# 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**: Basic web page design implemented to the point that the main web page with the country form launches and has appropriate controls to display the desired Country information and flag image files as displayed in the sample application. No JavaScript functionality is needed at this point.

**Phase 2**: The loading of the list of countries from the supplied Countries.json file implemented.

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

### Project Instructions

#### Use WebStorm 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. So, when you have completed all assignment work, put a “Ready for Marking” comment on the last code committed into GitHub. It is this “Ready for Marking” commit 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.**

*In case of GitHub emergency:*

If your GitHub repository has become corrupt or you have some other compelling reason to be unable to use GitHub, you can follow the below routine to submit your assignment. You normally do not do this. The D2L dropbox for the assignment only exists to give you a due date in the D2L calendar and as a backup submission method in **emergencies**.

Once you’ve completed all required programs for the assignment, add all .html, .js and supporting files (including the original Countries.json file) to a single .ZIP file. The .ZIP file name should include your name, W#, section # and Final Project. (ex. DoeJohn\_w0123456\_702\_FinalProject.zip).

Upload the .ZIP file to the D2L dropbox labelled **Final Project**, which can be found under the Assessments🡪Dropboxes link.

### Evaluation

To insure the greatest chance of success on this assignment, be sure to check the marking rubrics at the end of this document or in D2L. The rubrics contain 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.
* **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.

### 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.

## 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 web application.

**The following screenshots are meant as a guide, but you may pursue your own design for the web pages. The instructor used Twitter Bootstrap to basically style the forms in these examples and you might decide to do the same.**

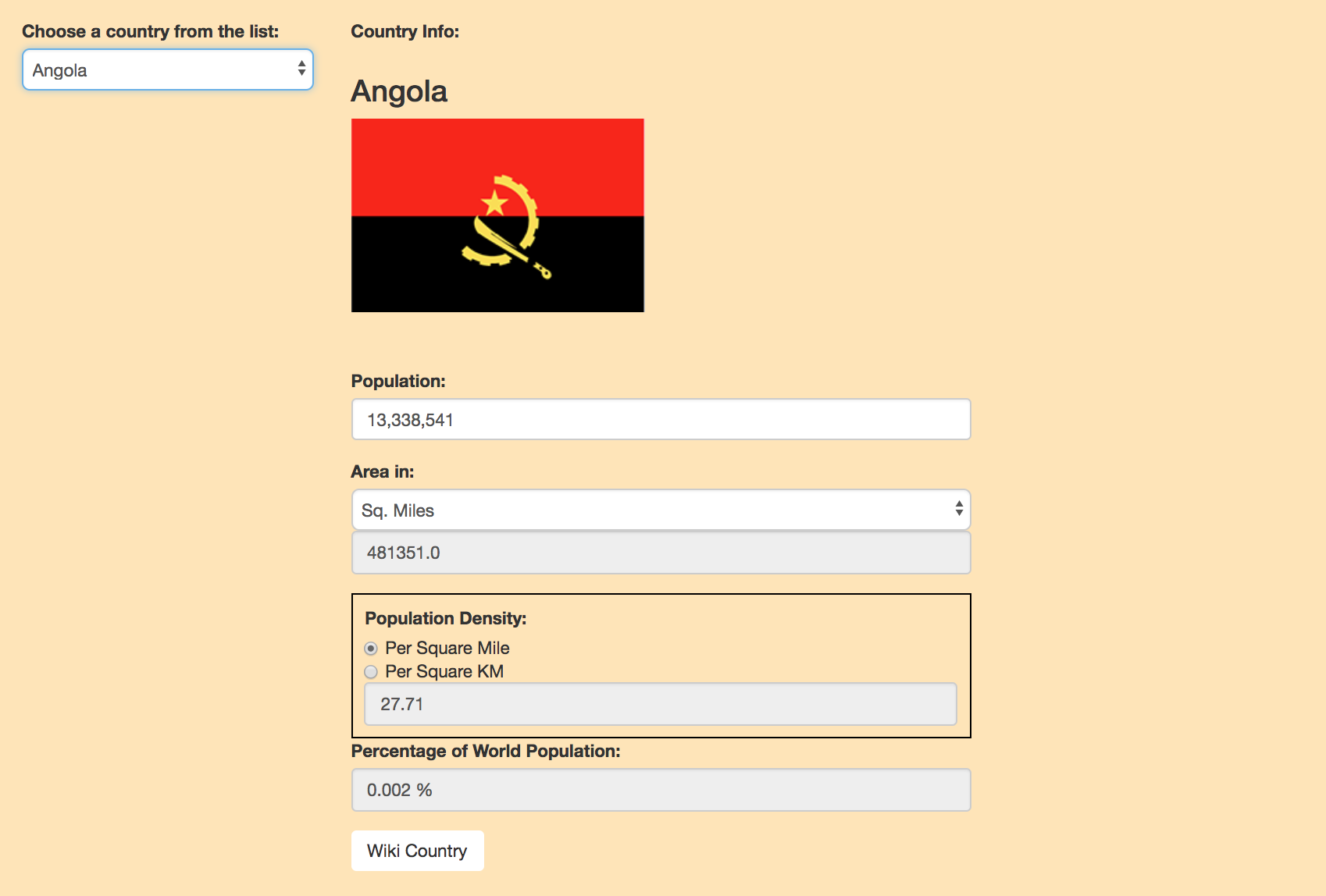
Efforts should be made to:

* Present clean, intuitive and user-friendly forms, with descriptive labels and acceptable layout.
* Include data validation wherever appropriate.
* 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.

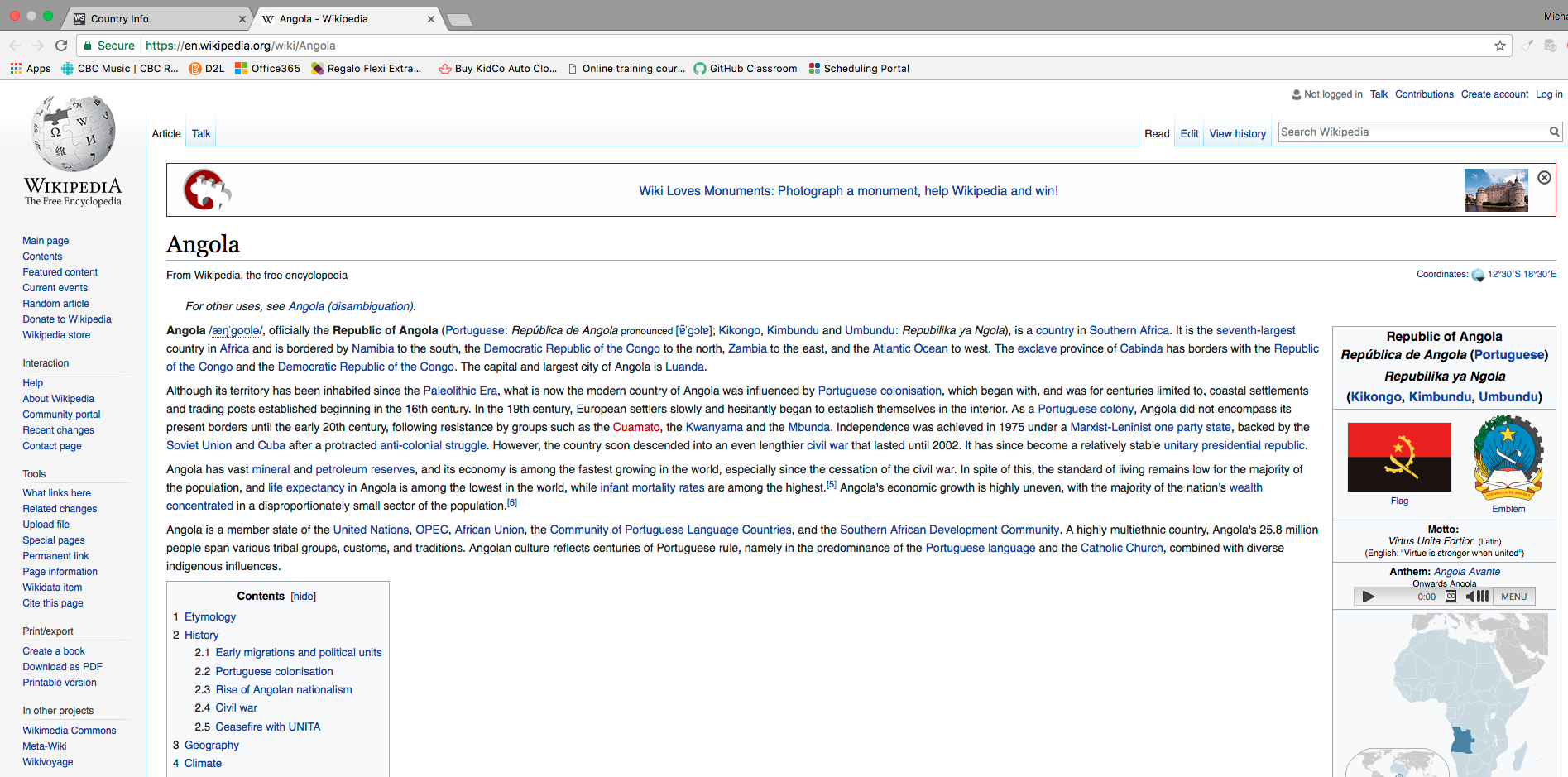
Sample Screenshots

|  |  |
| --- | --- |
| At startup, with country details area hidden | Selecting a country from the select box |
|  |  |
|  |  |

After a country has been selected from the select box



After Wiki Country Button Press



### 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 1 bug, the only time you can get it re-marked is during marking for phase 2. Phase 3 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 Implementation** | The Website layout was not laid out as required.  Not implemented or contains too many errors. | Most of the layout matched the requirements  Implemented, but contains at least one error, or accomplished using inappropriate methods. | All layout elements are sufficient to meet application requirements and are designed to be user friendly.  Implemented without errors. |  |  |
| **2** | **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. |  | 2 |
| **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. |  | 2 |
| **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 |
| **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. |  |  |
| **3** | **Data Validation** | No data validation used, or contains multiple errors. | Data validation not used wherever appropriate. User can cause an error through incorrect data entry. | Data validation used wherever appropriate. User cannot cause errors through incorrect data entry. |  |  |
| **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:** |  | **/30** |