

# Caminos de Agua Independence Watershed map maintenance

20<sup>th</sup> October, 2016

aaron krupp – [askrupp@gmail.com](mailto:askrupp@gmail.com)

(Updated 24<sup>th</sup> Oct, 2016 to reflect changes in the code structure).

## What is the map?

The map shows a point for every water sample *Caminos de Agua* (and perhaps some other groups) has/have ever tested in the *Independencia* watershed that includes San Miguel de Allende, Dolores Hidalgo, San Luis de la Paz, and many rural communities in Guanajuato, Mexico. Points are color-coded by risk level for various contaminants: fluoride, arsenic, etc. When you select a point, all of the historical data for the site, back as far as 2012, is displayed. When you click these points, or points without historical data, more details are displayed. Some of the points' pop-up info windows have links to water quality monitoring reports done by *Caminos*.

The map is a combination of html, css, sql, javascript, and various javascript libraries (leaflet, carto, and stamen). If that means nothing to you, no worries, but you probably shouldn't be making any functional changes to the map, unless you have significant non-web coding experience or feel like sinking a bunch of time into learning this stuff. That said, it's totally do-able if you've got the time.

## Where is the data?

The data lives in a .JSON file in Caminos' public folder in dropbox. The folder is called “Caminos public.” When you change that .json file, the map automatically uploads the new data. For more info on uploading, scroll down to the “**how do I upload new data**” section.

## Where is the code?

The code lives on GitHub.com. GitHub is a free file sharing and editing system that preserves changes and history, allows multiple people to simultaneously work on projects, and, most importantly for us, hosts and runs web code. For editing the code, see “**getting started with code changes**” section below.

## What's with all the files?

Excellent question. When GitHub hosts a webpage, it only looks at one file, the one called “index.html.” But in that file, you can tell the browser to look at other files. So in our case, “index.html” is the main page but “index.html” tells the browser to look at the rest of the files, too. A few brief descriptions:

1. index.html --- the main html file for the map
2. map\_styles.css --- stylesheet formatting for some of the elements like the dropdown menu, legends, and popup windows
3. global.js --- a javascript file that defines a bunch of global variables (good form, yeah yeah...)
4. display\_*Language*.js – (for now either display\_English.js or display\_Spanish.js) holds all the text that's displayed to the user for international translation. There's a line in index.html that looks like:

```
<script src="display_Language.js" type="text/javascript"></script>
```

When you want to change the language, make sure that the correct “display\_*Language*.js” file is uploaded on GitHub and that on this line in “index.html”, you enter the correct Language so

that “display\_ *Language.js*” matches the language file.

## How do I upload new data?

1. Go to the GitHub.com page (if you don't know what this is, please read **where is the code?** above. Sign into GitHub with the Caminos De Agua account.

2. Open the file “global.js” and find the line that looks like:

```
var DATA_URL = "https://dl.dropboxusercontent.com/s/ywihdduzt1estgh/ask\_test\_dataset.JSON"
```

This is a link that tells the code where to get the dataset. In this case, it's a Dropbox link to the file “ask\_test\_dataset.JSON” which is the file I'm using for testing. You have two options:

- (a) Change this link so that it points to your new dataset [NOTE: IF YOU DO THIS, YOU MUST DO IT FOR BOTH THE ENGLISH AND SPANISH VERSIONS!!!] or
- (b) Remove my dataset from the dropbox folder, replace it with yours, then rename yours with the same name that mine had.

From all the capitalized text above, you probably get that I recommend following option (b), so the rest of this tutorial will assume that you're going to use the existing dataset name, “ask\_test\_dataset.JSON”. However, if you do choose to enter a new file URL, when you copy the public URL from dropbox, it will be different from the above link. See **how do I adjust the Dropbox link?** section below for the information on linking to dropbox.

3. Open the Caminos public dropbox folder, “Caminos Public” either online or on your computer.

4. Find the old file (in this case “ask\_test\_dataset.JSON”).

5. Now comes the fun part: Go into your new dataset. Make sure it's columns are labeled the same way as the variable DATA\_NAMES in global.js.

6. Sort your dataset in excel BY DATE IN DECREASING ORDER. This will put the most recent samples at the top and is *really important* for the spidering to work correctly.

7. Export or save your excel dataset as a CSV (comma-separated values) file to somewhere on your computer. Maybe your desktop. This isn't important, it's just a placeholder.

8. Go to <http://www.csvjson.com/csv2json>, select the CSV file you just made and click “> Convert”. This will convert your CSV file into a JSON file, the file type we need for the code.

9. But the code is a bit particular and doesn't like irregular characters. So click in the “JSON” window, ctrl+a to select all, then copy the JSON.

10. Go to <http://jsonlint.com/> and paste the JSON into the big white box. Click “Validate JSON”. This site checks JSON files for proper formatting. It will tell you “Valid JSON” in a green bar if you're good to go. If not, it will highlight the offending lines in red. Usually the issues are irregular characters, like “+” or “<”.

11. Go back to your initial spreadsheet in excel. Get rid of all of the offending characters.
12. Repeat steps 6 – 10 until you get the “Valid JSON” message.
13. Once your JSON is valid, copy+paste it into your favorite text editor (I really like Notepad++, it does a good job color-coding code and files, but Notepad or TextEdit are fine).
14. Now, you want to get this file into the same Dropbox folder as the old file, with the same name. But we can't have two files with the same name. So I recommend now changing the old file to something like “ask\_test\_datasheet\_IM\_OLD\_PLEASE\_DELETE\_ME.JSON” or something similarly obnoxious. Once you've done that, you can save your new file with the same name as the old one, in this case, “ask\_test\_dataset.JSON”.
15. As long as you're connected to the internet, your Dropbox should sync within a minute or two. After your dropbox is up to date, refresh the map and – voila – your new dataset should be mapped!
16. Once you've verified that all your hard work has paid off, delete the old dataset (or, safer yet, move it out of the dropbox folder in case you need it later). Good work!

## **How do I adjust the Dropbox Link?**

Each dropbox account can have access to one public folder. Anyone with the correct link can access anything stored in public dropbox folder. For example, if you go to the Caminos public folder, there's a PNG image of a blue circle with an 'x' in the middle. We'll use this as an example.

### If you're using Dropbox on your computer:

1. Go into the Caminos public folder and right click on the file named “xButton\_blue.png”.
2. Click “Copy Dropbox link”.
3. Paste that link somewhere.

### If you're using Dropbox online:

1. Go to the Caminos public folder. Next to the file “xButton\_blue.png” click “Share”
2. If there is an option to “create link,” click it.
3. If you don't see “create link” it means a link already exists. In either case, click “Copy link”
4. Paste that link somewhere.

Great! Now you have a link. It should look something like this:

[https://www.dropbox.com/s/bjyqgvkgbmb9xlu/xButton\\_blue.png?dl=0](https://www.dropbox.com/s/bjyqgvkgbmb9xlu/xButton_blue.png?dl=0)

It could look slightly different based on whether you're using Dropbox on your computer or online, but it shouldn't matter. If you put that link into a browser, you should see a picture of the blue-x appear. Success! However this link has too much information for the map to use. We don't want any of the Dropbox nonsense, just a pure, unadulterated version of our PNG image.

So. We're going to modify the above link a bit. That “?dl=0” bit on the end? Doesn't really matter to us. So let's get rid of it. Now we have:

[https://www.dropbox.com/s/bjyqgvkgbmb9xlu/xButton\\_blue.png](https://www.dropbox.com/s/bjyqgvkgbmb9xlu/xButton_blue.png)

Great. But we don't want a dropbox link, we just want the content. So highlight “[www.dropbox.com](https://www.dropbox.com/s/bjyqgvkgbmb9xlu/xButton_blue.png)” and replace it with “dl.dropboxusercontent.com”. This gives us:

[https://dl.dropboxusercontent.com/s/bjyqgvkgbmb9xlu/xButton\\_blue.png](https://dl.dropboxusercontent.com/s/bjyqgvkgbmb9xlu/xButton_blue.png)

Try this link now. It should give you a picture of that same blue button, but all alone, no Dropbox content to be seen. Perfect! For any images/videos/spreadsheets that you're including in your code, make sure you follow this process to modify the links.

## Getting started with code changes

The code (that the internet browser uses when you look at the map online) is stored on GitHub. GitHub is a public site where people can store, update, and publish code. Our code is available to the public, but can be modified by invitation only. *Caminos de Agua* has a GitHub account. To change the code, login to our account or make your own account. Once you've made your own account, the *Caminos* account can invite you to modify its repository. (The invitation goes directly to your email.)

Once you've accepted *Caminos'* invitation and have permission to edit the repository (it's called “caminoideagua/Independence-Watershed-Point-Map-English”) or you're on the *Caminos* GitHub account, you can start making changes to the code!

When GitHub hosts a website, it goes into the “master” branch of the repository and looks for a file called “index.html”. It then creates a site from this HTML file. Find that file and take a look. Whew! Super long. That's cause I suck. My bad. However, there's also lots of grey text. Whew! That's cause I like you. The grey text is comments that describe what's going on in the code. You should be able to read through the comments and have a general understanding of roughly what is happening in each section of code.

[For more help with GitHub's repositories, branches, pushes/pulls or getting GitHub on your personal desktop (the same way as dropbox!) google “GitHub hello-world tutorial.” and play around with your personal account for a while before trying to make any changes to the map].

If if this is confusing so far and/or you don't have experience with web development, check out some tutorials on the structure of an HTML document, javascript, and CSS.

Basically, the document is broken up into chunks:

At the beginning, we bring in a bunch of other scripts/libraries that we'll use to accomplish specific, mapping-related tasks. Then we have a bunch of CSS that describes how lots of the objects on the map are going to look. After the CSS, we have the HTML body, where some objects are defined, like the legends, the dropdown menu, and the caminos logo. The HTML also tells the browser what to do when these objects are clicked on. After the HTML, we have a <script> tag. This is divided into two sections. The top section (in which you'll never see the word, 'function') is code the runs when the website load. This mostly defines visual parameters and lets the map grab some images. The rest of the stuff inside the <script> tag are a set of functions. You probably won't need to mess with these, but if you do, see the full functional descriptions in the comments or in “**how do I make drastic functional changes,**” below.

In GitHub, you can edit the code directly. However, the nice thing about web-development is that you

can run your code in any browser, like Chrome, Firefox, etc. So I'd recommend first downloading the code from GitHub (or just copy+pasting it into your favorite text editor – I'd recommend Notepad++, it gives you lots of nice colors – and saving the file). Once you have the code, you can open it in your favorite web browser and, voila! It runs! I like working in Mozilla Firefox, they have a nice debugger and console under Tools > Web Development. Once you're happy that you understand how the code works and how to modify it, you can start making changes in GitHub. Woohoo, best of luck!

### **How do I get the map onto a website?**

Since the code is hosted on github, this one is easy. If you're working on code in a github repository, you can set that repository to create a webpage from its code. (Do so in “settings” once you're in the repository you care about. GitHub takes the file called “index.html” and turns it into a webpage at the link: [https://\[user\].github.io/\[repository\\_name\]](https://[user].github.io/[repository_name]). So for us, this map is hosted at <https://caminosdeagua.github.io/Independence-Watershed-Point-Map-English>. If you want to view the map, just enter that link into a browser!

To embed the map in a website, you have a few options. If you're hard-coding a website, you can probably figure it out. Website builders like squarespace, wix, wordpress, etc., often have a box/element that you can copy+paste code into. For embedding this map, you'd probably write a bit of structural code telling the website how to display the map, then put the link.

For squarespace or wix, copy+paste the following into your code box:

```
<iframe width="100%" height="520" frameborder="0"
src="https://caminosdeagua.github.io/Independence-Watershed-Point-Map-English/" allowfullscreen
webkitallowfullscreen mozallowfullscreen oallowfullscreen msallowfullscreen></iframe>
```

Feel free to adjust the parameters here to size the map appropriately for your site.

### **How do I make small visual changes?**

This tutorial assumes you already have a GitHub account (or are using the *Caminos de Agua* one) and have taken at least 10 or 15 minutes to familiarize yourself with GitHub. If you don't know what GitHub is, please go up and read “**Getting started with code changes**” above.

This section explains how to make minor visual changes to the map. These include changing:

- point sizes
- point colors
- the styles/opacity of the points and lines
- the styles of spider x-out button
- the styles of labels and pop-up info windows
- the default zoom and zoom bounds on the map
- the map's base tiles
- the carto attribution in the bottom bar

To make tweaks to the map's current display, login to your (or *Caminos*') github account. Open the file “index.html” (the main file that holds all the code) in the master repository.

The first section of the <body> <script> is a series of global variables. The top of this section looks like:

```

225 by another function, or by a user event like a click or zoom. -->
226 <script>
227
228 ///////////////////////////////////////////////////////////////////
229 //                                     DEFINE GLOBAL VARIABLES
230 //
231 //    IF YOU'RE TRYING TO CHANGE THE FORMATTING / COLORS / STYLES /
232 //    DATASET IN THE EXISTING MAP, YOU SHOULD ONLY NEED TO CHANGE
233 //    STUFF IN THIS SECTION. DON'T CHANGE ANYTHING ELSE IF YOU'RE
234 //    NOT SURE WHAT IT DOES!!!
235 //
236 //    IF YOU WANT TO CHANGE SOMETHING FUNDAMENTAL, LIKE WHICH VARIABLES
237 //    CAN BE PLOTTED WITH THE DROPDOWN MENU, PLEASE REFER TO THE
238 //    TUTORIAL VIDEO POSTED ON TRELLO BEFORE PROCEEDING.
239 //
240 ///////////////////////////////////////////////////////////////////
241
242 var USER = 'cameron-catis'; // User for carto SQL queries
243 var map; // initialize the variable to hold
244
245 var DATASHEET_NAME = 'ask_test_dataset'; // Datasheet name from carto.com for retrieving data
246 var DATA_NAMES = { // And store the titles of the columns
247   date: "date", // (get from carto.com once you import
248   name: "community_name",
249   f: "fluoride",
250   as: "arsenic",
251   lat: "latitude",
252   lng: "longitude",
253   docs: "documents"
254 };
255
256 var MONTHS = ["Jan", "Feb", "Mar", // Array of names of months for displaying
257   "Apr", "May", "Jun", // the date in an accessible, clear format,
258   "Jul", "Aug", "Sep", // even for silly US people who choose to put

```

and this section continues for about 150 lines and almost exclusively defines variables. Please download `index.html` to your computer. There, you can open it in a text editor (I'd recommend downloading Notepad++ at <https://notepad-plus-plus.org/>, but any simple text editor like Notepad that lets you save an `.html` file is fine). Play with the variables IN THIS SECTION ONLY!!!

For example, changing

```
>>> var SMALL_ICON_SIZE = [16,16]
```

to

```
>>> var SMALL_ICON_SIZE = [60,60]
```

will make your points enormous.

Or changing

```
>>> var X_URL = "https://dl.dropboxusercontent.com/s/df3pabfdc7tzt4r/xButton_blue.png";
```

to

```
>>> var X_URL = "http://feelgrafix.com/data_images/out/28/992667-che-guevara.jpg";
```

will replace the x-button that closes the spidered points with a picture of Che Guevara. (You might also want to change the icon size to make that photo larger to adequately appreciate his flowing scruff.

Once you've made a change, save your file and just double click the file on your hard drive to view it in a browser! (Most modern browsers come with debugging tools. In Mozilla Firefox, for example, go to Tools > Web Developer > Web Console to view error messages. These often tell you on what line of your code you made a typo.)

Once your map (now stored on your computer) looks good – and remember, we're only talking about a change or two at a time – make the same edits in the version of `index.html` stored on GitHub. Make sure to test your map before you publish it to make sure you haven't introduced any bugs.

Congratulations, you've changed small visual stuff on the map!

(You can also change some display parameters in the CSS [roughly the 1<sup>st</sup> 100 lines of the code]). Feel free to play around with those as well in the same way described above.

## How do I make drastic functional changes?

This section assumes you already have a GitHub account (or are using the *Caminos de Agua* one) and have taken at least 10 or 15 minutes to familiarize yourself with GitHub. If you don't know what GitHub is, please go up and read “**Getting started with code changes**” above. Although, if you don't have weeks and weeks to sink into this, and aren't already familiar with html, javascript, CSS, and, perhaps, SQL, then making drastic functional changes is probably not worth you taking the time to learn all those things. If you're still committed to making them, read on! Also, if you are a decent web-developer, please change this code! (I'm not a web person, I know very little about web-dev. This code is structured to be pretty easily modified. Many aspects are not optimized. If you're keen to overhaul it and are confident in your ability to do so, be my guest!)

**Just note:** this document doesn't teach you any of those programming languages, it just describes the structure of the code index.html so you know where to look when you want to make changes. It assumes you're a competent computer programmer and are capable of learning what you need from various documentation sources. If that sounds like you, read on!

If you're still keen on adding or removing some of the map's functions, here is a table of contents of all of the map's functions with brief descriptions:

1. `init()`
  - Is called when the map first loads. Calls a couple of other initialization functions
  - No return values
2. `initMap()`
  - Initializes the map object
  - Defines what happens on `zoomStart` and `zoomEnd`
  - No return values
3. `applyBaseMap()`
  - Grabs and applies basemap
  - No return values
4. `loadAndPlotData(contaminantToShow)`
  - Takes in the contaminant to display on the map
  - Reads the JSON data from carto
  - Stores that data in a global var
  - Then calls the functions used to plot points
  - No return values
5. `plotMarker(type, data, contam, data_index)`
  - Takes in:
    - `type` --- string, “base” or “preSpider”
    - `data` --- the full JSON dataset
    - `contam` --- int/number, the index of the active contaminant
    - `data_index` --- the index of the relevant datapoint in 'data'

- Plots the relevant marks
  - Binds the relevant popups and labels
  - No return values
6. `getBin(data, bins)`
    - Takes in:
      - `data` --- a single row of the whole JSON array
      - `bins` --- a single array from the global array BINS. This can either be full of integers (in the case of arsenic or fluoride, for example, or can begin with a string and continue with the indices of the contaminants to compare
    - Returns a the bin as an integer  $\geq 0$ .
  7. `openSpider(data, i, contam)`
    - Plot's the spidered points for the clicked-on point
    - No return values
  8. `window.onclick = function(event)`
    - Defines what happens when the window is clicked
    - used to open-close the dropdown menu
    - No return values
  9. `adjustDDText(contam)`
    - Updates the dropdown menu headline text to that of the active contaminant
    - No return values
  10. `toggleDD()`
    - Toggles the dropdown menu open/closed
    - No return values
  11. `getBasePopup(i)`
    - Takes in the index of the relevant row of the JSON data
    - Returns a string to display in the popup infowindow.
  12. `adjustLatLng(lat, lng, total_pts, i)`
    - Takes in:
      - `lat` --- a latitude
      - `lng` --- a longitude
      - `total_pts` --- the total number of points to spider
      - `i` --- the index of this particular point among the rest of the spidered points
    - Computes the location of the *i*th spidered point
    - Returns an `L.latLng()` object with the shifted latitude and longitude.
  13. `presentIn2dArray(array, value)`
    - Takes in:
      - `array` --- any 2d array WITHOUT REPEATING VALUES
      - `value` --- a number
    - Checks to see if value is in array.
    - Returns a 2-element array:
      - `[0]` --- exists --- boolean, true if the value is in the array, false if not
      - `[1]` --- index --- a 2 element array with the coordinates of 'value' in 'array'
  14. `getLabel(type, i)`
    - Takes in:
      - `type` --- string "year", "community", or "hist"
      - `i` --- index of the relevant datapoint in the global JSON data



- Returns
  - The appropriate label --- string
- 15. hideLegend()
  - Hides all legends
  - No return values
- 16. showLegend(contam)
  - Takes in:
    - contam --- int index of the relevant contaminant
  - Displays the legend for that contaminant
  - No return values
- 17. closeSpider()
  - Closes the spider that is open (if one is)
  - No return values
- 18. removePoint(i)
  - Removes the point at index i in the global JSON dataset
  - No return values

There is also a bunch of HTML and CSS code that defines the dropdown menu, *Camino*s logo, and the map legends. Feel free to change those too, their behavior should be relatively self-explanatory.

If you want to change the fundamental behavior of the map, not just the appearance, feel free to modify any of these functions to your heart's content. If you're going to do so, however, please do it responsibly. Use GitHub, save previous working versions, test/debug often, and make sure the version that you're working on isn't publicly displayed on the *Camino*s website. Woohoo, enjoy yourself, and happy coding!!!

### **More questions?**

Please contact aaron at [askrupp@gmail.com](mailto:askrupp@gmail.com) or through his GitHub account, ask53.