Lab 8

Objectives

- Understand how to make a get request to a RESTful web services
- More practice working with JSON objects

References / Resources

- Lecture example using the Open Movie Database API:
  - https://jsfiddle.net/Aestey/39uycepn/28/
- JavaScript objects
  - https://www.w3schools.com/js/js_objects.asp
- Using JavaScript to perform actions to HTML:
  - https://www.w3schools.com/js/js_htmldom_methods.asp
- Getting started with jQuery:
  - https://www.w3schools.com/jquery/jquery_get_started.asp
- Translation between JavaScript and jQuery:
  https://www.w3schools.com/js/js_jquery_selectors.asp

SETUP – Download the necessary files

1. Go to the BrightSpace course page, and in the **Assignments > Lab 8** section, download lab8.html, lab8.css, and lab8.js.
2. You should now be able to open up these files in a text editor to edit them. You can also open up the lab8.html file in a browser to view the web page.

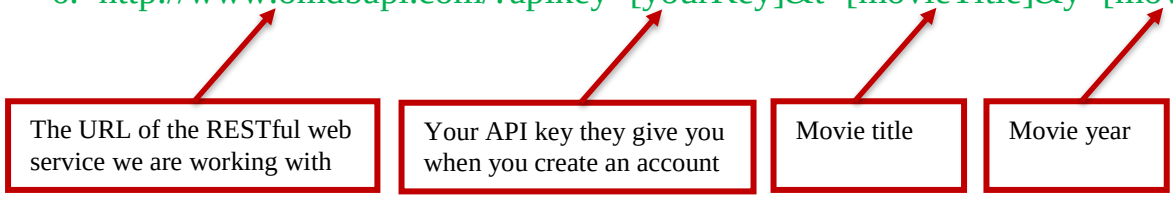
Part I – A RESTful web service GET request

1. Open up the **lab8.html** file in a text editor. Add the following three lines into HEAD section:

```
<script src="lab8.js" defer></script>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<link href="lab8.css" type="text/css" rel="stylesheet">
```

   The first file links our lab8.js JavaScript file with our HTML. The second allows us to use jQuery. The third links our lab8.css CSS file to provide style rules.

2. Open up the **lab8.html** in a web browser and see what happens when you click the submit button (right now it does not matter what is input into the input fields).
3. Open up **lab8.js** in a text editor. You'll notice two functions: the submit() function is called when we click the button – it uses jQuery to show the output section of our HTML when the button is clicked. The other function is what we will call after our REST request to handle the data returned from the RESTful API.
4. I recommend making an account at the Open Movie Database, by clicking the following link: http://www.omdbapi.com/apikey.aspx
5. Using the lecture slides and/or https://jsfiddle.net/Aestey/39uycepn/28/ as a reference, get the values entered by the user into the text fields and add them to the URL. The reference guide for interacting with the Open Movie Database is here: http://www.omdbapi.com/. We will need to set up our URL to make the request so that it has the following form:
6. http://www.omdbapi.com/?apikey=[yourKey]&t=[movieTitle]&y=[movieYear]



   So, if you created an account and your API key was 01234 and you wanted to search for Avengers from year 2012, the URL would be the following:

   http://www.omdbapi.com/?apikey=01234&t=Avengers&y=2012

7. You will write in your own API key manually; which will be emailed to you once you create a free account. You will need to add the movie title and year based on the values input into the text fields. (Reminder: to use the JSFiddle link above as a guide, as this is very similar to what we did in the lecture on REST).

   Remember how we add things to strings:

```
let text = "Anthony Estey";
text += "is the instructor"; // text value is now: "Anthony Esteyis the instructor"
```

8. After you have the correct URL figured out, making a REST request using jQuery is quite simple. The slides show the general form:

```
$.get(url, function(data){
    // add code here to do things with the JSON data received
});
```

the url you created above, with your api key, movie title, and movie year

Here is what my code looks like up to this point for the submit function:

getting the value entered into the text fields (lab 4)

the web service URL with my API key

adding the entered movie title and year to the URL

```
function submit() {
    $("#output").show();

    /* given Avengers, we want the URL to be: */
    /* https://www.omdbapi.com/?apikey=<YOUR_KEY_HERE>&t=Avengers&y=2012 */

    let title = $("#movieTitle").val();
    let year = $("#movieYear").val();

    let url = "https://www.omdbapi.com/?apikey=94677657";
    url = url + "&t=" + title + "&y=" + year;

    $.get(url, function(data){
          document.getElementById("raw").innerHTML = JSON.stringify(data);
          displayResults(data);
    });
}
```

outputting the raw data returned from the web service to the section labeled with id "raw" in our HTML

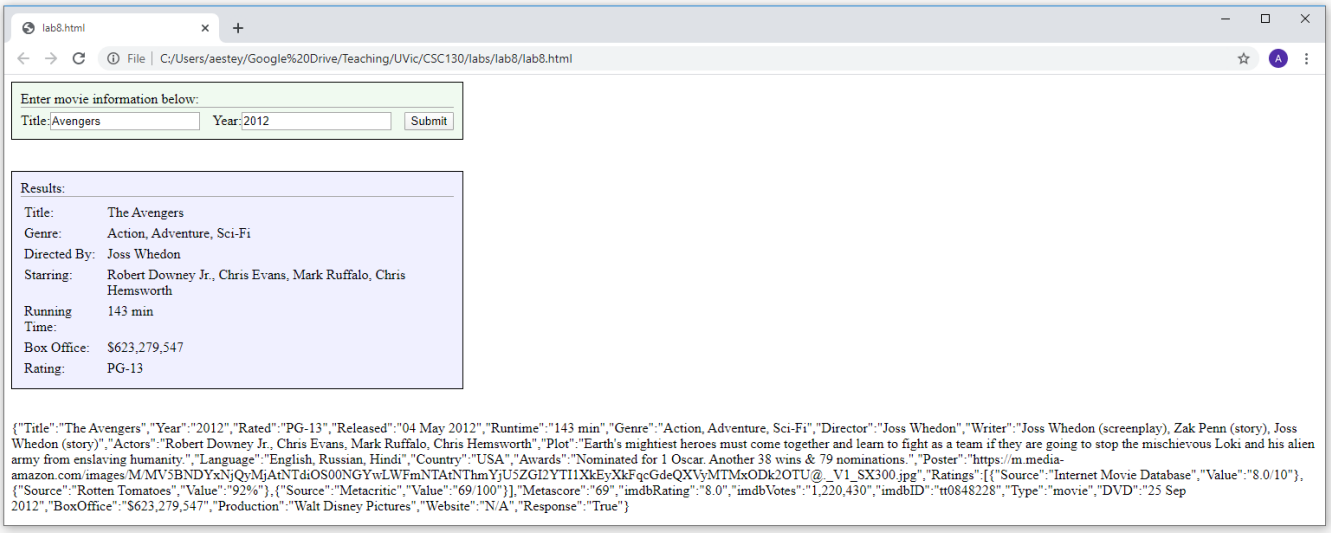
At this point, your webpage should do the following if Avengers is entered:

CHECKPOINT 1

Part II – Handling the JSON returned

1. Now that we have finished the submit() function, we want to handle the JSON data returned in the displayResults() function (that we called at the bottom of the submit() function).
2. Currently all the information (stored in a variable called 'data') is in JSON form, and we need to take out the title, genre, director, etc to fill in our table in the Results section.
3. Determine which attributes in the JSON data we need to access to fill out all of the different fields in the table.
4. We will also need to look at the HTML code to get the ids of the places in the table where we want to put this information (similar to how we needed the ids of the input boxes to see what movie the user wanted to search for).

   Here is some sample output for Avengers:



   Note that all of the information displayed in the blue Results table is taken straight from the raw data displayed at the bottom.

There are multiple ways of doing this, one with temporary variables and one without. First is using temporary variables to access each attribute from the JSON:

```
function displayResults(data) {
	let title = data.Title;
	let genre = data.Genre;
	let director = data.Director;
	let stars = data.Actors;
	let runningTime = data.Runtime;
	let boxOffice = data.BoxOffice;
	let rating = data.Rated;

	$("#resultTitle").html(title);
	$("#resultGenre").html(genre);
	$("#resultDirector").html(director);
	$("#resultStars").html(stars);
	$("#resultTime").html(runningTime);
	$("#resultBoxOffice").html(boxOffice);
	$("#resultRating").html(rating);
}
```

There second does not use any variables, it just sets the HTML of each of the table cells directly from the JSON data object variable:

```
function displayResults(data) {
	$("#resultTitle").html(data.Title);
	$("#resultGenre").html(data.Genre);
	$("#resultDirector").html(data.Director);
	$("#resultStars").html(data.Actors);
	$("#resultTime").html(data.Runtime);
	$("#resultBoxOffice").html(data.BoxOffice);
	$("#resultRating").html(data.Rated);
}
```

CHECKPOINT 2

Part II – Making images out of the movie Rating

1. The movie ratings following the MPAA standard, and there are images for each of the possible ratings (G, PG-13, R, etc).

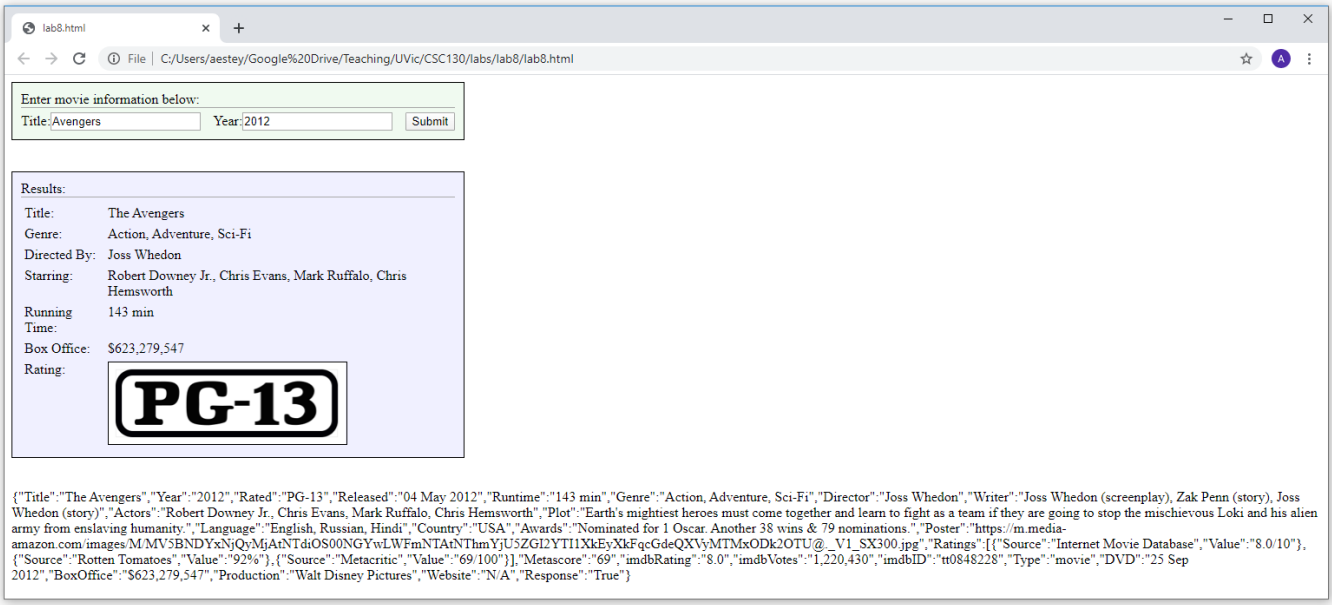
   At the top of your JavaScript file, add variables for each of the following movie rating image source URLs (copy and paste them from below):

```
let g = "https://docs-assets.developer.apple.com/published/218def98e6/267c8fd4-12a4-4fbb-89ef-ed88d909b9f1.png";
let pg = "https://docs-assets.developer.apple.com/published/f44edd2a0a/9a2a66de-b42e-447b-8894-89182d4b0b08.png";
let pg13 = "https://docs-assets.developer.apple.com/published/8cab854be4/c9b697bd-27a2-4b98-8330-38842529f91a.png";
let nc17 = "https://docs-assets.developer.apple.com/published/66836a12ac/d8e30ec4-21ce-41fc-a97c-f7067b49da4a.png";
let r = "https://docs-assets.developer.apple.com/published/f3448319da/9d1c255f-eae3-42a1-a557-dbf5f33eaaa1.png";
let unrated = "https://docs-assets.developer.apple.com/published/522c8ab8e3/a34a2e4f-be16-4d5f-b5fd-9922473c771b.png";
```

   (I have made the font really small, because multi-line links did not work for an earlier lab, and I hope to avoid that issue again)

2. Using the movie rating of the movie, display one of these rating images instead of the textual rating (so show the PG-13 icon instead of putting "PG-13" in text).

   For example:



   You will need to use if-statements to do this, and check whether the rating is G, PG, PG-13, etc and display the correct image accordingly.

Here is a sample solution:

```
let g = "https://docs-assets.developer.apple.com/published/218def98e6/267c8fd4-12a4-4fbb-89ef-ed88d909b9f1.png";
let pg = "https://docs-assets.developer.apple.com/published/f44edd2a0a/9a2a66de-b42e-447b-8894-89182d4b0b08.png";
let pg13 = "https://docs-assets.developer.apple.com/published/8cab854be4/c9b697bd-27a2-4b98-8330-38842529f91a.png";
let nc17 = "https://docs-assets.developer.apple.com/published/66836a12ac/d8e30ec4-21ce-41fc-a97c-f7067b49da4a.png";
let r = "https://docs-assets.developer.apple.com/published/f3448319da/9d1c255f-eae3-42a1-a557-dbf5f33eaaa1.png";
let unrated = "https://docs-assets.developer.apple.com/published/522c8ab8e3/a34a2e4f-be16-4d5f-b5fd-9922473c771b.png";

function displayResults(data) {
	let title = data.Title;
	let genre = data.Genre;
	let director = data.Director;
	let stars = data.Actors;
	let runningTime = data.Runtime;
	let boxOffice = data.BoxOffice;
	let rating = data.Rated;

	$("#resultTitle").html(title);
	$("#resultGenre").html(genre);
	$("#resultDirector").html(director);
	$("#resultStars").html(stars);
	$("#resultTime").html(runningTime);
	$("#resultBoxOffice").html(boxOffice);

	let ratingImageURL = unrated;
	if (rating === "G") {
		ratingImageURL = g;
	} else if (rating === "PG") {
		ratingImageURL = pg;
	} else if (rating === "PG-13") {
		ratingImageURL = pg13;
	} else if (rating === "NC-17") {
		ratingImageURL = nc17;
	} else if (rating === "R") {
		ratingImageURL = r;
	}
	$("#ratingImg").attr("src", ratingImageURL);
}
```

CHECKPOINT 3 - LAB COMPLETE