Seneca | FACULTY OF CONTINUING EDUCATION AND TRAINING

SENECA WEB PROGRAMMING PROGRAM
SUMMER 2019 SESSION

Module Number: WEB 304
Assignment #2

Instructor: Tim Lai
Title: JavaScript MVC UI Application
Assigned: Wednesday June 26
Draft Presentation Date: 6:30pm to 9:30pm Wednesday July 10
Draft Presentation Value: 5% of 40%
Final Due Date: 11:59pm Wednesday July 17
Final Value: 35% of 40%

Submission details: This assignment MUST be submitted in at least one of 2 ways, as specified below. If the assignment does not follow the specified instructions, it will be considered incomplete. *Failure to upload this assignment will result in a grade of zero for this portion of the assignment*.

1. This assignment should be uploaded to a **GitHub repository** which I will invite you to. When I download this GitHub repository, I must be able to access all your source files so that I can see your JavaScript, JSON, HTML and CSS code and test your application.
2. If you have difficulty uploading to GitHub, then you may alternatively upload your assignment as a ZIP file to an online cloud storage site such as **Google Drive** or **WeTransfer** and send me a link at **tim.lai@senecacollege.ca**. *DO NOT EMAIL ME A ZIP FILE AS AN ATTACHMENT. I WILL NOT RECEIVE IT.*

This assignment has 2 submission points which correspond with the dates indicated above:

1. **Draft Presentation:** The first submission will be a rough draft of the entire assignment. The goal of this draft is to demonstrate that you understand the requirements and broad concepts of the assignment. It is not necessary for every part of the assignment to work and there will be no deductions for errors. You will present your draft to the class, explaining how your code works and what the purpose of it is. You will provide feedback on the draft assignments of your classmates and accept feedback on your own.
2. **Final:** The second submission will be the completed version of the assignment. This version should be fully-functional and there will be deductions for any errors as indicated in the rubric.

Instructions: You will be creating an **MVC UI application** using pure **JavaScript** (ES6) which allows users to view a list of **objects** which can belong to a **class of your choice** (i.e. Spaceship, Dog, Clown etc.) *as long as it is NOT a Duck – be creative*. This application will store the objects in a **JSON file**

which is loaded using **AJAX**. *DO NOT USE A JAVASCRIPT SUPERSET SUCH AS TYPESCRIPT OR A LIBRARY OR FRAMEWORK SUCH AS JQUERY, ANGULAR, ANGULARJS, REACT, VUE OR EMBER*.

Your application should have a distinct **Model**, **View** and **Controller**, and should follow proper **object-oriented** and **MVC** principles to the best of your ability. It should also use logical folder structure and naming conventions. Your Model must be defined as a **class** with a **constructor method** which defines at least **4 properties** – an **id** which is a **number**, a **name** which is a **string**, an **image** which is a **string** and something **unique to your class** which is a **Boolean**. These properties should be defined using **getter** and **setter** methods which define properties which are accessed privately, using either **symbols** or **weakmaps**. Each setter method should check the property's data type and display an error message to the console if it does not match the required type.

The **Controllers** should be separated into distinct **Service** and **Component** files. The Service should be made up of one or more classes which are used to transfer data from the Model to the View using distinct **ajax**, **getObjects**, and **getObject** scripts to collect the object data from the JSON file using AJAX, cross-check the data types of the objects using the Model class and filter the data to display either all of the objects or a single object. You may use either the **XMLHttpRequest** constructor or the **Fetch API** for AJAX.

There should be a generic Component class which uses setters to define a selector property as an HTML element and that the service property's prototype belongs to a service class. There should be two more components, one of which should be used to show all objects with the other used to show one object. These components should use methods to collect data from the service. The single object component should also define at least **3 methods** which can be used in the View.

The application should start on a page which displays the **name** of each of the objects which have been retrieved from the JSON file. The user should also be able to view an **image** which is associated with each object. There should be a distinct **link or button** associated with each object which displays a single object when clicked. When the user views a single object, they should be able to use an **input** or **button** to alter content or appearance of the object using a **browser event**. The user should also be able to navigate back to the previous page from this page.

The application should have an intuitive and functional **user interface** which uses valid HTML and CSS. Nothing complicated or fancy is required but the user interface should be easy to navigate and use. It should make use of **buttons**, and **colour-coding**. Basic accessibility considerations should be made for non-standard users (i.e. alt text on <img> tags). Responsiveness and use of libraries such as Bootstrap, Foundation and Font Awesome is NOT required but is encouraged.

Deliverables:

- A Model defined as a **class** with at least **4 properties** including an **id**, a **name**, an **image** and something **unique to your class**, a **constructor method** and corresponding **getter and setter methods**
- A **JSON** file which stores all this data, and which follows valid JSON formatting rules
- One or more **Service** classes with distinct **ajax**, **getAllObjectsFromJSON** and **getSingleObjectFromJSON** scripts which makes use of either **XMLHttpRequest** constructor or the **Fetch API**
- A generic **Component** class which defines **selector** and **service** properties using getter and setter methods

- An **ObjectsAllComponent** and an **ObjectSingleComponent** which pass the **selector** and **service** to their respective templates with at least **3 other methods** in the **ObjectSingleComponent**
- At least 2 View page types: an **All Objects** page and a **Single Object** page (you should rename the pages to match your class i.e. All Dogs)
- A **button or link** which is associated with each object on the All Objects page which displays a single object when clicked
- An **input** or **button** which alters the content or appearance of the object using a **browser event**
- An external CSS file
- A functional and intuitive **UI/UX design** which makes use of **buttons** and **colour-coding**
- Accessibility considerations for non-standard users (blind, assistive devices, etc.)

Grading breakdown:

- Presentation of first draft **(5%)**
- Creation of a JSON file which follows valid JSON formatting rules **(5%)**
- Creation of a Model with properties, getter and setter methods and type checking **(15%)**
- Creation of View templates which display objects using HTML and CSS **(15%)**
- Creation of Services which retrieve objects from the JSON file using AJAX **(25%)**
- Creation of Components which transfers data and methods to the View **(25%)**
- Good coding/file organization practices (property names, bracket indentation etc.) **(10%)**

Your files should contain only valid (as determined by the W3C validator) HTML code. You can use invalid code in the HTML but if you do, you must include a comment beside the invalid code explaining your decision. *You will lose 2% per type of uncommented validation error*. Under normal circumstances, you should not have any JavaScript errors in your console. *You will lose 2% per type of JavaScript console error which could have been fixed*. You MUST have valid JSON code according to JSONLint (https://jsonlint.com/). *You will lose 2% per JSON error*.

Late Policy

All late assignments will be given a grade of zero.

Plagiarism

There are serious penalties for cheating and plagiarism offences and you are expected to be aware of our Academic Honesty Policy. Please refer to the Academic Policy at http://www.senecacollege.ca/ academic-policy/acpol-09.html for more information.