Video Title: “Video 3: The basics of reading JavaScript”
From “Rhetmap.org: Composing Data for Future Re-Use and Visualization”
by Chris Lindgren and Jim Ridolfo
(Kairos XX.X PraxisWiki)

Welcome back to the third video for the Missing tutorial. Reading JavaScript. I want to make sure that you have a basic understanding about how to read JavaScript and of course I can't teach you all there is to know about how to read JavaScript and how to write it. Mainly this video's about becoming acquainted therefore (talks).

So I have a little addendum here. Functional style. In my code I write what's called a functional style so we will be learning how functions work and they have scope and we'll also learn about those basic data types and how they operate in JavaScript.

Let's get started.

Here's what we need to get started. Open up your browser and on the left side you can see I'm just here at DuckDuckGo, just a simple site and what I've done is I've opened up what's called the element inspector tool that's in most major browsers. I'm using Chrome. Firefox will be very similar I believe even Opera, Safari have this as well but if you want to follow along so that it looks very similar, I recommend using the Chrome browser here. I also have the Rhet Map code over here which we're gonna get to but again we're not gonna focus our energy quite here yet but I'm gonna have it here for a second for reference later in the video. What we need right now though is this window and this element Inspector tool open.

So I just closed it right now and I'll show you how to do that. You just right click and go to not view page source but this inspect, select that and yours might actually pop up on this right (talks) side but it also might have popped up down here. It doesn't matter, there are ways to switch where you want it to be but I won't cover that here. It also will have other windows open and more visible. So I have the console open up right now. If that's not open for you, you can click on this button and it should pop open. There might even be messages in there if you've never used it before that you can possibly close. You can close those.

But here's what this an element inspector is. This is HTML for the page. This is all the CSS code for that page. We're not gonna be concerned with that 'cause we're actually doing a little thing about JavaScript. So we can write JavaScript in this console and that's what I'm gonna be doing is just showing you some basics.

Okay. Let's talk about some data types.

There are a few that you should be aware of you're reading JavaScript and you can store data in what are called variables. So you might have seen code in the past. If you have some experience with JavaScript you can type out a cue, a phrase V-A-R. That's a keyword that the JavaScript programming languages recognizes as, "Oh you're trying to what's called instantiate a variable,"

meaning I'm gonna store this in memory, I'm gonna store something and you can give it a name of some kind. So let's talk about strings. I'm gonna make a string. A string is essentially some kind of literal form of text of some sort whether it's in numbers or alphabetic characters or any other kind of character in any written language. So I can say assign to string variable. That's what the equal signs mean. That's what that means. To make a string, you put something inside of quotation marks. Okay, let's just say string literal. Maybe I'll just put in some numbers. One, two, three and some commas there all of that will be... This is a literal string right here that I can do lots of things to which we want to (talks) here in the video but if I want to close this statement, this variable statement I'm going to do a semicolon. And now when I press enter and yes it did say undefined but check it out, if I write a variable name string and you can see already it's got a ghostly image of what I defined before. If I press enter, it prints it out for me in the console.

I can also do what's called a console log. You know I'm in the console and type out string inside of that log, semicolon to close it. It printed out the string and notice how in this case, before when I typed out string, enter, it has quotations marks surrounds it that suggests, "Oh this is a string," in the console log when I typed in the variable name string. It actually just prints out the string literal itself. So those are strings.

What are some other variable and data types? Okay let's make some integers. Integers okay. I'm gonna assign to this integers keyword, just a number. So without quotations so let's just do one and now if I type in integers, oops, then I get the number one. If I'm more on the (talks), I can do the following. One assign to one, the actual number one. And then let's do var two, assign the number two.

Those are integers. So now what if happens when I type out two, I get two, one, I get one in the console. So now I can actually do something where I can add them because it's a programming language. So I can type out a statement like this and express expression one plus two using those variable names and it'll assume that I'm doing a statement of adding them together since they are integers. If I try to do that with strings it won't work.

So let's do this, one string equals one, two string is I'm sorry assigned, we're gonna assign to that. We should use consistent language. Okay now what happens if I try to add those together. Let's see here one string, two string. What's gonna happen? JavaScript did something funny there. It makes it look like (talks) the way I typed it. That it somehow got some funky math but that's not what happened, right? It actually combined those two strings. It's called concatenation. So that's another fancy word.

So when you use the plus sign in different contexts whether you're using strings versus let's say numbers like integers, it's gonna do different things. So even though it looked like it did funky math, one plus two is twelve (talks), no it just combined the string one and the string two. Okay so those are strings, integers and then also some computational work that can help you maybe see what's happening inside JavaScript. Those other types of numbers, you can add decimals, they're called floats. We won't work with those here but there is a difference.

Let's talk about those arrays again. Number arrays, they're lists. So let's make an array here. I'm gonna type... Actually I shouldn't... I mean I've been typing Python lately so let's not do that underscore, I've been doing that. In JavaScript it might look more something like this. Array example. So assign this. I can instantiate an array, arrayexample and right now it's gonna be empty. So now if I type an arrayexample, it shows me. Okay you've typed and empty array here. See how it says even there's a method length attached to it and there's all this other stuff.

Do you notice how we get all this? It seems like what? Why would I even need to know all this? These are all methods that the programming language of JavaScript has in relationship to this data type (talks). So I can use all of this to... Essentially I could use for each to traverse the entire array to search for something or do something to it for each item in there. I can also push new items to it.

So actually that's what I wanna do. 'cause I don't like that it's length zero. So let's do this. Let's say arrayexample.length. Let's do that first. Length is we get zero. Now let's add something to it. Remember that push. Push means append, add something. So let's add an item to it. Let's add a string and see if we can add something else. Let's add the number one, the integer one. Okay why did it print out two you might be wondering? That's the length. In the console here, it just assume, just does that. So if I type in this, it says we've got two items in there. If I wanted to find out. Let's see here, just what's inside of it, print it out. I just typed in the actual name and check it out.

Now we have two items. We have a string, (talks) string and then separate it with a comma, another item in that array list is the integer one. So that's how you create an array and inside of a list you can create lots of things. Usually it's more meaningful like an array list suggests, it's some kind of set of values that makes sense to put together but this is just me showing you that lots of things can go inside of it.

Finally I wanna show you one last data type that should be familiar because of our past, (talks) number two about data, are the object arrays which are also sometimes called dictionaries because they have keys and values. So instead of the square brackets, we're going to, let's see. Object array assign to it. Let's just make an empty one. Let's call instantiate. So I've created this variable, I'm sorry I didn't do that, did I? Our keyword there.

There. Now I have an empty object and I'm gonna add something to it now. Similarly to the array list, you can look at the object itself and a lot of the properties, including things like you might notice here the length and the name of it. Different things, you can peruse, try to understand. It gives you actually a lot of great keywords to type into the search engines if you wanna learn more about something. If you don't have the language and you're trying to cultivate it and you're reading a tutorial or a book about JavaScript et cetera et cetera.

So it's intimidating right now so we're gonna close it up and I'm just gonna show you how to add something to an object array. So in this case easily. You can do it more programmatically

when you learn how to write different kinds of functions and loops, conditionals et cetera but for now, what we can do to add to this object array is type it out.

So now I've called it up, it knows what, it recognizes it but I can use dot notation like this to actually construct some kind of object inside of this object array. So let's see. Let's do just the keyword example. So remember keys. So I've given it a key with the name example. I'm gonna assign to that key a name of some kind. So example value. There's a string there. I'm gonna close that expression, that statement and I'm gonna press enter and it gave me the output.

So now what happens if I type object array. Check it out. We have now inside of it a key value pair. Key example, value a string literal example that says example value. We can add another one. See here let's add object array dot keyword and number or maybe how about this int for integer equals, let's give it a value now. This key has a value, let's see. How about actually an array of integers. So one, two, three, four.

So let's print that out. Check it out. We have now two objects in this object array. We have an example key with a string and now an int key with a list, an array of integers. And I could do things to those now in JavaScript. I can iterate over them, I can add more things, take things out, rearrange them, reassign new values to those keys, lots of things. That's what programming languages do. They enable you to revise if you will and reorganize, categorize, write and read at scale.

So this has just been the basics of different data types and just some quick and simple ways to add things to them so that when we get to the JavaScript code on the right, we will not be totally confused. You can follow along sequence by sequence.

The last thing I wanna show you is how to read functions and tell you about what's called scope 'cause that's gonna be really important when we get over to this side here. So let's set it up.

Why is that important? You see these different values in the code editor right here? We have get data and inside of that or next to it we have print parentheses. What that says is call this function. You see these three different functions. My script for the data viz, if we go over here for this data viz is actually three different functions. I get the data, I format it and then I call it, just call it paintDataViz. So that should tell you that, "Well that's where we're gonna have some data viz work with the D3 library" but before I even got to the D3 library and how to paint it to the browser, I had to format it.

So in the next part here, I'm gonna show you how to read functions. What is scope? Meaning if I open this up ,you can see how the line numbers move from 91 to 223. That means there's stuff inside of there. And there's a lot that we're gonna walk through and we see all these curly braces. Those are defining scope.

So different parts of this broader function called formatData. I can actually simplify here real quick to show you that we actually start the scope here on 91 and this function ends down here

on line 222 where we even see a semicolon saying this function's closed. This entire function is closed.

So how do we read that? Let's take a look.

Okay before we learned about different data types but in the context of the JavaScript programming language, we learned about string literals, we learned about integers, we learned about arrays and object arrays within the context of writing them out and assigning them to a variable. So we give it a variable, a keyword name and then we use that equal sign to assign a particular data type to that keyword variable in memory.

In this part, I'm going to review functions because functions are a big part of JavaScript. It's not everything there is to know about JavaScript by any means but I wrote my JavaScript code in a functional style if you will. Some might argue not the greatest functional style but again the context of this is to teach people how to read this stuff. So I've written my code in a certain way so that's my little disclaimer there.

But let's learn about how functions work, how it relies on this idea of scope and you have to call it, give it what are called arguments and then pass parameters into that function to then use and do something too. So it's a way for you to actually create and manipulate, maybe that's not the best word but remember how I said in programming languages you can write and revise, you can reorganize. Functions are the big idea maybe even little ideas that functions are little workers if you will that do one thing well. You can nest functions within functions in JavaScript but you don't obviously wanna nest too many things in one thing.So the idea is that if you're gonna do an action repeatedly, you might as well create a function that can do that for you.

So let's begin.

This is a great resource by Wes Bos. I think that's how you pronounce his name. I apologize if not but if you wanna learn more about JavaScript and web development, he's actually a really great person to follow and has done a lot of things with Mozilla Foundation to create a lot of tutorials for people out there. And I thought this was great because it shows the scope and all the different pieces of a function, of a simple function.

So even though it's not connected to my code, I thought this is like really great resource to review and a chance for me to even share a person to follow. What's going on here? What we see going on in this case is that Wes has defined a function. Here at the top we have a function definition and we can see how it has encompasses this whole idea here where we have the keyword function. We get that function and name so that way if you don't give it a name, it's called an anonymous function which can be called but you definitely want it to document your code if you will by giving your function a name so that people know what it's supposed to do. You know what it's gonna do. In this case it's gonna calculate a bill.

So let's take a look at the parameters. Parameters are things you pass and use and you know you're gonna be using inside the function to essentially input data to then be able to output it. So let's see what's going on here. We have a parameter named meal and then a tax rate with an assigned value of looks like 5%. So we have two parameters that are passed into this function. We assume they're gonna be used in some way. And note we'll talk about that in a second. Note here we have next curly braces that suggests this is where this function starts and here at bottom, this is where it's gonna end. Inside of it we have what's called the body of the function where this is where again you can kind of see some relationship between writing and programming already where inside of this is where we're gonna be doing something.

So if you're wondering what this means, it's constant. So some kind of constant that you would return. I'm (talks) getting the details but we have... It's essentially a variable, a special kind of variable that we assign something to so we have the name total and look now we're first using meal, we're gonna talk about where that comes from in a second. And this means times and this gives scope to this right here. JavaScript is gonna read this and compute it and then multiply it to whatever the value of meal is. So one plus the tax rate and (talks) notice how again. What's that? The semicolon. It says that's when the expression statement ends. Whenever you have a function, if you're going to be calling it somewhere else, normally you're going to return that value so that you write a return statement. So in this case you only have one simple thing of total.

So you input some value parameters, you compute it add it to variable total and then return that value before the end to wherever you call that, this function calculateBill. So that's the part (talks) is missing right? What's going on? Oh look down here. We have a variable that we create and we instantiate that we call myTotal in this case and we're gonna assign it the returned value of calculateBill and now I'm hoping you see why it's important to name your functions for the most part. There are maybe some specific situations where it's okay not to but then note here we have arguments. This is where you can actually have dynamic content passed to or function based on recurrent situations.

We're being rhetorical here, right? So in this case, we can see that the order of these values matter. In this case, we have $100 for the meal and then in this case they passed a specific tax rate that would override that default value that's up here. So instead of it being .05, it would be .13. So these values can come from a lot of different places from other variables that are stored somewhere based on input but in this case they're hardcoded if you will mean just for an example, let's call calculateBill, give it the meal value of a hundred and the tax rate of 13% in this case. This calculateBill when it's called will go to this function, pass as parameters these arguments and then do something to them right here in the body then it'll return it back to this area and assign that value that's returned to myTotal.

And boom you have input and output and that's a function that has scope and you can use it in lots of different ways beyond this way and it'd be written in many different ways that you see, then you see here.

The one thing I'll note that I maybe didn't mention is notice how it matters, I didn't note how the order matters like what you say in what order with regards to calling it but it also matters for documentation the order of use inside the function body, I didn't mention that second part. So notice how the meal is used first then you gotta write one here and then two, oh that's ugly sorry then two tax rate. So that's another little tiny thing that you don't find quite easily until you start writing it and then things get thrown, errors happen and people complain about your code. That's happened to me in the past. So I learned the hard way so hopefully this helps you read functions.

Okay now let's take a quick look. Here's the functions I got cooking in the data processing code before we actually look at it. We'll look at more closely. I have three, one, two, three and so I have a function to get the Google sheet data that we looked at then I'm processing the data and format data and then the code we're not gonna look at but here's where the data viz work happens here in the paintDataViz. So I've separated my concerns. I've said first get the data, make sure that works. Once that works, send it to this, that data to the formatData function.

So this very first note here of getData, that data call, notice how getData doesn't have any arguments because I'm just trying to call the function to get the code working inside of this index file itself. And that's why that's the case there. If you look at inside the code of formatData, we see a few things happening. So it just means to get you acquainted before we look at it a little bit more closely (talks) is notice how many variables I name here, right? (talks) the top of formatData function. I can use those variables inside of formatData anywhere. And so it matters when I assign these variables. It matters then I can't use any of these outside of formatData. So if I tried to call and use let's say any of these arrays somewhere else outside of formatData, it'd be like the JavaScript would throw an error and say that's not instantiated. It's because scope protects data types and things stored in memory and that matters as well. It even gets into security. I'm not gonna say my code is the most secure either but it's definitely something to learn about along the way.

One other thing to mention before we go is how do I go from get data to format data? Well here it is right here. This is just one part of get data but there's a request that I send to Google essentially in this case. And if the request that I send to get the data is successful and I get this little message here that's coded as 200 success. Now I format the data as a JSON object here. So here's where I get the response request data. I put it in a text format and I pass it as JSON, remember JSON objects and I store it under the variable with keyword data. I'll review this one more time when we get to it but I wanna note here look what happens then is I pass that as an argument to formatData. So this is just a little warm up until we get to the code now.

So let's go. Let's start reading the data processing code.