# Project 2

## Objectives (Upon finishing this project, you should be able to):

- Write HTML, CSS, and JavaScript code to create a web page from scratch.
- Write HTML code to add elements to a page that collect user input
- Understand how to use core programming constructs including variables, if-statements, loops, arrays, and objects.
- Write JavaScript code that interacts with a page's HTML elements
- Write JavaScript code to interact with a RESTful web service

## Deadline:

- 11:59pm on Friday, November 26th, 2021

## Submission:

On the BrightSpace course page, under the Assignments section, click Project 2.

In the submission textbox, you must enter the address of your webpage (example: http://webhome.csc.uvic.ca/~aestey/project2.html).

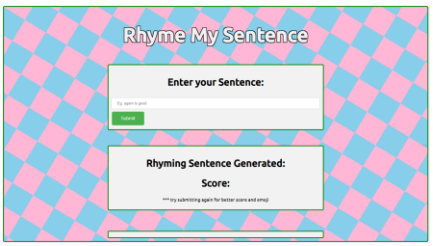
For this project, you must have all of your page components uploaded to the UVic servers before the project deadline.

## Requirements:

The core part of Project 2 is the interaction between your web page and a RESTful web service. Overall, there are two main parts to Project 2:

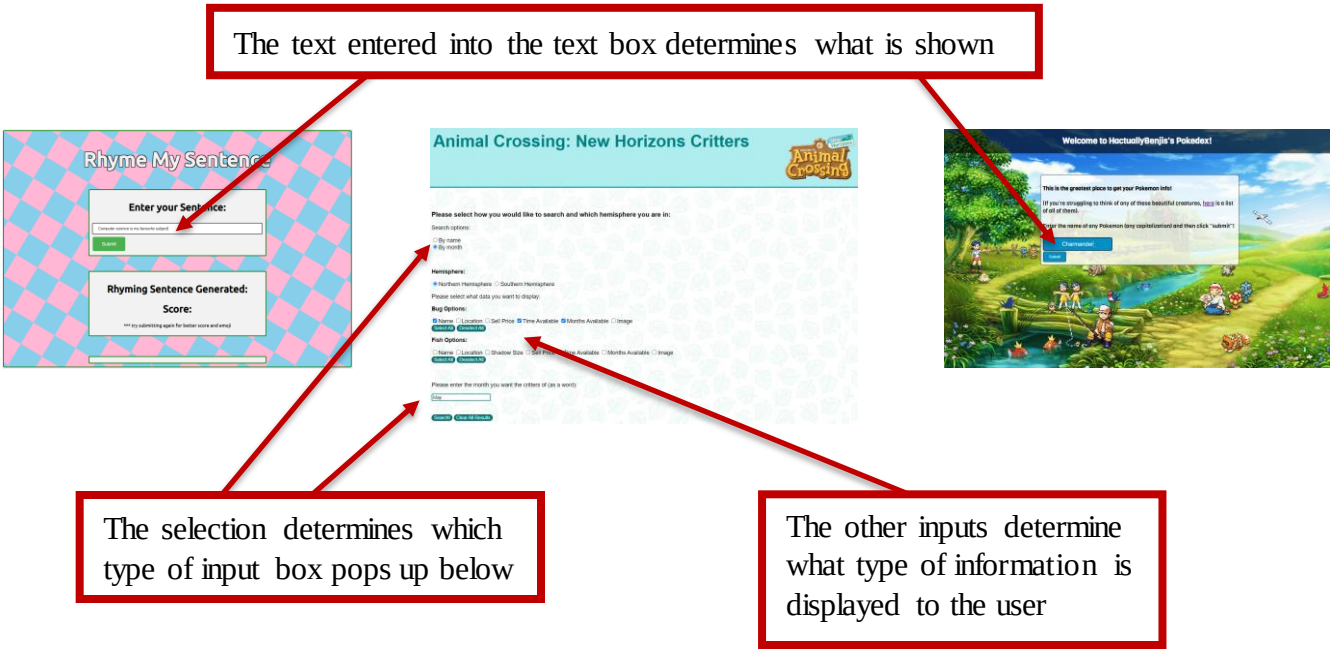
1) The default view of your web page where a user can enter information. The type of information the user enters depends on the theme of your page. In Lab 5 you got some practice with a number of different ways you can collet input from a user.

   You must write the code for this part of your page in HTML and CSS. The HTML will determine which input elements you chose, and the CSS will style them. Everything you see in the three pages below is written in HTML and CSS:






   You will need to write JavaScript to handle the user input. All of the labs from Lab 5 onward have provided you with some experience with the connection between HTML and JavaScript (ie. by calling functions with button clicks). The key thing for this part of the assignment is to grab the information the user entered (so it can be used to set up the URL to send to the RESTful web service for the next part).



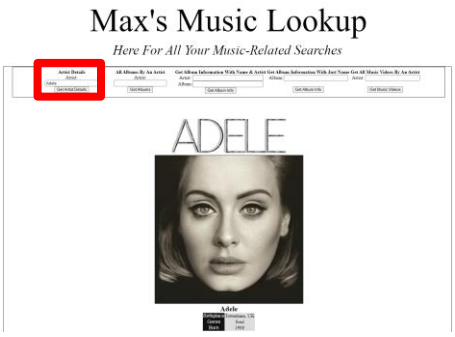
2) The second part of the page involves the content that is added to your page based on the information received from the RESTful web service. Your task for this part is to determine how to interpret the JSON data received, and pull information out from the JSON data to add content to your page.

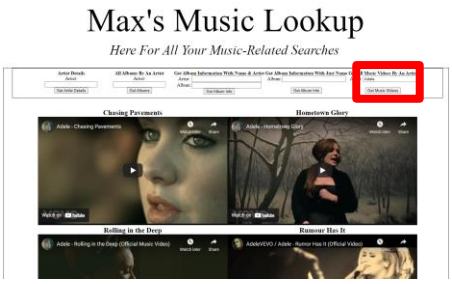
   All of the labs from Lab 5 onward have provided you with experience updating an HTML element by first selecting it from JavaScript (ie. by selecting an element with `document.getElementById`, or with jQuery's `$("#elementID")`). The key to this part is to be able to (i) pull the important information from the JSON and (ii) use that information to update the page contents. For example, the following JSON is returned from The Open Movie Database about Avatar:

{"Title":"Avatar","Year":"2009","Rated":"PG-13","Released":"18 Dec 2009","Runtime":"162 min","Genre":"Action, Adventure, Fantasy","Director":"James Cameron","Writer":"James Cameron","Actors":"Sam Worthington, Zoe Saldana, Sigourney Weaver","Plot":"A paraplegic Marine dispatched to the moon Pandora on a unique mission becomes torn between following his orders and protecting the world he feels is his home.","Language":"English, Spanish","Country":"United States, United Kingdom","Awards":"Won 3 Oscars. 89 wins & 131 nominations total","Poster":"https://m.media-amazon.com/images/M/MV5BMTYwOTEwNjAzMl5BMl5BanBnXkFtZTcwODc5MTUwMw@@._V1_SX300.jpg","Ratings":[{"Source":"Internet Movie Database","Value":"7.8/10"},{"Source":"Rotten Tomatoes","Value":"81%"},{"Source":"Metacritic","Value":"83/100"}],"Metascore":"83","imdbRating":"7.8","imdbVotes":"1,159,379","imdbID":"tt0499549","Type":"movie","DVD":"22 Apr 2010","BoxOffice":"$760,507,625","Production":"N/A","Website":"N/A","Response":"True"}

   Depending on the theme of the page, there are many different pieces of information that could be pulled from the JSON code above. On a page about movie ratings the Rotten Tomatoes score might be used. On a page about movie history, maybe the year of release is important, or the genre or director. There is also a link to the the poster image, so the following image could be displayed based on that URL:

The user input determines which information you pull from the JSON. In the page below, if the user searches for Adele in the **Artist Details** box, then general information and a picture of Adele is displayed. On the other hand, if the user searches for Adele in the **Get All Music Videos By Artist** box, then a collection of her YouTube videos are displayed:




In JavaScript, you will need to write code that handles the user input, and creates a URL to send to the RESTful server to make a request for information. The information you provide in the request will depend on the type of RESTful server you are working with (ie. For the animal crossing page, a time of year is given to the server, whereas in the Music Lookup example, it is an artist's name).

Setting up the URL so that it has the right information to give to the server will be a key part of this step. Once that is done, the main task is to interpret the JSON.

## Grading Criteria:

| Grading item | Points |
|---|---|
| **Submission** | |
| Working link submitted on time to BrightSpace (all components work) | 2 |
| **Page content (before server interaction)** | |
| User input content (input fields, check boxes, buttons, etc) | 2 |
| CSS style and look of the input components | 2 |
| Input component functionality (connects to JavaScript correctly) | 1 |
| **REST** | |
| Collect information input from user correctly. | 2 |
| Create REST URL correctly (base URL + user input) | 2 |
| **JavaScript** | |
| Populating page with JSON information | 2 |
| JSON data adds something beyond text to page | 2 |
| Overall Quality/Effort of content added to page from JSON | 5 |
| **Total** | **20** |

The University of Victoria follows a percentage grading system in which the instructor will submit grades in percentages. The University will use the following Senate approved standardized grading scale to assign letter grades. Both the percentage mark and the letter grade will be recorded on the academic record and transcripts.

| F | D | C | C+ | B- | B | B+ | A- | A | A+ |
|---|---|---|---|---|---|---|---|---|---|
| 0-49 | 50-59 | 60-64 | 65-69 | 70-72 | 73-76 | 77-79 | 80-84 | 85-89 | 90-100 |

| Grades | Description |
|---|---|
| A+, A, A- | **Exceptional**, **outstanding** or **excellent** performance. Normally achieved by a minority of students. These grades indicate a student who is *self-initiating*, *exceeds expectation* and has an *insightful* grasp of the subject matter. |
| B+, B, B- | **Very good**, **good** or **solid** performance. Normally achieved by the largest number of students. These grades indicate a *good* grasp of the subject matter or *excellent grasp in one area balanced with satisfactory grasp in the other areas*. |
| C+, C | **Satisfactory**, or **minimally satisfactory**. These grades indicate a *satisfactory performance and knowledge* of the subject matter. |
| D | **Marginal Performance**. A student receiving this grade demonstrated a *superficial grasp* of the subject matter. |
| F | **Unsatisfactory performance**. Wrote final examination and completed course requirements; no supplemental. |

## APIs to use:

Music:

- Spotify: https://developer.spotify.com/documentation/web-api/
- SoundCloud: https://developers.soundcloud.com/docs/api/guide
- YouTube: https://developers.google.com/youtube/
- The Audio DB: https://www.theaudiodb.com/api_guide.php

Fashion:

- 22 Fashion APIs: https://www.programmableweb.com/category/fashion/api
- 12 Clothing APIs: https://rapidapi.com/collection/clothing-api

Politics:

- 47 Politics APIs: https://www.programmableweb.com/category/politics/api
- Civic Information API: https://developers.google.com/civic-information
- Vote Smart API: https://votesmart.org/share/api
- Non-Profit APIs: https://www.programmableweb.com/category/non-profit/apis?category=20309

Sports:

- The Sports DB: https://www.thesportsdb.com/api.php
- MySportsFeeds API: https://www.mysportsfeeds.com/data-feeds/
- Yahoo Fantasy Sports API: https://developer.yahoo.com/fantasysports/guide/
- The Best Sports APIs: https://rapidapi.com/blog/best-sports-apis-ranked/
- Top 10 Sports APIs: https://www.programmableweb.com/news/10-top-sports-apis-2021/brief/2021/07/30

Movies

- The Open Movie Database: https://www.omdbapi.com/apikey.aspx

Series:

- Marvel: https://developer.marvel.com/
- Star-Wars: https://swapi.dev/

Food and Drink:

- Links to Food and Drinks APIs: https://public-apis.io/category/food-and-drink-apis

Finance:

- Links to Finance APIs: https://rapidapi.com/category/Finance
- Google Finance API: https://rapidapi.com/blog/google-finance-api-alternatives/

Gaming:

- Riot: https://developer.riotgames.com/
- Blizzard: https://develop.battle.net/
- Pokemon: https://pokeapi.co/
- Top Gaming APIs: https://rapidapi.com/blog/top-video-game-apis/
- 525 Gaming APIs: https://www.programmableweb.com/category/games/api

Social media:

- Discord: https://discordapp.com/developers/docs/reference
- Reddit: https://www.reddit.com/dev/api
- Slack: https://api.slack.com/web
- Telegram: https://core.telegram.org/
- Twitter: https://developer.twitter.com/en/docs
- Facebook: https://developers.facebook.com/docs/graph-api/
- Instagram: https://developers.facebook.com/docs/instagram-basic-display-api
- Instagram (option 2): https://developers.facebook.com/docs/instagram-api/

Fitness

- FitBit: https://dev.fitbit.com/getting-started/
- Nutrition: https://www.nutritionix.com/business/api

Weather:

- Weatherbit: https://www.weatherbit.io/account/create
- Darksky: https://darksky.net/dev/register
- OpenWeatherMap: https://openweathermap.org/
- Wunderground: https://www.wunderground.com/login

Reddit post about fun APIs to play with:

- https://www.reddit.com/r/webdev/comments/3wrswc/what_are_some_fun_apis_to_play_with/