Video Title: “Video 4: Reading Rhetmap’s data-processing code”
From “Rhetmap.org: Composing Data for Future Re-Use and Visualization”
by Chris Lindgren and Jim Ridolfo
(Kairos XX.X PraxisWiki)

Okay, we're finally to the point where we can start looking at the data processing code inside of the RhetMap code that I wrote for Ridolfo's market comparison data. I have two windows open. On the left side, we see the website where I host the RhetMap. Dataviz @lingeringcode.github.io/rhetmap-time-series, hyphenated. And on the right side we have a code editor open with the RhetMap code that I wrote inside of it. Before we open up and dig through some of these functions, let's quickly review the console logs that I wrote, that print out to the console on the website. So this is meant for pedagogical purposes. I wrote these console logs inside of the code here.

So if we look. If you inspect the element, open it up, you should see inside the console. In this case, right now, we have index, which is the file, index.html. On line 62, I apparently wrote something that printed out retrieved spreadsheet data, parsed it as a JSON object. So let me just verify that. If I open up the kit data function and scroll down the line 62, we see a console log that indeed, enabled me to print out that message it says, OK retrieve spreadsheet data, new line parsed it as a JSON object, new line and then I even printed out the data itself that it retrieved. Before we really dig into the JavaScript that's the rhythms I set up with the console logs here in the browser that we are going to review first before we take a look at it more closely in the JavaScript code. 'Cause it's important that we understand first the data and how it went from one form to another, to another, and another, it was processed.

All right, so again first I had to get the data. And this should look very familiar on index line 64 I've printed out the data that I retrieved. From the previous video we should recall how this is a JSON object, it starts, it opens with the curly brace to suggest that this is an object. And inside of it we have some key value pairs and what we care about is the "feed". So, here we have "feed" and inside of "feed" that's an object as well. You can see how it's nested. Inside "feed" we have 17 different items in an array of objects. So, here we have the first one which would be implicitly we know as see week one.

So if I close this out by the way, right we have zero through 16 with a length of 17 there's 17 weeks, so this would be the first week and if you recall the things I was curious about and wanted to grab are these ones right here. With these with these, I think I called it gobble-dee-gook. And what I like about, again, Chrome's console and inspector, Firefox might do this, as well. I just use this more, typically. We can even see the nested functions here. We can see feed dot entry, we have the first item zero and then specifically with dot notation we can see feed dot entry dot. Again this particular key of GSX$_CN6CA corresponds to an object with a key of "T" value of string week one dash nine 12, which is the date. According to Ridolfo right, in his table data. And then for that, those table data it's then compared across the years, if you recall okay, here's a week which is a row, and across that row this is for the first week, I'm sorry the first year 2012-13, 13-14, 14-15, 15-16, and so on. Right? So that's how that works.

And that's what I was curious about getting is the stuff inside of feed, entry, and then these particular key value pairs across here. So, but what's interesting right, is that I had to first get the data. So, what we can see here online 62 and 64 is okay I actually received the data. So let's go over to the code now. And this get data function, I'm not gonna belabor every piece of it, I try to document it (laughs) maybe over document it, to provide some means for people to look through it, research what it means to actually request data from a different URL, like I did. And that requires instantiating what we call a "request object" in JavaScript.

So how's your computer talk to another computer and request a page and it's contents? That's what a request does. And this whole function is designed around requesting some kind of data object from a specific uniform resource locator, or link, a URL. And here's how I built the URL with variables up here. So strings, that I pieced together to create Ridolfo's specific feed. And where the magic happens again is inside of, when it actually loads successfully.

So, I again I'm not going to go over the details of this particular function, but what you can see and hopefully start reading based on your ideas of scope, assignments, functions, you know, variables, that are instantiated in given certain kinds of values. Concatenation with these plus signs.

Knowing all of that I hope this helps you understand how, "Oh, I can look up request objects now and see how Lindgren did this. Oh, he had to do a lot of different kinds of checks too." If it's actually successful, a successful message then I can do something. But sometimes it errors out so you need to account for that.

But the important part is I got it I saved to a variable named "Data" so that became a JSON object. 'Cause I parsed this, request object, rendered as response text. It's a method specific to requests. And I parsed it JSON dot parse, I applied that method to it to say I want that to be JSON object so I can read it and format it in format data. So that's why I did what I did there. You can research request objects more on your own time.

But the important part is that I had to get the data from that URL that we reviewed in other videos as well. And made it into specifically a JSON object that I can pass as an argument to format data. So let's close up get data, and look at what's going on inside of format data.

So you can see here, that argument became a parameter. And I just named it the same thing for consistency, you can name it anything. I could have name it Chris (laughs) or just the letter D. But in this case I wanted to be kind of consistent across like how the data is moving from one format or function to the next.

So let's move back to the Chrome browser. And take a look at how I did this. In my console logs, I wrote out on line 92 "the requested spreadsheet data in the JSON format has been passed as a parameter for the format data function."

So let's open that up in the code, line 92. Oh I'm sorry, we're supposed to be in format data now. There we go. So we can see 91 is where the function is defined as format data. I've given it a name and this is where I wrote it out. I just said, okay if you don't quite understand how arguments and parameters work, this is it. I passed it from format, or get data to format data. And look, here it is. I've, this and this are a console log as relying on this parameter right there.

Okay, so before we get mired in some of the details of like why am I defining all of these variables, let's take a look at what happens inside of this. Again the same object is present here as it is there because passed it along. And now inside of it let's see what'd I do? Let me close this up so it's not distracting. In the code I documented with logs again starting on line 162 I wrote out a little log that said "Hey look, I wrote out each of those weeks data and I tried to separate them out per each year. I tried to figure out how can I bundle these in each year because that's the problem that I was facing. How do I create multiple lines with the data that I was given. Because right now they're not bundled by year, they're bundled per week. I don't want a per week line, right? I want a per year line, that is then tracked across weeks.

So that's the data processing problem. If you recall is that I needed to bundle these weekly data entries across 17 weeks per year. So that's a different aggregate perspective, right? I'm not looking per week in this chart, I'm looking at per year, per week. So that's the problem. So I needed to take those those 17 values and based on how I knew they were structured, so again if I look at feed let's do it up here actually 'cause this is I think already listed out for me, yeah under entry recall one or you know I got 17 different weeks and this is year one, 2012-13, and if we open up the second one, right here, again this is for 2012-13 but again it's per week not per year. So I needed to take all of this key value pairs and put them in, an array. A list, a meaningful set of values, so I used a for loop and I bundled them into a variable for each year and here's 2012 and 13, 68 through 205 so all 17 weeks for that year.

Okay, let's take a look at that in the code, so line 162, bear that in mind. I knew I needed these these buckets these these arrays so that's how I remember scope is about inside this function I'm gonna need a space to put those values that are in the JSON object. So I wrote a consistent kind of naming scheme for each one and typed it as an array meaning I said this is definitely gonna be an array a list of items so let's type it as that.

And then what I had to do is essentially go down here and I wrote a for loop and when I'm what I did here is I got the length of the JSON object itself and then I applied that length value to right here so, that I could say start at zero and then iterate across a length of it so that's what this value does here. I guess that's one thing we haven't covered, but if you want to read more about for loops definitely do so. What I essentially did is I program this to for each entry push the specific year data to those array lists. That's how this is working at a top technical level.

How do I go from let's see here this feed data feed and each entry that has this data right here, and then start piecing them out so that I can put one by one kind of plop them in. Think of it as like, I'm taking a little value like this 68, and I'm pushing it to each array. That's what I'm doing here. So in this case, okay let's define first how do I get there?

Give me a sec here.

First, we have to tell JavaScript we need to push something to a specific array. So I knew for 2012 and 13 I have an array that I've created I'm gonna push from the data object, the JSON object, I have to look inside the feed object, so I say data dot feed. Then I have to go another nested level right? Dot entry, and if you remember each one has a value associated with it which is an index zero through 16 in this case so I have to go through each entry which are there are 17 and each entry has also 17 entries.

So I'm programmatically going through each data dot feed dot entry to then push the specific year data at the particular indices of zero through 16 that's what the I becomes. I is a now a dynamic number that changes from zero and it goes through until it gets to 16. Zero through 16 is 17 total right? So that's how we are able to tell the computer I want that specific value at. In this case we have GSX just RC dot dot text value. In this case it'll be this one right here, for the first one of 68 and the next one would be the value of 83. See how its text that that T value T key 83 value.

All right, so stay with me.

This for loop goes through each instance within data dot feed dot entry to push the specific column cell value to the year data that I want. So all the week data for each year I can push specifically, based on the recurrent key value pairs. And let me close that up so we don't get too muddled up I've printed out after the for loop here's an example of what happened. So instead of it now being separated per week, I now have an array that I printed out at line 165. I have 17 values inside of it. From 68 emerges how 68,83 all the way through in the order that I desire as well by the way.

So now I have these nice bundled arrays per year, but now I have to actually label them. I need that label of years for my database, so I wrote a little function called write years within format data. Within that scope I say okay you did all that great. Now go to this function that's defined down here and bind together the year label with that, those arrays. Also remember instantiated these as well. These objects we also might call them dictionaries.

So now I can label the year with a key of year and a value of a text value 2012-13 comma and I want another object in there or another key value pair. Here's where I apply it those week arrays. I give it a key of weeks (wks). And then I add their right right there.

So let's see what happens so at right years, I write them all and check it out, afterwards here's right years that starts and closes. Let me close it up. I'm sorry don't want to close up quite yet, and here's the closing part. and I did it before that. So I bundled them all together and I said after right years is complete, here's an example and it's the year 2012-13 again for consistency on lines 205 and 206 So here it is 206 check it out. I have an object now that I've defined as year 12/13, that has one year 2012-13 string and then I have an ArrayList of string values for each week. So now I've put them together bundled them, brilliant.

But now I need to bundle those objects that are separate into an array of objects. So I wrote this function, right market data. So this to me is the last step. I just need now to, in this function, create a variable that's typed as an array and another one as an array. So I can push all of those object arrays as a list, one nice master list. And then what I do just for good measure is I take that full-year data array of objects. and I assign it to another array. And then I write out my console logs out here.

So now I have seven years and each year has together with it, the 17 weeks of postings. So to me that's what I needed to be able to create these individual lines that tracked cross per week, so that's how I did it. After I got that year data variable there, I sent it on a merry way to the paint data biz function that's defined down here. And you can look more into the different templates for multi-lines in D3. But maybe just for good measure if we open it up. I've collapsed everything and the most important well there's multiple spaces that I use this but, I take that data that's formatted in that way and this is where I use it specifically to start making those different lines.Again I'm not going to review here but, this is where the magic does happen.

So that's a job for you to start going on your own D3 visualization journey to start figuring out, okay now that I know how to read data I know how to read code a little bit there's a lot to learn still right, that I haven't covered and I'm not able to cover in this tutorial series. But I hope at least you feel a sense that you can understand that what are all these curly braces in square brackets dots variables as diamonds what are all these things doing? And I hope that helps you be able to parse and understand a lot of the tutorials that are already out there. Because to me that's often the missing set of knowledges, is the more stable stuff when it comes to like data structures are pretty much one of the more durable artifacts and digital environments. So if you start figuring out those durable texts, if you will, you can start figuring out how to write them programmatically with programming languages.

I hope this has been useful. Let me know and also Ridolfo know if you have any questions about anything we'd be happy to review those.

Alright good luck out there.