How do you copy by value a composite data type?

Since the default behaviour for composite data type is to copy the reference of one variable to another, we need other options to copy the values from one to another. This way we do not depend on single source for the values. So, if something happens to the original Object, Array etc we can still have the copy of the value in another place.

Here are some ways you can copy by value a composite data type to new variable: -

* Text

  Description automatically generatedUsing for loop - We can simply run through the Array or object and copy each of the values into another array. In the example below as you can see source and target are not the same object but contains same values. Values copied from source into target.
* Text

  Description automatically generatedUsing Spread Operator – Spread operator is the new feature in ES6. While spread operator is generally used to add new properties to an existing object, it can be used create completely new arrays with the same values of another array. In the example below, we can that source and target are different but consists of the same value.
* Text

  Description automatically generatedDeep Copying (JSON.parse(JSON.stringify)) - There are many ways in JavaScript to do this, but this creates a problem when we have inner values which are also of the composite data type. When we simply use the above methods, we only copy the top-level values so, any inner objects or arrays will still behave like copy by reference. This concept is known as shallow copying. To copy everything, we need use deep copying. To deep Copy we first convert the whole object into string and then parse it again as json. In the example below, we see that source and target are different and when the source’s inner array value is changed it does not affect the target array thus successfully deep copying. We can also achieve this manually by using recursion.

Why is there a difference in behaviour for copying contents in primitive and non-primitive type?

Diagram

Description automatically generatedThe Non-primitive (composite) data type simply means a datatype can hold multiple values. Examples include Arrays, Objects, functions etc. Composite data types contain multiple values. Sometimes the variable of a certain composite data type might contain huge amounts of data copying (by value) for each value inside the variable would by a costly affair in case of time and space complexities. Thus, copying of a non-primitive data is to another is done by copying the reference of the other non-primitive value. Here’s a picture showing Pass by reference.

Now let’s take an example.

Imagine getting data from an API which returns millions of users for a Social media account. We store that to another array by reference. So the array is generated only once, we simply use the address of the array to act on it. If we copied by value, we would have to run a loop a million times which not only be very slow but copying every value again might crash the application as we run out of ram.

A screenshot of a cell phone

Description automatically generatedUse typeof in all the datatypes and check the result.

Write a blog about objects and its internal representation in JavaScript.

Objects conceptually represent real world items that we want to represent logically in our programs. All objects have properties and behaviour that define the real-world item. Behaviour is referred as functions and properties are variables inside the object.

Text

Description automatically generatedIn Java Script Objects are represented as key value pairs. Everything in JavaScript is an Object except for primitives. Java Arrays are also stored as Objects.

In the above example we have an empty object but when converted to String, it shows that it has an object inside. This object is known as prototype of the object. This links it to the Parent Object which all objects in JavaScript inherit.

A picture containing text

Description automatically generated

Above is shown the internals when an object is created or how JavaScript handles object. funnyguy is a user-defined empty object, which by default consists of the prototype object. All functionalities such as toString(), valueOf, are present in this prototype object.

What is the difference between Window, Screen, Document in JavaScript?

# **Window**

Window is an Object that is on the top of the JavaScript Object Hierarchy. All global javascript objects, functions, Variables, etc automatically become members of the Window object. The window represents the current browsing context. It contains things such as window.location, window.history, window.screen etc. Every browser supports the Window Object. While the browser is open each tab has its own top-level window object.

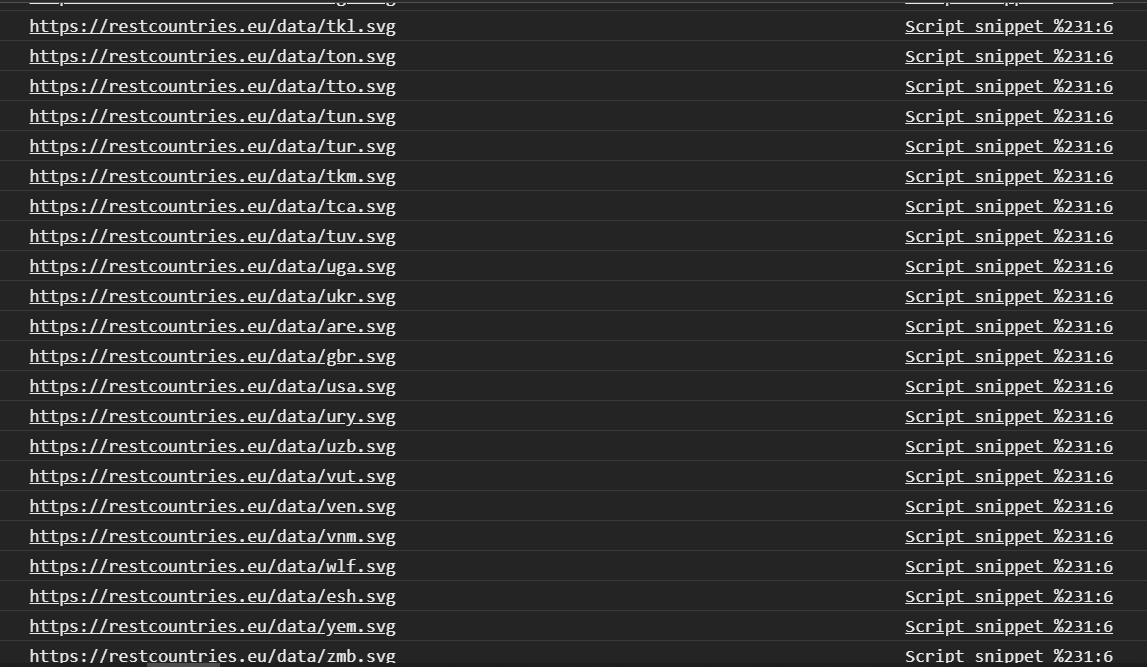
# **Document**

Document represents the web page loaded in the browser and serves as an entry point into the web page’s content which is the DOM tree. When an HTML is loaded onto a browser it becomes the Document Object. This object is loaded in the Window object. This document Object has properties such as title, URL, cookie etc.

# **Screen**

Screen represents the information about the dimensions of the physical screen. It consists of data such as screen height, screen width, colour depth, pixel depth etc. Screen also resides in the window object.

Extract and print the flag URL of all the countries in console.

Text

Description automatically generated