# PROG 1700 JavaScript – Final Project

# Client-side Javascript programming

Assignment Value: 15% of overall course mark.

Due Date: **See due date designated on the Final Project dropbox on D2L.**

Late submissions will receive the standard late submission penalty as stated in the course outline.

(5% overall deduction per day late, and 0% after assignment handed back to the class.)

### Project – Phased Submission

In no way, shape, or form should this project be attempted in its entirety the day before it is due. If attempted, this will lead to the Earth spinning off of its axis, the complete breakdown of society, and the abolishment of many popular flavours of ice cream, including moon mist. Seriously, this project is not designed to be done in one sitting. The following phases of delivery will be followed, each with their own due dates in subsequent classes:

**Phase 1**: Wireframe and storyboard designs for the main web page, plus necessary page for the Wikipedia lookup of each country.

**Phase 2**: Basic web page design implemented to the point that the main web page with the country form launches and the other page (wiki) can be launched by clicking on the appropriate item on the main page. Demonstrate pseudocode for handling data flow between JSON file and application to support phase 3 functionality.

**Phase 3**: The major functionality of the Countries of the World web page completed, including: the loading of the list of countries, the appropriate information shown when selecting a country including its flag image, and the ability to switch the output between kilometres and miles. The Wikipedia lookup page completely working.

***Bonus Phase 4****: Wireframe and storyboard designs for the quiz web page. Demonstrate pseudocode for quiz and high score to support bonus functionality. The complete functionality of the Flags of the World Quiz web page working along with high scores.*

### Project Instructions

#### Use VS Code to create an HTML5 website with Client-side JavaScript (and supporting CSS, JSON, and image files) in which you’ll code the solution to this program.

### Submissions

You will submit your work for this assignment via GitHub. While you will have frequent commits/pushes of your assignment code to GitHub as your work on it, the instructor needs to know which version to mark and when it was committed. So, when you have completed all assignment work, put a “Ready for Marking” comment on the last code committed into GitHub. Then, submit a simple text document to the Brightspace Dropbox that contains the full URL to that commit. It is this Dropbox submission that will be used to determine late penalties, so make sure to do so prior to the assignment deadline.

**Once you have committed the code, make sure to visit the repository page on GitHub’s website to verify that the final version has been pushed to GitHub as that is where the instructor will go to get your code.**

### Evaluation

To insure the greatest chance of success on this assignment, be sure to check the marking rubric contained at the end of this document or in D2L. The rubric contains the criteria your instructor will be assessing when marking your assignment.

# Program – Countries of the World

Required Resources– Provided by the instructor

* **PROG\_PROJ\_CountriesOfTheWorld\_Sample.zip** – A zip file containing the following:
  + **countries.json** – A JSON file containing a list of world countries, including name, population and area data.
  + **flags (folder)** – A folder named flags, which contains image files for every country’s flag.
  + **original\_high\_scores.json** – A JSON file containing five dummy values for high scores on the flag quiz. Will be updated with actual high scores achieved by the program users.
* **A video on D2L demonstrating how the finished website should work**.

### Program Specifications

Create a dynamic HTML5 website that displays information about any country the user selects from a list.

For any country selected, the application will display the following data:

* Name of the country
* An image of the country’s flag
* The population of the country
* The country’s total area, shown in sq. miles (by default) or sq. kilometres, according to user preference
* The country’s population density per square mile (by default) or square KM, according to user preference
* The country’s percentage of the total world population

The country data being displayed will change automatically whenever a new country is selected from the list.

On the display for each country a button will be presented to launch a second web page, which will load the Wikipedia entry for the selected country.

*BONUS:*

*Another dialog window can be launched via another button/link, which will present a Countries of the World flag quiz in a new web page. This quiz will randomly show a country’s flag, present a drop-down with the correct country and three other random country choices, and allow the user to proceed through a quiz of 10 questions. Once complete, the user will be shown their quiz score. If it qualifies as a top-5 high score, they will be prompted for a high score username to be recorded. Finally, the all-time high scores list will be shown.*

### Program Functionality

When the application starts, the main form of the web page will be mostly blank. To start, the user will be shown only a list of the country names in a select box populated from the countries.json file. Once the user selects any country from the list, the main form display area will show information specific to the selected country, as detailed above.

In addition to displaying the country’s flag and basic demographics, the webpage should include a dropdown list to allow users to change the Total Area display from its default setting of square miles to square kilometres. When this occurs, the “Population Density per” output should be updated to show the current user preference, and the country’s population density should change to reflect the value in the chosen unit of measure.

Users will be able to press a “Wiki Country” button at the bottom of each country’s display that will launch a new webpage that will load and display the Wikipedia page for that particular country**.** This should open in a new tab so that the user can easily return to the main web page.

*BONUS:*

*The user will be allowed to launch another web page via a link or button on the main page, which will present a Flags quiz in the same tab as the original page. This quiz will randomly show a country’s flag, present a drop-down with the correct country and three other random country names, shown in random order, and allow the user to submit their choice. The user will be immediately informed of their updated score after each guess via a score progress output on the page. The user will be taken through a quiz of 10 questions, and will have buttons to cancel (and close) or restart the quiz at any time.*

*Once the last quiz question is answered, the user’s final quiz score will be calculated and displayed on screen. Their score will be evaluated to see if it is a new high score, through a comparison to values originally loaded from the high-scores.json file, but later updated and maintained in HTML5 local storage in the web browser. If the final score qualifies as a top-5 high score, the user will be prompted to enter their three-letter (e.g. MHC) username, which will be saved as a new high score in local storage. Only the five highest scores are saved, so a new entry will effectively bump the lowest off of the list. The score must be higher than one of the previous entries, so earlier scores “win” in case of a tie (ie. the first 10/10 will be the highest score forever). The username, score, and timestamp will be saved to local storage in the format indicated in the high\_scores.json file. Do not forget to drop the lowest score if there is a new entry as we only store 5 highest scores. Regardless of whether there is a new entry or not, the highest scores will then be displayed to the user in a well formatted way, in the area of the web page near the section for guessing a country. The user will still have buttons at the bottom of the page to either cancel (ie. Close) the web page and return to the main page or restart the quiz.*

## Program Structure

The JavaScript code should be structured to organize the code into appropriate units of related logic, with an effort to avoid duplication of code and other programming inefficiencies.

To help code’s organizational structure, the following functions/methods are required in your solution:

function DisplayCountryData()

function DisplayPopulationData()

function CalculateTotalWorldPopulation() 🡨 Returns a float

function CalculateAreaInKM(countryAreaInMiles) 🡨 Returns a float

**Please note:** Your final solution will absolutely need more functions than just those listed above, these requirements are intended as a starting point only.

## Program Quality

Your solution should meet standards most modern users expect from a quality, responsive web application.

**The following screenshots are meant as a simple guide, but you may pursue your own design for the web pages.**

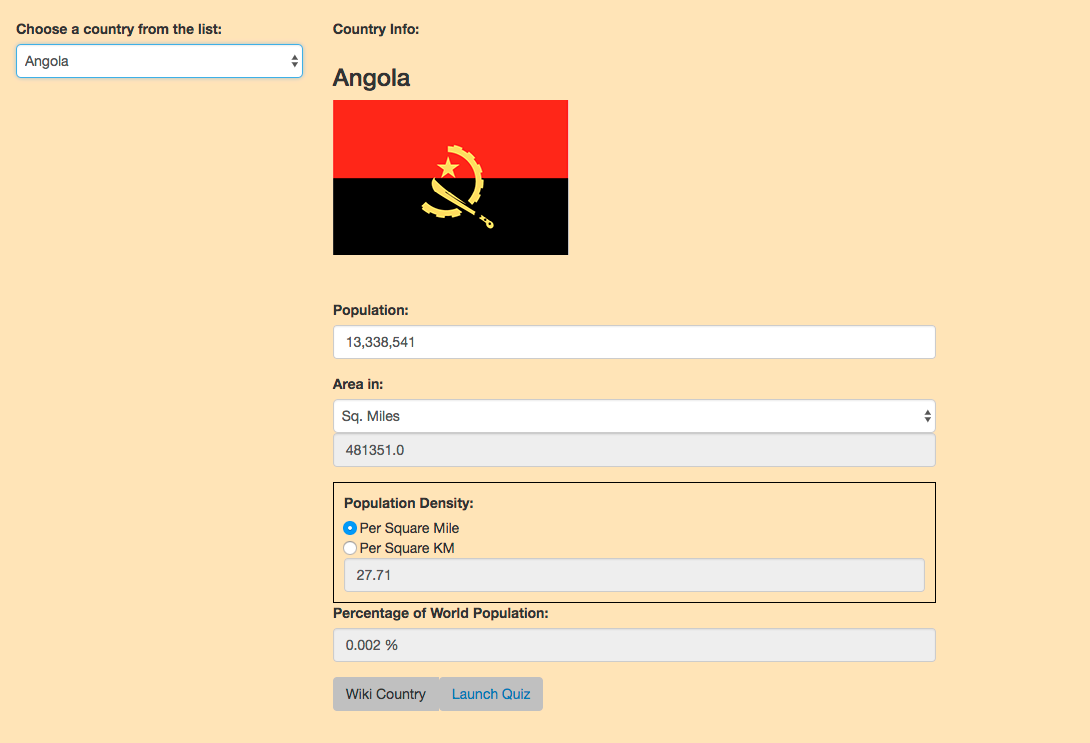
Efforts should be made to:

* Present clean, intuitive and user-friendly forms, with descriptive labels and acceptable layout.
* Include data validation wherever appropriate, although unlikely to be necessary for the base project.
* Include proper formatting of any data/information displayed to the user.
* Use proper scope for all variables, and use constants where appropriate.
* Handle for any reasonably-predictable errors that could occur in the application.
* Display helpful and descriptive error messages, if an exception is encountered.

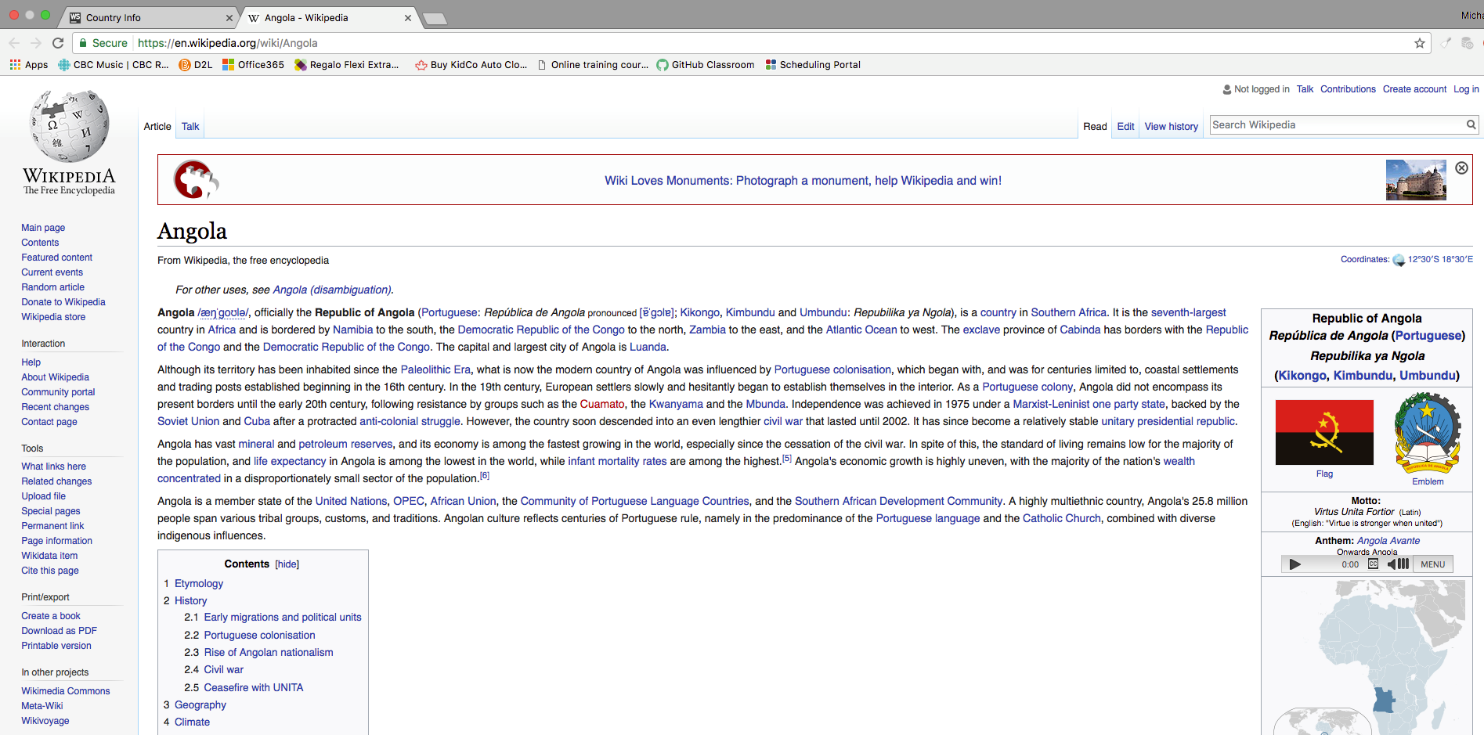
Sample Screenshots

|  |  |
| --- | --- |
| At startup, with country details area either hidden or not populated | Selecting a country from the select box |
|  |  |
|  |  |

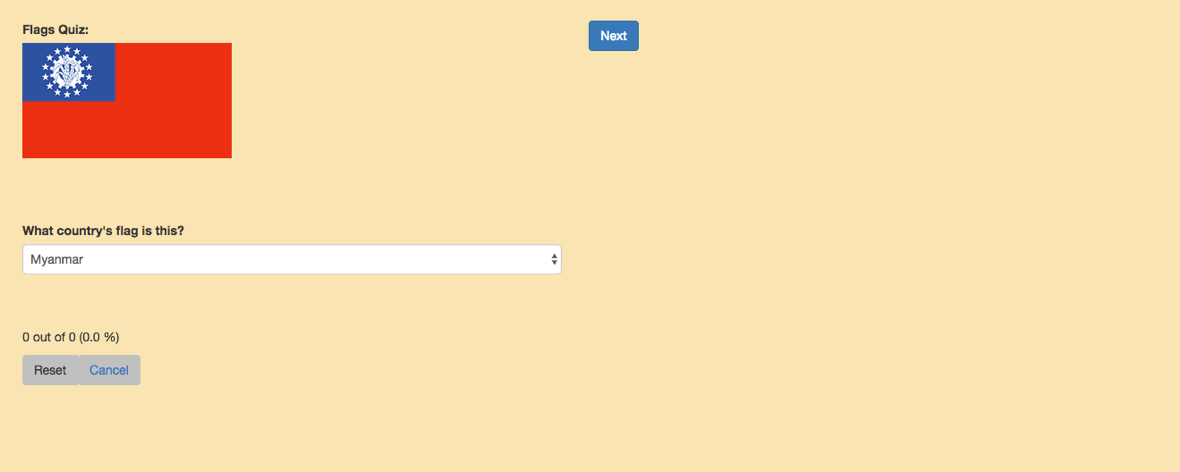
After a country has been selected from the select box



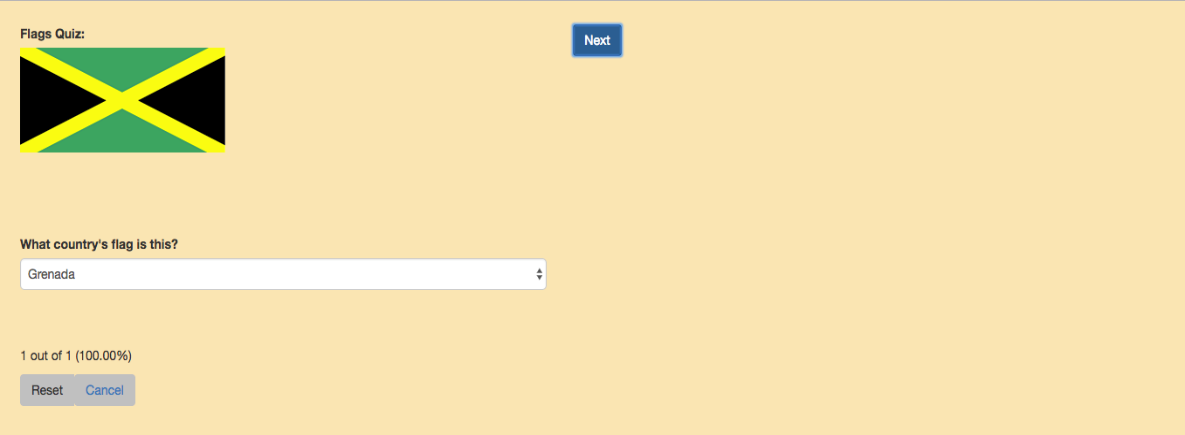
After Wiki Country Button Press



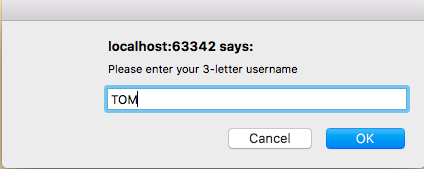
*BONUS: After Quiz Launch*



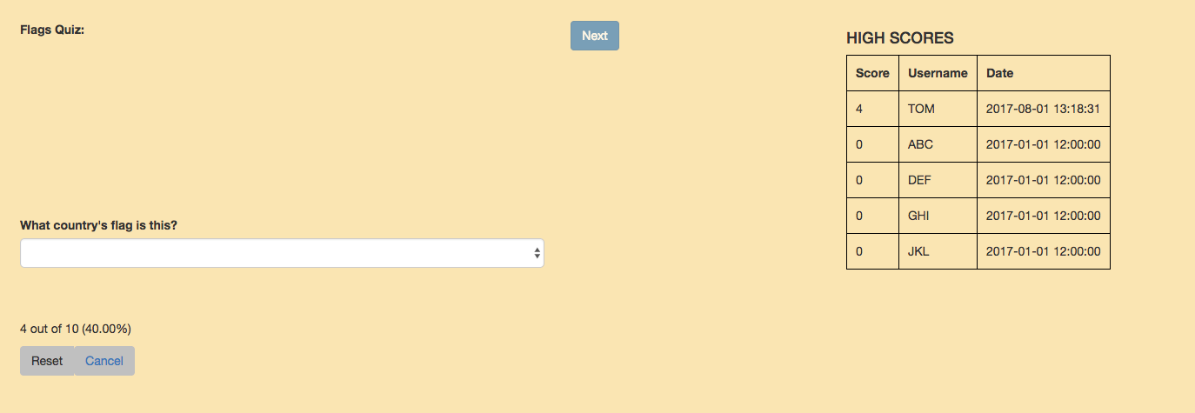
*BONUS: After first Quiz Question Successful*



*BONUS: Prompt for High Score Name Entry*



*BONUS: High Score Report*



### Project Submissions – Phased Marking Explanation

#### Submission Schedule

Each phase of this project (except for the last) will be marked in person, in class with the instructor, during the last classes of the semester. As each phase comes due, your solution to that point will be marked, according to the phased schedule/items listed in the project document.

#### Improving the Previous Phase’s Mark

During each marking phase, you may have the opportunity to improve the marks you received on items from the prior phase. If you passed (60%) the previous phase, you can “make up” points you missed on the previous phase by showing the fixes/improvements you made to correct those errors or omissions. Tracking any prior-phase changes and asking for them to be re-marked is **entirely your own responsibility**. This mark improvement scheme will ONLY apply to items from the previous phase. In other words, if you fix a phase 2 bug, the only time you can get it re-marked is during marking for phase 3. Phase 4 is too late.

#### Cumulative Late Penalties

The phased delivery schedule for this project has been implemented to help students plan and manage their time working on the final project. As such, late penalties will be assessed for each marking phase, and will applied cumulatively to the final project mark. (Ex. A student who was 2 days late for Phase 2, 1 day late for Phase 4, and 1 day late for Phase 5, would incur an overall late penalty of 20% on the final project mark.)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Final Project Rubric - Countries of the World** | | |  |  |  |  |
| **Phase** | **Criteria** | **Unsatisfactory (0 pts)** | **Partially Correct (1 pt)** | **Excellent (2 pts)** | **Marks** | **X** |
| **1** | **Website Layout** | The Website layout was not laid out as required.  Too many errors in the layout exist | Most of the layout matched the requirements | All layout elements are sufficient to meet application requirements and are designed to be user friendly. |  | 2 |
| **2** | **Website Layout Implementation** | Not implemented or contains too many errors. | Implemented, but contains at least one error, or accomplished using inappropriate methods. | Implemented without errors. |  |  |
| **2** | **Pseudocode – Data Handling** | Little to no effort was made to plan out the program using pseudocode | A reasonable but incomplete effort was made to plan out the program using pseudocode | A comprehensive effort was made to plan out the program using pseudocode |  |  |
| **3** | **JSON File Loading** | Country data not loaded, or contains many errors. | Country data is only partially loaded into the application, not loaded in the required manner, or contains some errors. | All country data is correctly loaded from the JSON file when the application starts. |  |  |
| **3** | **Country Data – Basic Display** | More than one element of country data missing or incorrect when displayed. | At least one element of country data missing or incorrect when displayed. | Country name, population, total area and pop. density are all correctly displayed on the web page when expected. |  | 3 |
| **3** | **Country Data – Flags** | Flag images not used/displayed, or contain multiple errors. | Flag images are displayed, but with some errors. | Flag images properly displayed when expected, with no errors. |  |  |
| **3** | **Total Area** | Total area functionality not implemented, or contains many errors. | Total area can be displayed in either unit of measure, but related web form controls do not update as expected, or contain small errors in calculations | Total area can be displayed in either unit of measure, as per user preference. All related web form controls update as expected, with no errors in calculations |  | 2 |
| **3** | **Wikipedia Page Functionality** | Not implemented or contains too many errors. | Implemented, but contains at least one error, or accomplished using inappropriate methods. | A new browser tab can be launched to display the selected country’s Wikipedia page via a button/link on the main page. |  | 2 |
| ***Bonus 4*** | ***Pseudocode – Quiz & High Score*** | *Little to no effort was made to plan out the program using pseudocode.* | *A reasonable but incomplete effort was made to plan the program using pseudocode* | *A comprehensive effort was made to plan out the program using pseudocode* |  |  |
| ***Bonus 4*** | ***Flags Quiz – Quiz Functionality*** | *Not implemented or contains too many errors.* | *Implemented, but contains at least one error, or accomplished using inappropriate methods.* | *Quiz web page functions as expected:*   * *Random flag image displays* * *Random country name options shown, with no duplicates* * *Quiz progress/score tracked* * *Quiz can be reset or cancelled* |  | *2* |
| ***Bonus 4*** | ***Flags Quiz – High Score***  ***Functionality*** | *Not implemented or contains too many errors.* | *Implemented, but contains at least one error, or accomplished using inappropriate methods.* | *High score saving functions as expected:*   * *When high score achieved, user prompted to enter initials.* * *New high score saved appropriately.* * *New high score bumps lowest score (only ever save 5 scores).* * *High score summary displayed at end of quiz, whether or not a new high score was achieved.* |  | *2* |
| **3** | **Variable Scope** | Proper scope not used in most cases, or at all. | Proper scope has been used in most, but not all cases. | Proper scope has been considered and used where appropriate throughout the application code. |  |  |
| **3** | **Program Structure** | Application is not well structured, or has more than one required function missing or incorrectly used. | Application is reasonably structured, but contains some inconsistencies, or at least one required function is missing or incorrectly used. | Application structure is well thought out and adheres to required standards. All required functions are included and work together as expected. |  | 2 |
| **3** | **Data Formatting** | Very little or no formatting used when displaying data. | Most data uses appropriate formatting when displayed, but not all. | All data displayed to user uses appropriate formatting where appropriate. |  |  |
| **3** | **Error Handling** | Minimal or no error handling used in program. | Error handling used in most cases, but not in all, or user can crash the program, or useful error messaging not communicated to the user. | All reasonably-predictable errors are handled by the application. No program errors encountered during typical use and user informed of any problems via descriptive and helpful message box messages. |  |  |
| **3** | **Comments & Best Coding Practices**  (At least 60% of the functional requirements must be complete) | Little to no organizational or explanatory comments used.  No apparent naming convention was followed or was inconsistently applied. Source code was poorly formatted. | Some organizational or explanatory comments are used, some are meaningful and easily understood. A naming convention was used for part of the program, but deviated often. Improvements could be made. | Organizational or explanatory comments are used extensively, most are meaningful and easily understood. A consistent naming convention was used for most of the program and deviated very little. Source code was clean, consistently well-formatted and easy to read. |  |  |
|  |  |  |  | **Total:** |  | **/38** |