Welcome to Scribbles on Scripts!

So. Function arguments. Despite the name, they're generally pretty laid back and straightforward, but there's some weirdness that can happen and it can all be because of the datatype you are using. Let me explain.

You know when you pass in a string to a function, hoping to modify a bit of text, maybe convert it to lowercase with a handy string method, add an s to the end to pluralize, but somehow none of your changes are sticking? You've checked your syntax and your spelling, added in a bunch of console.logs, and everything's looking fine, but you're still not getting the result you were hoping for. The good news is that you're doing everything right and your code's not broken.

What's actually happening is that JavaScript made a copy of your string, and gave this copy to the function. This clone lives it's entire life inside of the function you passed it to, and any changes you make to it are not reflected in your original string that stays outside. Strings are not the only datatypes that work this way, many of the other ones you are familiar with do this too, including numbers and booleans.

So what's the solution? It depends. You could always return a new string from your function, or define a variable just outside your function and not pass it in... but this changes what the function fundamentally does. It no longer updates the exact bit of data you passed it. If we want to preserve this functionality, which is a very valid thing to want, we can wrap our data in an object. Let's see what this would like.

This works because objects are not exactly what they seem. An object inside a variable doesn't actually contain the data in the object, instead, it holds an address. This address leads to the location in memory where the object lives. So JavaScript is still making a copy of the argument getting passed in to the function, but a copy of an address will still lead you to the same location.

If you found this video helpful, please give it a like, and subscribe for more just like it. If there's a topic you'd like me to cover in a future video, let me know in the comments down below! Thanks for watching!