Using JavaScript to Consume REST Services

A **web service** is a service that utilizes standardized mechanisms by which a software application (or client) can connect to and communicate with another software application (or client) using web protocols. A web service typically uses HTTP (for communication) and JSON/XML (for data exchanges). Here is a good tutorial on YouTube on REST API concepts that we covered in Modules 4 and 5.

Creating a Service Portal

Your task for this exercise is to create interfaces (or clients) within a single HTML file that utilizes jQuery to consume at least **two** (related) already existing web services of your choice. Your HTML file must use Bootstrap (Bootswatch or similar) and uses **tabs** for switching between one service interface to another. You are free to use any format or design to display the results. Your HTML page must incorporate a style sheet of your own choice (e.g. you can use an existing one or build your own). You will be graded based on creativity and presentation. *Your web services do not need to run on Node.js environment.* Please view the Module 3-5 Examples to assist you with building the web service interfaces required for this exercise. In addition, please use the GitHub API example as a starting point. Ensure that you properly format and document your code.

Bonus: 2 marks if integrates Google Maps API into the exercise.

Where can I find web services?

There are many service providers that offer web services on the web. Some of these services are for free whiles others require subscription. Typically, service providers that require subscription offer a free trial to test a web service. You may need to register for an account and obtain an API key to consume the service. The following is a list of web resources that will help you find web services of interest. You can use the jQuery ***fetch*** or ***get*** methods in asynchronous functions.

- Public APIs: https://github.com/public-apis/public-apis
- Any API: https://any-api.com
- Rapid API, https://rapidapi.com
- APIs.io, http://apis.io
- Google API Explorer, https://developers.google.com/apis-explorer/#p
- API List: https://apilist.fun
- Programmable Web: https://www.programmableweb.com/apis/directory
- Weather API: https://openweathermap.org/api
- Twitch API: https://dev.twitch.tv/docs/api
- YouTube API: https://developers.google.com/youtube/v3

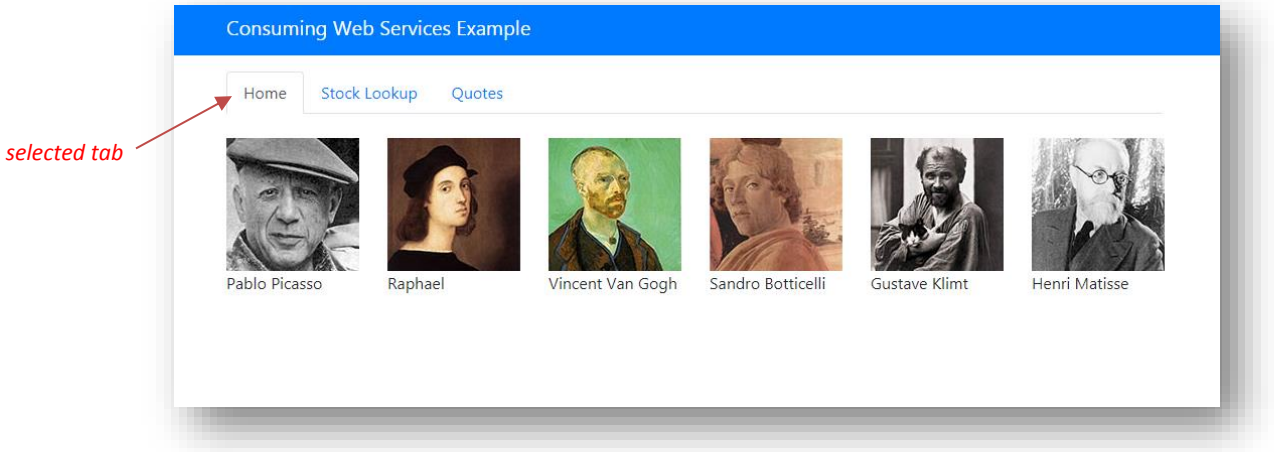
What are some ideas for creating the interface?

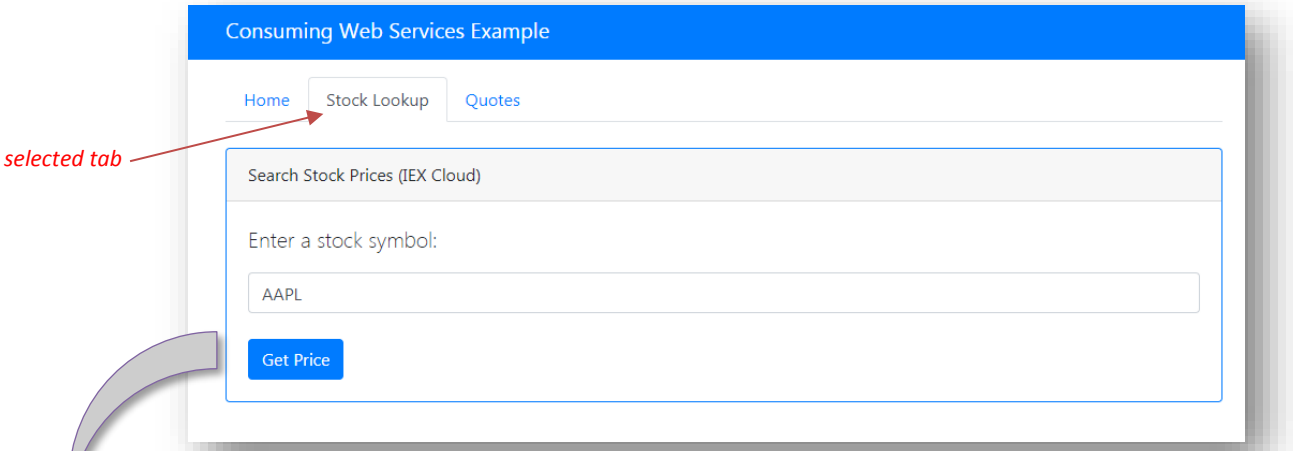
You may implement an interface that provides multiple options to search for specific channels or users on a streaming/media platform (e.g. YouTube, Twitch.tv), using machine learning or artificial intelligence to identify images/emotions using JavaScript and Tensorflow.js (Google's machine learning platform). Below are some YouTube videos that will inspire you on how to build your client interface and features you can include.

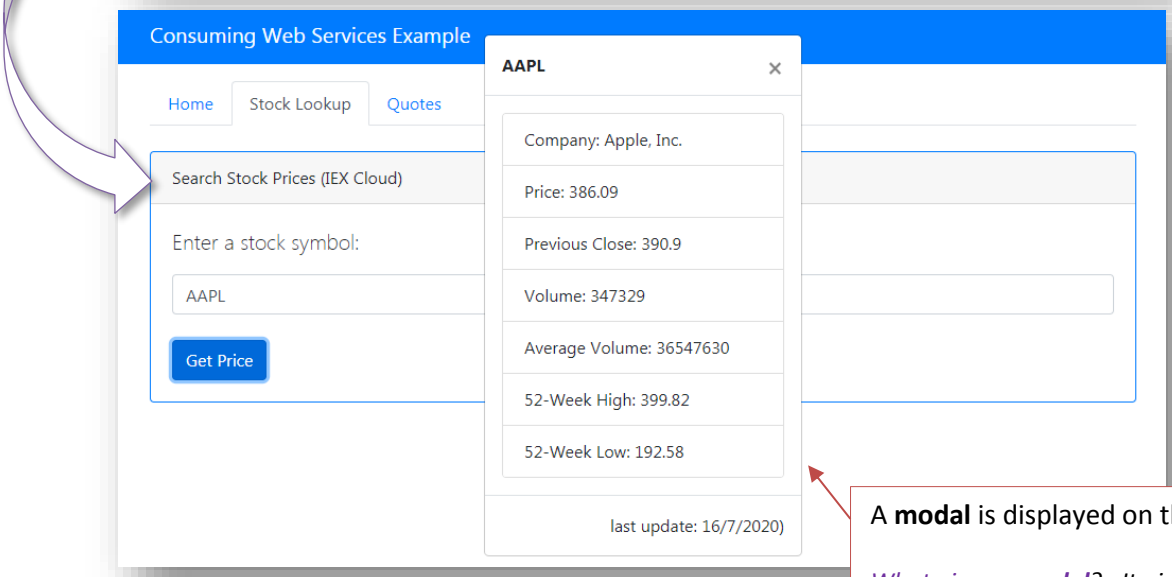
- Build Real Time Face Detection With JavaScript: https://www.youtube.com/watch?v=CVClHLwv-4I
- Make your own Twitch Dashboard: https://www.youtube.com/watch?v=VTY6ZzDTV3A
- Google Maps JavaScript API Tutorial: https://www.youtube.com/watch?v=Zxf1mnP5zcw
- Weather Application using JavaScript: https://www.youtube.com/watch?v=wPElVpR1rwA
- Air Quality using JavaScript: https://www.youtube.com/watch?v=Tiot877orkU
- Sending SMS using JavaScript: https://www.youtube.com/watch?v=dRFI0emtQDw

An example of two services are shown below:

Service 1: A Stock Symbol Lookup (used IEX Cloud web service):



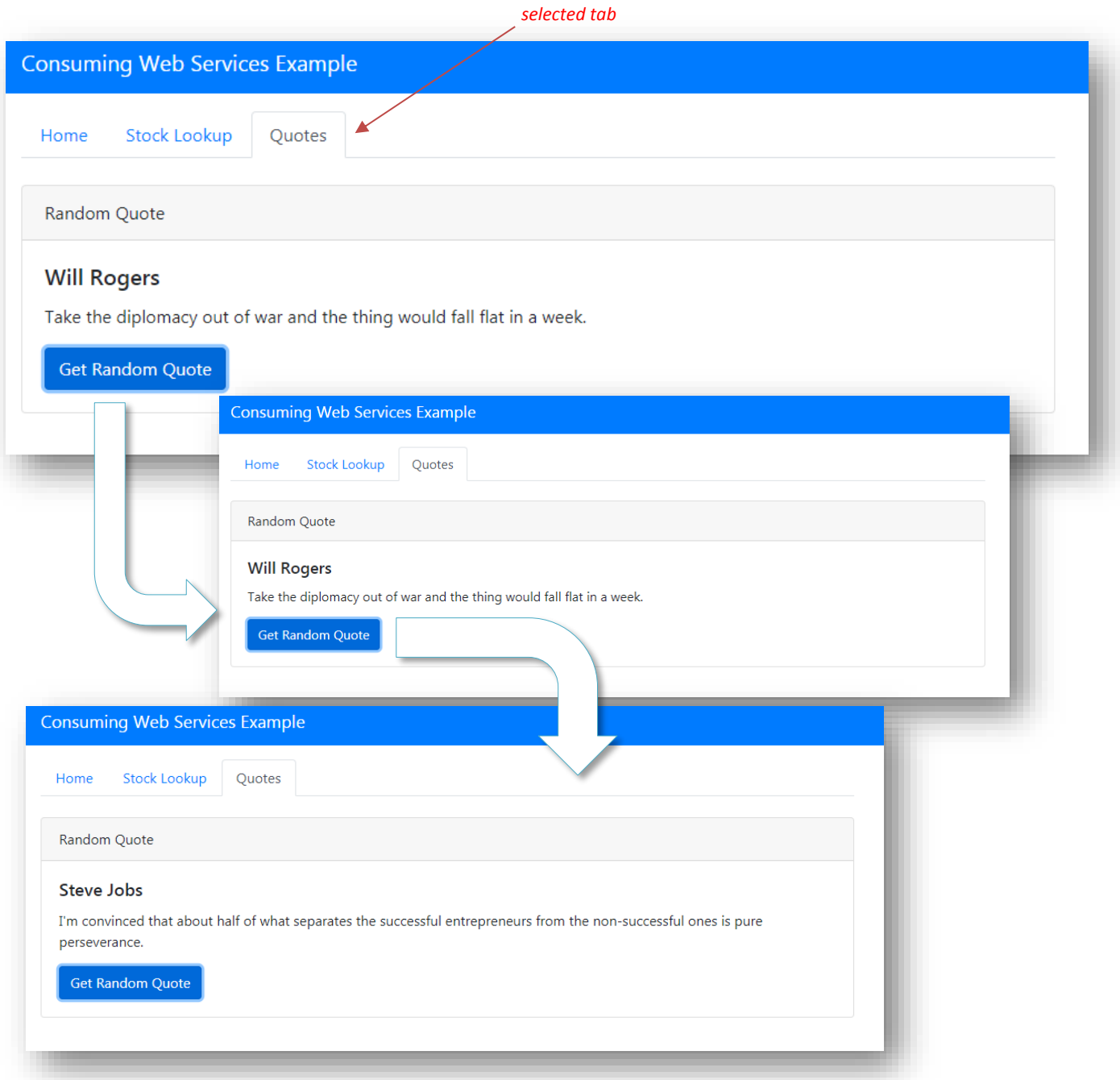




A **modal** is displayed on the same page.

*What is a **modal**? It is an interactive graphical dialog window similar to that of window.alert(), except you have full control over how it is programmed or displayed.*

Service 2: Random Quotes (used [Quotes REST API](#))



SUBMIT What to submit:

1. Create a readme.txt file that describes the services you used and their locations (e.g. endpoints).
2. Compress all of the files (html, JavaScript, CSS, images, etc.) for exercise 3 into zip format. Call this file **userid-ex3.zip** (where userid is your UW userid). The compressed file should include the same structure as that of the one you have downloaded with the additions you have applied throughout completing the exercise.
3. Upload your **userid-ex3.zip** file to Canvas | Assignments | Exercise 3.
   [Bonus] 1 bonus point for publishing the exercise on GCP or Heroku.

Grading Rubric: 15 points: 8 points for implementation, 4 for creativity and 3 for presentation