 5 but rather than set it to 5, we'll use the length of the array to obtain it
   timer – this will be used when we deal with the timer later
   The code for creating these 4 variables is shown here:

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


    var images = ["img/beach.jpg", "img/autumn.jpg", "img/mystical.jpg", "img/valley.jpg", "img/bridge.jpg"];
    var index = 0;
    var numImages = images.length;
    var timer;
```

9. Next, let's create the function that will be attached to the interval timer and will use these variables to change the current image from the array. Since we want to go through the array in order, the main operation is to increment our index variable by 1 (i.e. 0 → 1, 1 → 2, etc.). However, once we reach the end of the array, our index should reset back to 0 rather than continuing to increase infinitely. By resetting it, we can cycle through the array of images seamlessly. Use a simple conditional to perform this action:

```
    function changeImage() {
        if (index < numImages - 1) {
            index++;
        } else {
            index = 0;
        }
    }
```

10. Let's just review what this function is doing before we continue on. Remember that index is going to track which image is being displayed so it will change values often, while numImages will be constantly the value of 5 because there are always 5 images in our array. Because the indices begin at 0, the last image in the array will be at position 4, not 5. This is why our if-statement checks if index < numImages – 1 (numImages – 1 is 4). For those indices (0, 1, 2, and 3), we perform the operation index++ which simply means increment its value by 1. The longform equivalent statement would be index = index + 1, but the ++ operator makes it much more succinct. Then we have an else statement which is triggered if the index is NOT less than 4 (when it is exactly 4) and in here we reset index back to 0 so that the cycle can continue.
11. Now that this indexing portion is complete, we need to add one last line within our function. This line will update the img element with a new src filepath based on the paths provided in the array. We use our newly update index value as the array position here. The entire function should now look like this:

```
function changeImage() {
    if (index < numImages - 1) {
        index++;
    } else {
        index = 0;
    }
    document.getElementById("carousel-img").src = images[index];
}
```

12. When this function is first triggered, index is 0 so the first if-statement will be executed. The index++ operation will cause index to increase to 1. Then the last line will execute so it will

change our img with ID "carousel-img". The image's src will be changed to images[1] which is "img/autumn.jpg" and that autumn picture will immediately be displayed instead of the beach image that was previously shown. The next time through is very similar. Index goes from 1 to 2 so the image then becomes images[2] or "img/mystical.jpg".

13. This function is finished now so we can test it by calling it to see what happens. If we try to call the function within the head section, it won't work because the img element hasn't been added yet. Thus we must add another script tag in the body after the elements are made.

NOTE: Remember that an element must be created BEFORE it can be modified by JavaScript, so make sure your JavaScript code is placed after the div (or other element) that is being changed.

14. Your body text should now look like this with the newly added JS at the bottom:

```
<div class="wrapper">
    <div class="carousel-box" id="a">
        <img src="img/beach.jpg" id="carousel-img" />
        <div id="tint"><p>PAUSED</p></div>

        <div id="carousel-controls">
            <img id="playBtn" src="img/play.png" />
            <img id="pauseBtn" src="img/pause.png" />
        </div>
    </div>
</div>

<script type="text/javascript">
changeImage();
</script>
```

15. Save this file and reload the browser. You should see that the autumn picture is now displayed instead of the beach. Copy the function call line changeImage(); a few times in a row and save and reload the browser to see how a different image will show up depending on how many function calls you have there. If you have 5 function calls in a row, the beach image will be visible again because index was cycled around the entire array!
16. Once you've had enough fun with that, remove those function call lines but keep the empty script tag in the body section there. We will need it again later.
17. Go back up to the JS in the head section. Below the changeImage function, create a new function called play. This will be the function to start playing the slideshow.
18. Within this play function, we only need to add 2 lines of code. The first one will create the interval timer using the setInterval(***function***, ***time***) function. In our case the function to call will be changeImage and we will go with 2000 as the time parameter (NOTE: the time is in milliseconds so 2000 means 2s – if you just put a 2 in there, the images will change so quickly you might feel sick!). For many cases, that would be fine as is, but we know that we will need to cancel the timer when the pause button gets pushed so right now we need to store the handle of our setInterval function call into the timer variable (the 4th global variable we made above). This line of code is:
    timer = setInterval(changeImage, 2000);
19. The other line we need in this function is removing the tint element and "PAUSED" which is conveniently placed within the tint. This is simple to do by changing the CSS display style:
    document.getElementById("tint").style.display = "none";
20. The completed play function should look like this:

```
function play () {
    timer = setInterval(changeImage, 2000);
    document.getElementById("tint").style.display = "none";
}
```

21. Select this entire function, Copy (Edit > Copy or Ctrl+C), and then click below it and Paste (Edit > Paste or Ctrl+V) so you have the same function twice. In the second one, change its

name to pause. In the second line in this pause function, change "none" to "block" so that CSS display style is reverted to make the tint and text shown again.

22. In the pause function, remove the first line that has the setInterval function call. Replace that line with:
clearInterval(timer);
23. Recall that timer is the variable that we use to store the timer made in play. Here we are clearing it (removing it) so that the interval timer is not running while in pause mode. This pause function should look like this:

```
function pause() {
    clearInterval(timer);
    document.getElementById("tint").style.display = "block";
}
```

24. We're nearly done this exercise. We need click event handlers on the play and pause button to call their respective functions when clicked. We can quickly add these in using inline event listeners in the img tags that are already present. The updated code for these icons should look like this:

```
<div class="wrapper">
    <div class="carousel-box" id="a">
        <img src="img/beach.jpg" id="carousel-img" />
        <div id="tint"><p>PAUSED</p></div>

        <div id="carousel-controls">
            <img id="playBtn" src="img/play.png" onclick="play();" />
            <img id="pauseBtn" src="img/pause.png" onclick="pause();" />
        </div>
    </div>
</div>
```

25. Save the file and reload the browser. Click on the play button and watch the images cycle through (watch for 20+ seconds to see that it cycles through the array properly. Click pause to make sure the tint and "PAUSED" text shows up and that the images stop changing. Click play and pause a couple more times to ensure they work correctly.
26. The last step is very simple. We want to default the slideshow to playing rather than being paused initially. Scroll down to the empty script tag in the body section, where you previously tested changeImage, and just add a function call to play() so that it will automatically start before clicking any buttons. This call looks like this:

```
<script type="text/javascript">
    play();
</script>
```

27. Save and reload the browser to see that it plays immediately now. Test the pause and play buttons again to ensure they still work fine.

EXERCISE 2: INSTANT MESSENGER

We are going to use JS to create a simple IM! It won't be a real 2-way messenger – you will be talking to a random bot who rarely makes sense – but it's still going to be a fun exercise! To create this module, we will use another click event for sending a message as well as a keyboard event to track the number of characters in the textarea as we type. For the automated bot, we are using another interval timer so that a random message will be sent from the bot every 10 seconds.

Have you noticed that sites like Facebook will change the title property (remember this is the title that shows up in the browser tab) when you get a new message? We will do this with our site too!

1. Navigate to your USB folder (F:) and into the ***cs2033*** > ***lab07*** subfolder.
2. Within ***lab07*** create another folder, called ***exer2***.
3. Open http://www.csd.uwo.ca/~bsarlo/cs2033b/labs/Lab07/exer2/, download ***exer2.html*** and ***styles.css***, and save them into ***lab07*** > ***exer2***.
4. Open exer2.html in Brackets and immediately save it in the same folder as ***exer2_complete.html***.
5. Create a blank file in Brackets and save it into ***lab07*** > ***exer2*** with the name ***messages.js***
6. Most work in this exercise will be done in messages.js and a few minor changes will need to be added to exer2_complete.html
7. We will need 6 functions in messages.js so let's begin by defining each of them and then we can go back and fill in the code in each one below. All 6 functions have 0 input parameters. The functions will be:
    - *sendMessage* – to send an IM that you type out
    - *updateCharCount* – to count how many characters you have typed
    - *autoMessage* – to create the timer for the random message bot
    - *randomMessage* – to randomly select a message from the bot and display it
    - *updateTitle* – change the title to indicate a new message
    - *revertTitle* – change the title back to its original text
8. Defining these 6 functions should be simple. The code looks like this:

```
function sendMessage () {

}

function updateCharCount () {

}

function autoMessage () {

}

function randomMessage () {

}

function updateTitle () {

}

function revertTitle () {

}
```

9. Now we will fill in the code for each of these functions.

sendMessage

10. First up, we need to get the msgBox textarea as a variable and then the text within it, which is stored as an attribute called value:
```
var msgBox = document.getElementById("msgBox");
var msg = msgBox.value;
```
11. Next we will use JS to create a new div using the built-in createElement function. Once we create that div, we will assign it the classes "im_msg" and "im_msg_right" because messages from you will be shown on the right while bot messages will be on the left. All the CSS has been provided for you so you don't have to worry about making any styles.
12. We also need to set the content of the new div, using innerHTML, to be the message that we typed in the textarea, currently stored in the msg variable.
13. Then we can add this new div to the big inbox element using the built-in appendChild function. The code for these last few steps looks like this:
```
// Create new div in inbox section to show message.
var newMsg = document.createElement("div");
newMsg.className = "im_msg im_msg_right";
newMsg.innerHTML = msg;
document.getElementById("inbox").appendChild(newMsg);
```

NOTE: Remember that lines beginning with // are called comments. These are human notes so we know what we're working on and are ignored by the browser. You don't have to type the same comments I did, but it's good practice to include some kind of commenting in your code.

14. Still in sendMessage, we have one more thing to do. We have to reset our textarea to be empty so that our previous message is no longer there. We can do this by accessing the value attribute again and setting it to "" (empty string). Include a function call to updateCharCount after this so that the character count is also reset to 0 here:
```
// Clear textarea once message is sent.
msgBox.value = "";
updateCharCount();
```

updateCharCount

15. This function will keep track of how many characters we have typed in the textarea. We have a limit of 64 characters (although we don't actually enforce this limit!) so if our character count exceeds 64, it is displayed in red instead of black.
16. The first step again is to retrieve the message from msgBox like we did above, but here we can do it all in one variable. Once we retrieve the msg, use the length attribute to obtain how many characters are within the msg:
```
var msg = document.getElementById("msgBox").value;
var charCount = msg.length;
```
17. Now that we know the number of characters in our message box, let's display it. First get the char_msg element as a variable and then use the innerHTML attribute to change its content to reflect the number of characters out of 64 (i.e. 0 / 64 is the default):
```
// Show the number of characters typed in the textarea.
var charMsg = document.getElementById("char_msg");
charMsg.innerHTML = charCount + " / 64";
```

18. The last feature of this function is to check if the number of characters, charCount, is greater than 64 or not and set the text colour of this charCount message to red or black accordingly. We can use a simple if-statement to check this:

```
// If more than 64 characters are entered, show it in red to indicate that it's too long.
if (charCount > 64) {
    charMsg.style.color = "red";
} else {
    charMsg.style.color = "black";
}
```

autoMessage and **randomMessage**

19. The autoMessage function is very short. We are just going to use the setInterval function call to create an interval timer which will trigger randomMessage (the next function we'll fill in) every 10 seconds. Remember these are in milliseconds so use 10000 here. Since we won't need to remove this time at any point (unlike the one in the carousel exercise), so we don't need to store our timer handle in a variable. Here's the code for this function:

```
function autoMessage () {
    // Start a timer that triggers the randomMessage function every 10 seconds.
    setInterval(randomMessage, 10000)
}
```

20. Now click into randomMessage. This will be the function that actually selects a random message and displays it as a bot message in the inbox panel.
21. Create an array called msgs and add at least 10 different messages. Mine are shown here but you can make up your own if you'd like:

```
var msgs = ["what's the weather like?", "I love pizza!", "sticks and
stones may break my bones but words will never hurt me.", "how are things
going?", "programming is so much fun", "I'm sleepy.", "I went to the
movies on the weekend.", "reading week was a blast", "lol", "James and I
are going to the party", "gotta do some homework", "have a good time"];
```

22. Next we will use a random number generator to select a random integer between 0 and the position of the last item in the list (i.e. if you have 12 messages, this will pick a number from 0 to 11). Once the number is picked, we take the message at that index using the square brackets to grab an individual element:

```
// Choose a random message from the list of possible messages.
var rnd = Math.floor((Math.random() * msgs.length));
var randomMsg = msgs[rnd];
```

23. Similar to what we did earlier in sendMessage, we should now create a new div and assign it the appropriate classes (remember bot messages will use the im_msg_left class to show up on the left side) and the random message as its content. We then add the new div to the inbox panel. The four lines of code for this portion are shown here:

```
// Create new div in inbox section to show message.
var newMsg = document.createElement("div");
newMsg.className = "im_msg im_msg_left";
newMsg.innerHTML = randomMsg;
document.getElementById("inbox").appendChild(newMsg);
```

24. The last thing to do in randomMessage is a simple function call to updateTitle so that the title property will show that a new message has come in (this function will be filled in next):

```
updateTitle();
```

updateTitle and **revertTitle**

25. These last two functions are very short and easy! We are going to change the document's title attribute to set a new title text in each of these two functions. In updateTitle, it will be: "* New Message * | My Simple IM"
26. In revertTitle, the title will be the same as it was from the original HTML: "My Simple IM"
27. These two functions in their entirety look like this:

```
function updateTitle () {
    document.title = "* New Message * | My Simple IM";
}

function revertTitle () {
    document.title = "My Simple IM";
}
```

Event handlers

28. Save the messages.js file and open exer2_complete.html in Chrome (or reload it if you had it open from before). If you type in the textarea, the characters aren't being counted, and if you click the Send button, nothing happens. Also there are no messages from the bot no matter how long you wait. This is because we need to attach a few event listeners to the HTML so that our JS functions will be executed. Open exer2_complete.html in Brackets.
29. First we will attach an onload event listener to the body. This is an event that triggers once the entire body is finished loading in the browser. Call the autoMessage function from this event because this is the function that creates the interval for the bot's messages.
    <body> *should be changed to:*
    <body onload="autoMessage();">
30. Now, remember that a new message from the bot is going to change our title property using updateTitle. Now we will revert the title back to its original text by calling revertTitle when you click anywhere in the inbox. Add an onclick event listener to call revertTitle.
     should be changed to:

31. Next we need a keyboard listener so that our character count is being done as we type each character in the textarea. We will use onkeyup which triggers when a key is released rather than when it is first pushed down (it makes a difference if someone holds a key down instead of just pushing it quickly).
    <textarea class="msgBox" id="msgBox"></textarea> *should be changed to:*
    <textarea class="msgBox" id="msgBox" onkeyup="updateCharCount();"></textarea>
32. The final addition to our code is a click handler on the button so that we can send our message when clicking the button.
    <button class="send_btn">Send</button> *should be changed to:*
    <button class="send_btn" onclick="sendMessage();">Send</button>
33. Save the file and reload it in Chrome. Test each of the components to make sure it works:
    - Watch the character count change as you type and delete characters. Type over 64 characters to see that it changes to red, then delete some to make it black again (don't worry that the message sends if you have over 64 – we didn't enforce it!)
    - Send messages and see them show up in the inbox AND that the textarea and character count are reset after you send.
    - Watch the bot's random messages come in every 10 seconds AND the title property change when they come in.
    - Click on the inbox to see the title revert to its original text.

EXERCISE 3: FORM VALIDATION

This exercise introduces you to simple form validation with JavaScript. The form is provided for you and you will be learning how to add onblur, onchange, and onclick (this one you've seen before!) to the input fields. For name and age we will perform simple if-statements to check if they are valid based on their length and/or value types and change the border colour accordingly. Then you will learn how to dynamically hide and show an input type based on a selection from another input. The Spouse Name textbox will appear if you select that you are Married, but it will disappear if anything else is selected. The final portion is less about validation and more of a useful tool within forms. You will learn how to automatically select all checkboxes at once by checking off a master checkbox and letting JS do the work for you!

1. Navigate to your USB folder (F:) and into the ***cs2033*** > ***lab07*** subfolder.
2. Within ***lab07*** create another folder, called ***exer3***.
3. Open http://www.csd.uwo.ca/~bsarlo/cs2033b/labs/Lab07/exer3/, download ***exer3.html***, and save it into ***lab07*** > ***exer3***.
4. Open exer3.html in Brackets and immediately save it in the same folder as ***exer3_complete.html***. All work will be done in exer3_complete.html.
5. Open exer3_complete.html in Chrome as well. Notice that the form is all there but nothing special happens when you type or click on the inputs.
6. Click into the empty script tag in the head section. Just like in the last exercise, let's add our function definitions first and then go back and fill them in. We need 4 functions, each of which takes 0 input parameters:
    - *checkName* – to check if the entered name is valid
    - *checkAge* – to check if the entered age is valid
    - *updateStatus* – to hide/show the spouse name depending on the status selected
    - *selectAll* – to check or uncheck all the food checkboxes at once
7. The function definitions will simply look like this:

```
function checkName () {

}

function checkAge () {

}

function updateStatus() {

}

function selectAll() {

}
```

checkName

8. For the sake of this exercise, we will say that a name is valid if it's 5 or more characters long and invalid otherwise. Before we perform this check, we need to get the name textbox

and its text value as variables. These are the first 2 lines in checkName:

```
var nameBox = document.getElementById("name");
var name = nameBox.value;
```

9. Now we can add the if-statement to check if the name is long enough. We will use the length attribute to get the number of characters in the name. If it's less than 5 characters, we'll set the border colour to red, and set it to gray otherwise:

```
if (name.length < 5) {
    // Name is too short.
    nameBox.style.borderColor = "red";

} else {
    // Name is long enough.
    nameBox.style.borderColor = "gray";

}
```

checkAge

10. The age will be dealt with a little differently since we expect it to be a number and we'll impose the additional requirement that the age must be 10 or more so that young kids aren't allowed on the site (nobody would ever lie about their age! ☺)
11. Start this function the same as the name one; obtain the age textbox and the value that was typed into it:

```
var ageBox = document.getElementById("age");
var age = ageBox.value;
```

12. We are ready to add out conditional to check if the age is valid. There are 2 requirements for the age: it must be a number and it must be 10 or more, so if the age is either text (not a number) or it is less than 10, the age is flagged as invalid. JS has a built-in function called isNaN() which indicates if the given variable is a number or not so we'll use this for our first requirement, followed by a simple < comparison for the second one. Remember that if either condition is not met, then the age is invalid so we will use the || (or) operator between the two sub-conditions. The border colour will change accordingly:

```
if (isNaN(age) || age < 10) {
    // Age is not a number or is less than 10 years old.
    ageBox.style.borderColor = "red";
} else {
    // Age is a number AND is 10 or higher.
    ageBox.style.borderColor = "gray";
}
```

updateStatus

13. This function will hide or show the spouse name textbox depending on the relationship status selected. We are really only concerned with the "married" radio button since that's the only one that should cause the spouse name box to appear. So we'll start by obtaining that married radio button. While we're at it, get the spouse name box and its label into variables as well:

```
var marriedStatus = document.getElementById("married");
var spouseNameLabel = document.getElementById("spouseLabel");
var spouseNameBox = document.getElementById("spousename");
```

14. Now that we have these variables, we can perform a simple conditional to determine if the married radio button is selected ("checked" is used for radio buttons as well as checkboxes). If it is checked, we will change the classes assigned to the spouse name box and label to be visible, and if not, the classes will be changed to hide them. The following code will perform this check and update the classes accordingly:

```
if (marriedStatus.checked) {
    // If user selects "Married", display the textbox for spouse name.
    spouseNameLabel.className = "visible"
    spouseNameBox.className = "textbox visible"
} else {
    // If user does not select "Married", hide  the textbox for spouse name.
    spouseNameLabel.className = "hidden"
    spouseNameBox.className = "textbox hidden"
}
```

checkAll

15. In this function we want to use the master checkbox to check or uncheck the other checkboxes below it. First we obtain the value/status of the master checkbox by accessing the "checked" attribute of the checkbox element. We also need to gather the entire array of checkboxes below it so we'll use the getElementsByName() function to get them all at once:

```
var isChecked = document.getElementById("all").checked;
var checkboxes = document.getElementsByName("interests");
```

16. Next we perform an if-statement to see if our master checkbox was clicked to check it or to uncheck it. If it's true, then we want all the interests checkboxes to be true; if false, then they should be set to false. In either scenario, we need to perform a for-loop to iterate over the entire list of interest checkboxes and change each one to be the respective status (true or false). The for-loops are identical in both cases and the only difference is the status we are assigning to the checkboxes in the loop. The following code must be added to checkAll:

```
if (isChecked == true) {
    // Check all checkboxes.
    for (var i = 0; i < checkboxes.length; i++) {
        checkboxes[i].checked = true;
    }
} else {
    // Uncheck all checkboxes.
    for (var i = 0; i < checkboxes.length; i++) {
        checkboxes[i].checked = false;
    }

}
```

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)! ☺