GEOG 487 - Web Mapping Assignment

Task 1:

A Samsung Galaxy S2x was used to obtain GPS readings for both methods, thus the level of accuracy was the same for both the ArcGIS Collector App and the notebook used to record the latitude and longitude. Typically, the accuracy was ~10 m, but increased to ~15 m in the downtown alleyways between tall buildings with little sky access. The ArcGIS Collector App used to record is very user-friendly and simplified. It's minimalist user interface would make it a highly effective tool for casual users, as it guides them through the workflow However, this guided workflow can be restrictive to those with experienced with GIS and mapping, as it is difficult to break away from this workflow to accomplish tasks such as altering the GPS accuracy minimum. There is also a distinct lack of advanced elements, such as the inability to create line features. The main drawback to only using point features is that the size and shape of sidewalk hazards is not distinguishable. For example, if a tripping hazard runs the length of a block, a single point feature cannot capture this. However, some of this information can be portrayed in the text description of a point, or as an attached image popup. Also, a web map containing solely point features is easier to spot general trends on, rather than if it were a mix of points, lines, and polygons.

The hazard domain used for the project is comprehensive for most of the existing sidewalk defects, but it is unable to provide any of the detail. For example, a 1 cm tall tripping hazard would be classified the same way as a 10 cm tall tripping hazard, despite the two have a very different severity. This issue could be avoided by giving ranges to the hazard ID's such as "Tripping Hazard < 5 cm" and "Tripping Hazard > 5 cm" to categorize the severity. Also, it would be sensible to include an "Other" category as a safeguard, to prevent a situation in which there exists an accessibility issue not being contained within the hazard domain.

When using crowdsourced data, there is an increased risk of error in the point features collected, as inexperienced users may create erroneous points. Hazard points collected by a variety of users could be mislabeled, or even purposely falsified. This risk can be minimized by having clear a clear hazard domain, and by looking for general trends in the points collected, rather than specific data. For example, if an area on the sidewalk has a tight cluster of 10 "Tripping Hazard" points and a single "Unevenness" point, then it is likely that a tripping hazard exists at that location.

Task 2:

	OpenLayers (2 vs. 3)	Leaflet	ESRI (ArcGIS) Javascript	Web App Builder
Development Environment	4/10 (7/10 for advanced users)	9/10	8/10	10/10
Organizational Requirements	9/10 (NGO), 6/10 (corporate)	10/10 (NGO), 6/10 (corporate)	10/10 (corporate), 8/10 (NGO)	8/10
Functionality	7/10 (2), 8/10 (3)	8/10	8/10	5/10
Performance	5/10 (2), 8/10 (3)	9/10	7/10	6/10

For each of the four web mapping API, we have divided our opinions and review of each by the following table above. a) Development Environment, b) Organizational Requirements, c) Functionality, d) Performance.

Leaflet:
Leaflet Sidewalk Issues.html

a) Leaflet is the leading open-source javascript for mobile-friendly interactive map. It has all the mapping features that most developers need. Leaflet is designed for simplicity of coding, good performance and usability for users. It works efficiently throughout all major desktop and mobile platforms. It has an easy to use and well-documented API along with simple, and readable source code. It was pretty easy to learn the basic tasks as it is the simplest language among the three web mapping programs we were required to learn. However, the more advanced functionality was a little more difficult to learn especially importing the GeoJSON as there were incompatibilities with the coordinate system used by ArcGIS Online. Many of the features we needed, like changing markers, creating popups (and loading feature properties within them), loading OpenStreetMap, and importing GeoJSON files (after a fix) were documented well, and were easy to implement. Leaflet also has advanced capabilities that is demonstrated with various other third-party projects, such as GeoJSONLint which inputs a copy/pasted text block of GeoJSON code, and outputs it on a Leaflet map.

b) There weren't any financial costs associated with the Leaflet option as it is distributed as an open-source software, however Leaflet cannot be found responsible for any damages with use of this software. Leaflet is an open-source software package under the Open Data Commons Open Database License (ODbL) by the OpenStreetMap Foundation (OSMF), thus it is free to use as long as the data is credited towards the OSMF. Any alterations with the code or OpenStreetMap data must be licensed with the ODbL agreement. While Leaflet is free to download, any modifications or variants of distribution must be uploaded with the license, thus if the Town of Kitchener would prefer to privatize any technologies or code that will be implemented with Leaflet, they should look for an alternative.

c) All of the capabilities needed for the task was available. Leaflet is provided with libraries and tutorials that allow the user to create and import (via JavaScript) GeoJSON files, and attribute them based on their properties. However, for efficiency, one capability for leaflet was missing which was importing the GeoJSON generated by ArcGIS Online of points into the basemap as a layer so the points has to be converted from CSV to GeoJSON and we were unable to import all 639 points as it wouldn't be efficient and it would have taken a lot longer than importing only our points. This can be seen with GeoJSONLint (powered by Leaflet), where inputting the GeoJSON of the sidewalk data generates bad data. This approach must be done as the GeoJSON file exported from the ArcGIS Online management system was not compatible with Leaflet, as the coordinate system was different. By exporting a CSV and converting it to a GeoJSON with the ogr2ogr web client (and then formatting it as a JavaScript), the transfer was easy. The capability produced the output, we expected but as explained above, it wasn't as efficient. Leaflet when compared to Esri JavaScript and OpenLayers was in our experience was the easiest to learn, immediately implement with limited web programming knowledge, and has the most features that have straightforward directions for our assigned task.

d) The maps load and refresh pretty quickly so the code did work properly. The major limitation that Leaflet has would be likely due to Internet download speed or computer performance issues, all which would not be inclusive to Leaflet, but the other web mapping API as well.

ESRI Javascript API:

a) ESRI has a comprehensive database of examples, but has little commentary explaining them, as it assumes that the user already has an existing knowledge of using HTML, javascript, and CSS to perform web mapping tasks. While this resource is very useful for experienced users who can pull what they need from an example without assistance, untrained users will struggle to understand the code.
Difficulties were encountered in converting these examples to working code. When run off the website, the examples ran as indicated. However, upon downloading and running the code off a computer, they failed to work.

b) The ESRI Javascript API itself is free for use, but only when being used with ArcGIS products, such as ArcGIS online runs on an annual subscription based payment model, based upon the number of users and credits. Credits are used for storage, analytics, and demographics. For a sizeable organization such as the City of Kitchener to use ArcGIS Online, it would cost approximately $10,000, providing access to 50 users and 10,000 credits. However, for a community run NGO, ArcGIS Online would be entirely free, using the Free Development and Testing plan, which is available to public, non-revenue generating, non-government usage. The ArcGIS API does not have the limitation of modifications to be released as open-source, as the user cannot distribute this API, is free to modifying the API as needed, and is not part of the agreement to redistribute the code, and thus is preferred if they have any in-house technology or code they wish to implement.

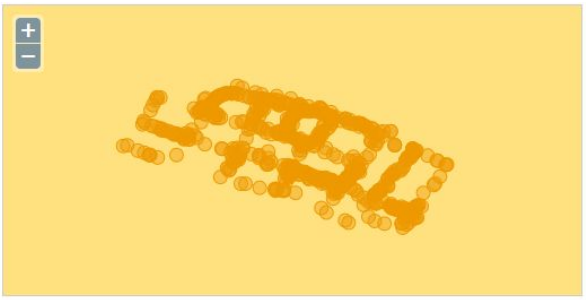
c) ArcGIS Online is capable of fulfilling all the function requirements of the project, although in a more complex manner than the others. However, this complexity is the result of a wide range of customization and options, which sacrifice ease-of-use for giving more control to the user. For the most-part, the output matched what was expected.

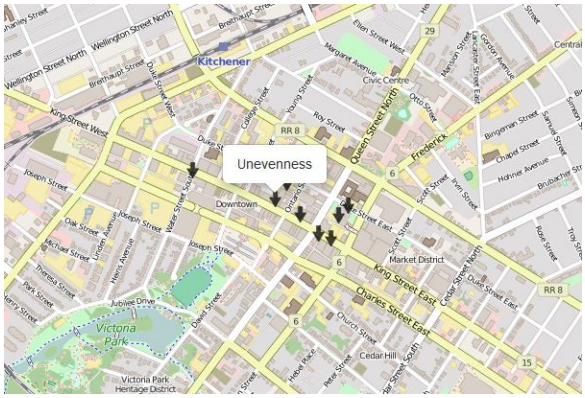
d) The maps load quickly and operates smoothly, but is slightly behind the load times of Leaflet. However, if a larger and more complex dataset is used, load times may begin to severely impair the end-user experience.

OpenLayers:

OpenLayer 2 Sidewalk Issues.html
2 (GeoJSON uploaded, no popups)

OpenLayer 3 Sidewalk Issues.html
3 (custom points, popups and custom icon)

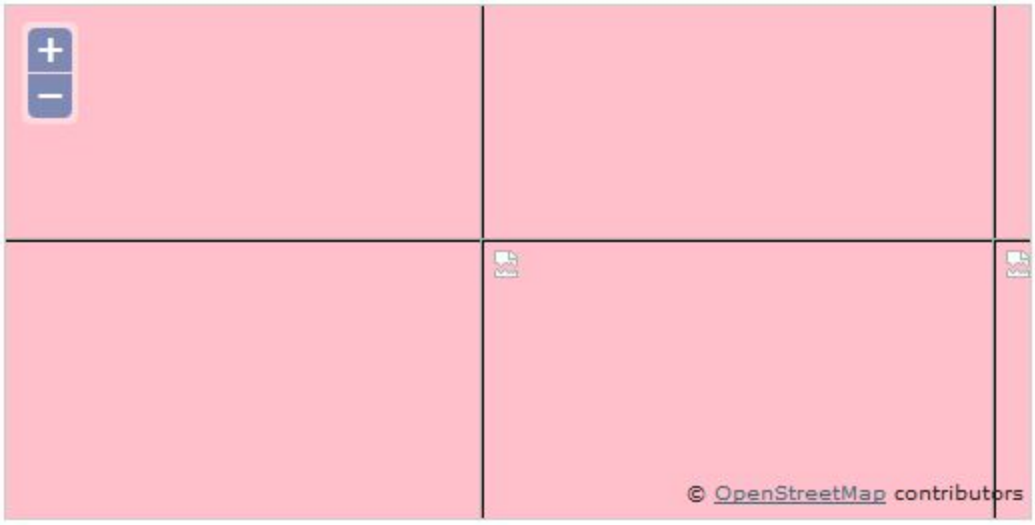




a) Compared to Leaflet, OpenLayers does not offer as many features and integrated libraries that allow for easy implementation, nor do they offer a tutorial to run the program. OpenLayers is likely developed for more experienced users of JavaScript, as much of the tutorials rely on the user understanding the API. OpenLayers does offer a page dedicated to various examples, however due to the inconsistency of the formatting, users who are not experienced with JavaScript, HTML or OpenLayers will have trouble trying to implement or combine the example code offered. Additionally, OpenStack has two main versions that are used and well supported, both which have different libraries that make each one incompatible with one another. While the OpenLayers website separates the support, using external support may be more difficult as some of the legacy version 2 code may not be differentiated from version 3. OpenLayers was the most difficult of the three to learn, and is the one we had the most problem running into.

b) OpenLayers is licensed under the BSD 2-Clause License, which allows the party or user to use OpenLayers for commercial use, modify the client to their needs, re-distribute, and place a warranty on the software. However, OpenLayers will not hold liability if any damages were to occur when using the software, and the user must include and retain the copyright and license used when working with OpenLayers. OpenLayers is free to use, however, any modifications must be open-sourced, thus similar to Leaflet, if there are any plans on privatizing code, then an alternative should be searched for.

c) While through examples we found that the capabilities of OpenLayers met our expectations, the implementation process was much more difficult, and thus we were not able to complete our task in one project. We worked with version 2 and 3 of OpenLayers, where with version 2, we were able to import the GeoJSON, while with version 3, we were able to get a working popup with a custom icon, however importing GeoJSON and changing the style to was not as effective. Many problems arised with OpenLayers. For example, with version 2, loading OpenStreetMap on a local drive did not function as well as intended, where the tilemap loaded only errors (see example below).

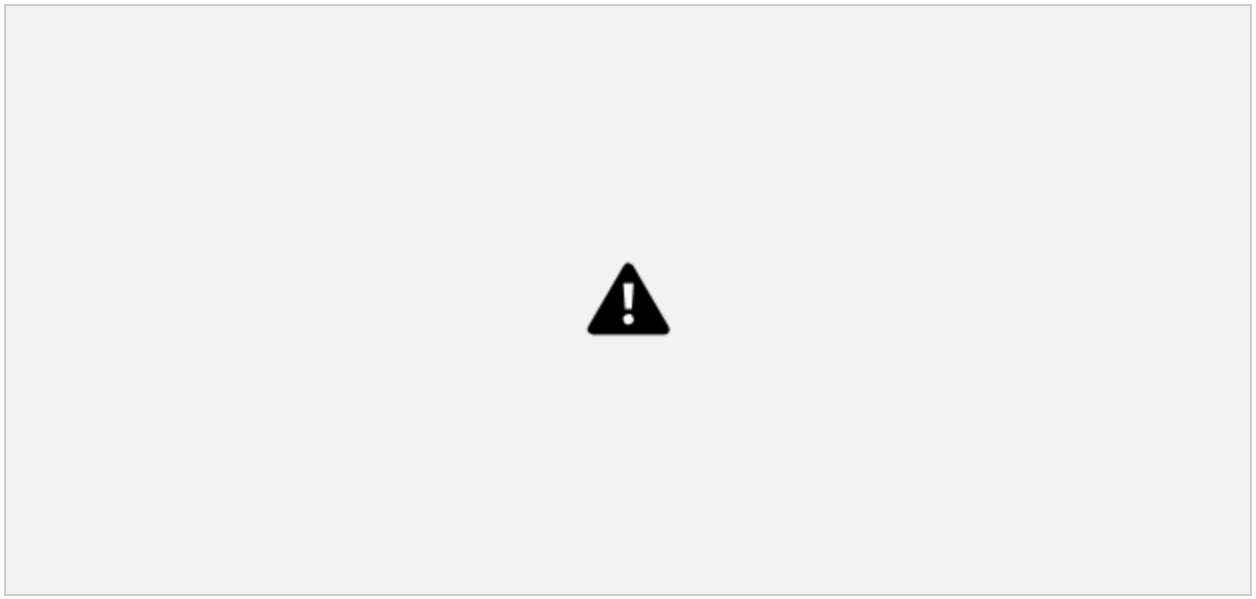


Additionally, with the map we used with our OpenLayers 2, there is a maximum zoom extent that was not preset by us. Much of the features that are necessary for us to complete our assignment (popups, changing and accessing features in the GeoJSON) were much more difficult to work with.

OpenLayers 2 however has the same issue with Leaflet where conventional XY data must be used over a special type that ArcGIS Online outputs, thus the same process must be done with CSV. Unlike Leaflet and OpenLayers 2 however, OpenLayers 3 is compatible with our GeoJSON export from ArcGIS Online. Importing OpenLayer 3 into jsfiddle.com and loading the GeoJSON generated by ArcGIS Online was functional.

d) Like with the previous two web mapping software, OpenLayers was able to load vectors and the map with ease. However, as mentioned above, OpenLayers was unable to map OpenStreetMap when called, and the zooming functionality was limited when trying to import the OpenLayers WMS (see OpenLayers 2). OpenLayers 3 was more functional with OpenStreetMap. Likewise, performance is dependent on the quality of the scripts developed and implemented.

Web AppBuilder:
https://uwaterloo.maps.arcgis.com/apps/webappviewer/index.html?id=61e59f2fe6064486a98c120974e91f7e
https://uwaterloo.maps.arcgis.com/apps/webappviewer/index.html?id=bf3fd54feaa248f3bd63452ec3c22d3e



a) Web AppBuilder is a very streamlined process and can easily be used by an inexperienced user to display data in a basic web map within minutes. It has a straightforward tutorial, and an in-depth help guide that is geared towards a brand new user who has little to no experience in web mapping applications.

b) Using Web AppBuilder commercially follows the same process as the ESRI Javascript API, using the credit system.

c) While it excels at its basic functions, it is extremely limited in its functionality. It was capable of displaying a feature layer, but simple tasks such as drawing new polygon features are outside of its capabilities, making it a very poor choice for an experienced user who wants control over their web application.

d) Web AppBuilder was the slowest to load, being slighty slower to load than the ESRI Javascript API, and far slower than both Leaflet and OpenLayers. A larger dataset could lead to long loading times, making it a poor choice for any large scale projects.

Conclusion:

This “brief foray” has taught us that using the ESRI JavaScript API from ArcGIS Online is a long and complicated process which may not be as efficient. However, Leaflet and OpenLayers can do just about the same amount of work. We have concluded that Leaflet for this assignment has

the best written documentation and ease of implementation, and thus is a good alternative to the ESRI JavaScript API or the Web App.