Lab 4 Task 2

Open `Lab4Task2.html inside browser.`

Try clicking on any of the three buttons on the page. You will notice that none of them do anything currently. This is because our JavaScript and jQuery code has not yet been put in place.

The desired behavior for each of the buttons are:

- **Toggle Colors**: Toggles the color between white and cardinal.
- **Toggle Rounded Borders**: Toggles the appearance of a border on the boxes.
- **Add a New Box**: Adds one new box to the container.

Now, open `Lab4Task2.html` in your text editor. Take a moment to understand HTML/CSS code that's present. Note that

- The `<div>` with `id="boxes"` contains three boxes.
- Each of the boxes are of `class="box"`.
- The `<div>` with `id="toggles"` contains three buttons.
    - Button `id="button_toggle_colors"` should toggles the color of the boxes
    - Button `id="button_toggle_roundedges"` should toggle the rounded borders of the boxes
    - Button `id="button_add_box"` should add an additional box

We have already added in the `.click()` listeners for each of the buttons for you. Your job is to fill in the code within those listeners. You may refer to Exercise 1 to refresh your memory about jQuery button click listeners.

Instructions

You will need to:

1. Add the `outlined` CSS class to each of the 3 boxes.
2. Modify the `background-color` CSS attribute on the boxes whenever the **Toggle Colors** button is clicked.
3. Add/Remove the `round-edge` class on the boxes whenever the **Toggle Rounded Borders** button is clicked.
4. Add a new box `<div>` whenever the **Add new box** button is clicked.

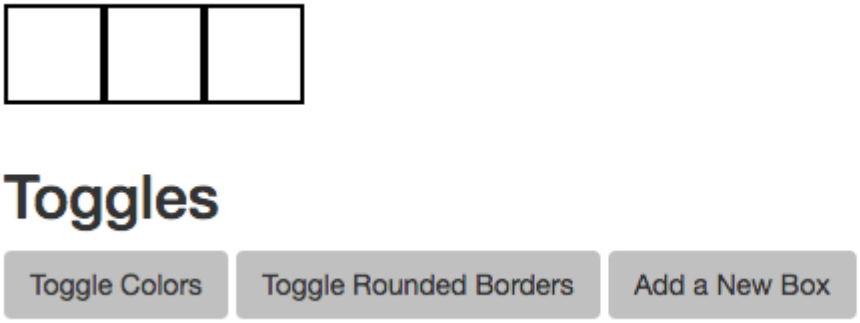
Detailed Instructions

Step 1: Adding a CSS class manually

We want there to be borders around all our boxes, and in this step we do this using a manual process. We have already created a CSS class named `outlined` for you, so we need to add that class to each box `<div>`. This is done by adding the string `outlined` to the class attribute of the boxes, after the `box` string. Remember to include a space between the two words. You should end up with the following code.

```
<div id="box1" class="box outlined"></div>
<div id="box2" class="box outlined"></div>
<div id="box3" class="box outlined"></div>
```

Refresh the page, and you should now see the following:



Step 2: Modifying CSS attributes with `.css()`

We will now handle the button that changes the boxes' background color. We have already created the color `colorTeal` which we would like to color the boxes with. We'd like to take an approach to accomplish this task which doesn't involve the use of CSS classes, and instead modifies the CSS templates of an element directly.

Locate the button click listener for the `id="button_toggle_color"` button. Once again, we will need to to use a jQuery Class Selector and and the jQuery `.each()` method to select all elements with the CSS class `box`, on the selector. You should add the the code and have the following:

```
$("#button_toggle_color").click(function() {
  $(".box").each(function () {

  });
});
```

Next, we want to check if the element has the CSS attribute `background-color` with a value of `colorTeal`. We make use of the jQuery method `.css()` to check the current value of the element's CSS value.

```
if ($(this).css('background-color') == colorTeal) {
    // element currently has cardinal background color
    // remove it

}
else {
    // element currently has the no background color
    // add it

}
```

We made use of the `.css()` method, with one argument, to check the value of an element's CSS attribute. In order to set the value, we use the same method, but include a second argument to specify the value to set the CSS value to.

```
if ($(this).css('background-color') == colorTeal) {
    // element currently has cardinal background color
    // remove it
    $(this).css('background-color', '');
}
else {
    // element currently has the no background color
    // add it
    $(this).css('background-color', colorTeal);
}
```

Step 3: Adding/Removing classes using jQuery's `.addClass()` and `.removeClass()`

In step 2, we modified CSS templates directly, which can get unwieldy when you have to get, check and set multiple values each time. This is why we make use of CSS classes to define how we want certain elements to look, and subsequently attach them to the relevant elements. This is what we did in in step 1 where we added the class `outlined` to the `<div>` elements manually. The method introduced in step 1 is a manual process, and one can see how this becomes unfeasible with many elements.Also, what we want is to procedurally toggle the classes on the elements when the appropriate buttons are clicked. As such, we shall use jQuery to achieve these aims.

We will now handle the button that changes the boxes' border style from straight-edges to round-edges.. We have already created the CSS class named `round-edge` for you. What we want to do is to add the `round-edge` class to each of the boxes if they do not have them, which would cause them to be have rounded edges. If a box already has the `round-edge` class, we remove it.

Locate the button click listener for the `id="button_toggle_roundedges"` button. The first thing we will need to do is to use a jQuery Class Selector to select all elements with the CSS class `box`. Because we want to manipulate each element, we will need to make use of the jQuery method `.each()` on the selector. Type the following into the `.click()` listener:

```
$(".box").each(function () {


});
```

Take a moment to understand what's happening with this code. The selector `$(".box")` is a Class Selector which selects all the elements with the CSS class `box` assigned to it. The `.each()` method acts as an iterator, such that any code that we put within the `function()` block affects the individual elements. Within the `function ()` in the `.each()` method, we can access the element we are currently affecting using the `$(this)` method. You probably have seen similar mechanisms for self-reference in other programming languages such as `this` in Java or `self` in Python.

The first thing is to detect if the current element has the `round-edge` class assigned to it already. We use the jQuery method `.hasClass()` method to do so. So, add the following code within the `function ()`:

```
if ($(this).hasClass('round-edge')) {
 // element currently has rounded edges
 // remove it
```

```
}
else {
  // element does not have rounded edges
  // add it
}
```

Finally, we need to fill in the code for each of the two cases. For the first case, the element already has the `round-edge` class assigned to it, so we make use of the jQuery `.removeClass()` method to remove the round-edge class from the element. For the second case, the opposite case is true, meaning that the element does not have the `round-edge` class. We make use of the jQuery `.addClass()` method to add the `round-edge` class to it. Your code should look like this:

```
if ($(this).hasClass('round-edge')) {
  // element currently has rounded edges
  // remove it
  $(this).removeClass('round-edge');
}
else {
  // element does not have rounded edges
  // add it
  $(this).addClass('round-edge');
}
```

It is worth noting that this is a verbose way of performing toggle functionality. To leverage the power of jQuery even more, you can actually make use of the jQuery `.toggleClass()` method to achieve everything you did in the previous steps. In other word, the entire `if-else` block can be replaced with just the following line of code:

```
$(this).toggleClass('round-edge');
```

This line of code does everything we want it to do. It checks for the presence of a class on an element, and removes it if it is already present. If not, it adds it. This showcases the power of using jQuery and CSS classes.

Step 4: Adding new elements

In our final step, we will show how to dynamically add elements to your web interface using a combination of JavaScript and jQuery. With the **Add a New Box** button, it needs to create a box similar to what we had originally. This is achieved by creating a new `<div>` element. However, to be a box, it needs to have the classes `outlined` and `box`.

Locate the button click listener for the `id="button_add_box"` button. Within the listener, we need to create a new `<div>` element. This is achieved by using JavaScript's `document.createElement()` method. Add the following line into the listener method

```
var new_box = document.createElement('div');
```

First, we need to set the attribute `id` for the new box. We currently have three boxes, so we would like the next one to have `id="box4"`, and the next `id="box5"` and so on. We simply need to count the number of existing `box` `<div>`s in the document. We first can use a jQuery Class Selector to get all the required elements, and then make use of the jQuery field `.length` to get the number of elements.

```
var new_box = document.createElement('div');
var existingBoxes = $(".box").length;
```

We thus just need to set the `id` attribute of the new box by using the jQuery `.attr()` method to set the `id` attribute of the new box.

```
var new_id = existingBoxes+1;
$(new_box).attr("id", "box"+new_id);
```

Next, we want to add the two classes, `box` and `outlined` to the new box. We once again make use of the `.addClass()` method.

```
$(new_box).addClass("box");
$(new_box).addClass("outlined");
```

Finally, to add the new `<div>` we just created for the new box to the current list of boxes, we use an ID Selector first to find the element `boxes`, and then use the jQuery `.append()` method to add the new box.

```
$("#boxes").append(new_box);
```

Testing

You should now test your implementation to make sure that everything works correctly. Perform the following steps, and your set of boxes should resemble the output we present.

Perform the following

1. Click Toggle Colors
2. Click Add a New Box
3. Click Add a New Box
4. Click Add a New Box
5. Click Toggle Colors
6. Click Add a New Box
7. Click Add a New Box
8. Click Add a New Box
9. Click Toggle Rounded Borders

If you follow the above steps, your output should resemble the following image.