1. **How do you copy by value a composite data type?**

By default, all the primitive data types follow copy by value method of copying data, whereas composite data types follow copy by reference method.

But there are ways to copy data from composite data types using copy by value method.they are :

* By Object.assign() method;
* By Spread operator;
* By JSON.parse() & JSON.stringify();

**→ By Object.assign() method :**

**let array1 = ["x", "y", "z"];**

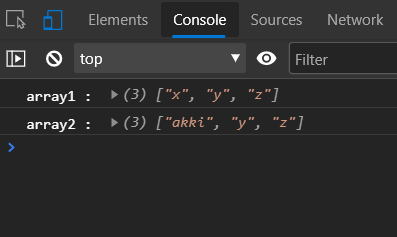
**let array2 = Object.assign([], array1);**

**array2[0] = "akki";**

**console.log("array1 : " , array1);**

**console.log("array2 : " , array2);**

**Console:**

****

**→ By Spread Operator(...) :-**

**let array1 = ["x", "y", "z"];**

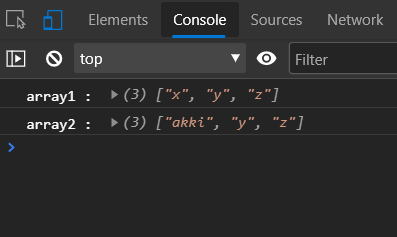
**let array2 = [...array1];**

**array2[0] = "akki";**

**console.log("array1 : " , array1);**

**console.log("array2 : " , array2);**

**Console :**

****

**→ By JSON.parse() & JSON.stringify :**

**let array1 = ["x", "y", "z"];**

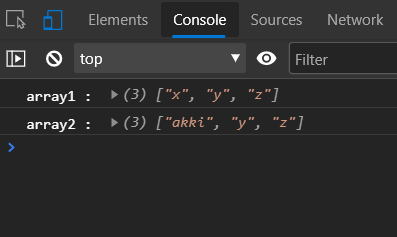
**let array2 = JSON.parse(JSON.stringify(array1));**

**array2[0] = "akki";**

**console.log("array1 : " , array1);**

**console.log("array2 : " , array2);**

**Console:**

****

**2 .What is the difference in behavior for copying contents in primitive and non primitive type?**

* **Copying contents of primitive data type:**

Primitive data types follow the “copy by value” method of copying contents to another variable. In copy by value method, a new memory location is created and only data is copied to a new variable. Even if we change the contents of the new variable, the old variable’s contents remain the same.

Example :-

let string1 = "akki";

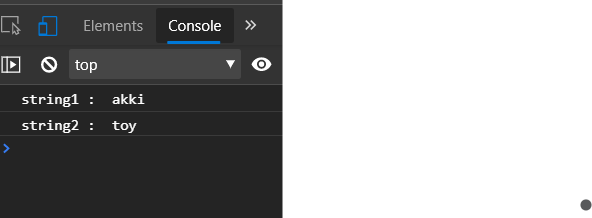
let string2 = string1;

string2 = "toy";

console.log("string1 : ",string1);

console.log("string2 : ", string2);

Console:



* **Copying contents of non-primitive type :-**

Non-primitive data types or Composite data types follow the “Copy by reference” method of copying contents to a new variable. In copy by reference method, no new memory location is created for new variables. Instead the new variable is pointed to the memory location of the old variable. So when the contents of new variable is changed, the contents of old variable also gets changed.

Example :-

let array1 = [1,2,3,4,5];

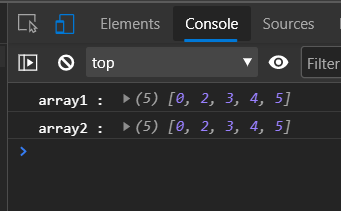
let array2 = array1;

array2[0] = 0;

console.log("array1 : ", array1);

console.log("array2 : ", array2);

Console :-



**3. Use typeof in all the datatypes and check the result**

typeof(1) - **number;**

typeof(1.1) - **number**;

typeof("1.1") - **string**;

typeof(true) - **boolean**;

typeof(null) - **object**;

typeof(undefined) - **undefined**;

typeof([]) - **object**;

typeof({}) - **object**;

**4 . Write a blog about objects and its internal representation in Javascript.**

JavaScript objects are containers for named values called properties or methods.

The values are written as key value pairs such as {key:value} separated by a semicolon.

Objects can contain any combination of primitive(string,number,boolean) and composite data types(array,function,object).

Example of object in javascript:

let student = {

name : "akki",

grade : 5,

marks : [90, 88, 95],

attendance : 35}

**Creation of Objects :**

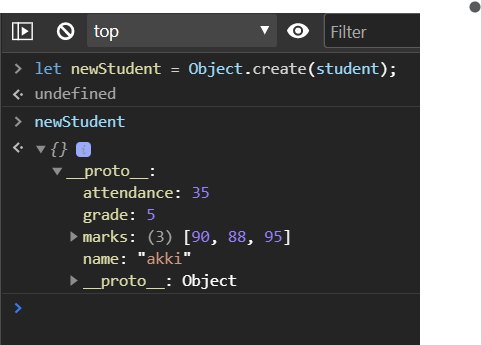
* **Object literal :-**

Objects can be created directly by Object literal syntax using bracket notation {} as done in below example.

Ex :- Let obj = { key1 : “value1” , key2 : “value2” };

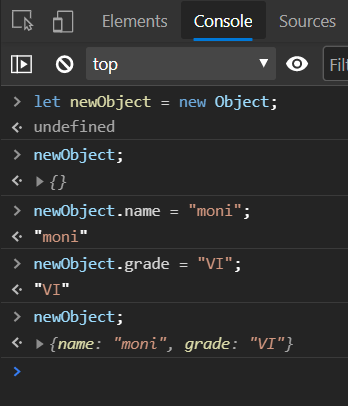
* **Object.create() :-**

object.create() method creates new object with properties and prototype of specified old object.



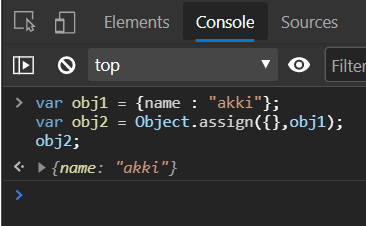
* **Object Instance method :-**

By using new keyword, a new empty object can be created. Required key value pairs pairs can later be assigned to the object.



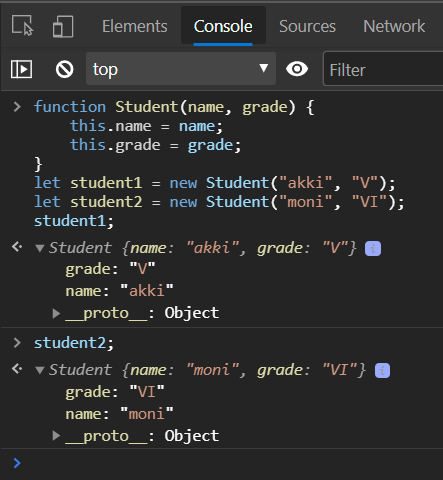
* **Object.assign() :-**

By using Object.assign() method, we can assign properties of one object to another object.

****

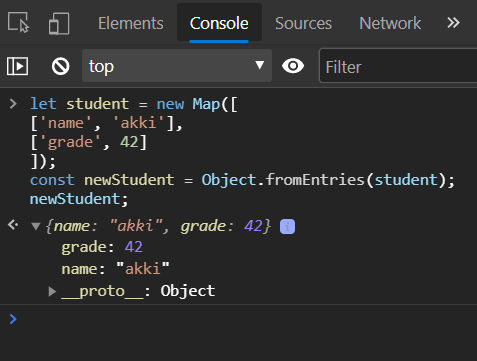
* **Object Constructor :-**

A constructor is a function that creates an instance of a class which is typically called an “object”. Object Constructors can be used to create multiple objects without having to redefine objects each time.



* **Object.fromEntries() :-**

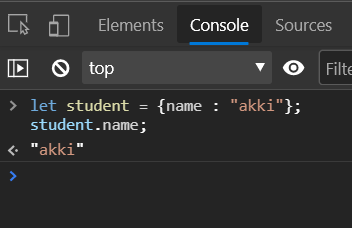
Object.fromEntries() method transforms a list of key value pairs into an object.

****

**Accessing Objects :-**

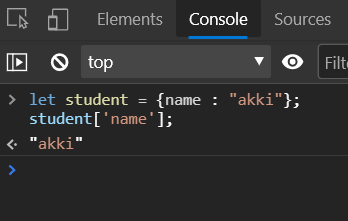
Objects can be accessed by two ways - Dot notation & Square Brackets.

**→ Dot notation :-**

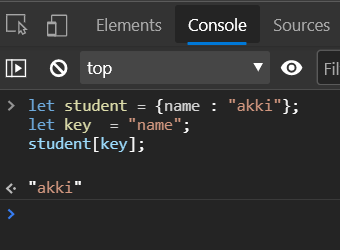


**→ Square brackets :-**

* By directly using keys of objects.



* If the keys are stored in a variable name, square brackets can be used without any quotations.



**5. Execute and see at least 15 cli commands. like mkdir, ls etc.**

|  |  |
| --- | --- |
| **Command** | **Description** |
| dir | List the directory (folder) system. |
| cd pathname | Change directory (folder) in the file system. |
| cd \ | Move to the root folder of the file system. |
| cd .. | Move one level up (one folder) in the file system. |
| copy | Copy a file to another folder. |
| move | Move a file to another folder. |
| type filename | Type a file. |
| mkdir or md | Creates a new directory (folder). |
| rmdir or rd | Removes a directory (folder). |
| cls | Clears the CLI window. |
| exit | Closes the CLI window. |
| help command | Shows the manual for a given command. |

**6. What is the difference between window, document and screen in Javascript**

## **Window :-**

The JavaScript window object is the top of JavaScript Object hierarchy and represents the browser window. The window object is supported by all browsers.

All global JavaScript objects , functions, and variables automatically become members of the window object. The window is the first thing that gets loaded into the browser . This window object has the majority of the properties like length, innerWidth, innerHeight, name, if it has been closed, its parents, and more.

Example :- window.innerWidth; // 150

The window object represents the current browsing context . It holds things like window.location, window.history, window.screen, window.status, or the window.document . Each browser tab has its own top-level window object. Each of these windows gets its own separate global object. window.window always refers to window, but window.parent and window.top might refer to enclosing windows, giving access to other execution contexts. Moreover, the window property of a window object points to the window object itself.

Example :- window.window;

Window {parent: Window, opener: null, top: Window, frames: Window, …}

**Document :-**

The Document interface represents any 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 document is loaded into a web browser , it becomes a document object. It is the root node of the HTML document. The document actually gets loaded inside the window object and has properties available to it like title, URL, cookie, etc.

HTML documents, served with the "text/html" content type, also implement the HTMLDocument interface, whereas XML and SVG documents implement the XMLDocument interface.

Example :-

document.getElementById(“header”); // gives the value of header

As document is property of window the above code is same as

window.document.getElementById(“header”); // gives the value of header.

**Screen :-**

Screen is a small information object about physical screen dimensions . It can be used to display screen width, height, colorDepth, pixelDepth etc. It is not mandatory to write window prefix with screen object. It can be written without a window prefix.

Properties:

screen.width

screen.height

screen.availWidth

screen.availHeight

screen.colorDepth

screen.pixelDepth

**7. Extract and print the flag url of all the countries in the console. Use the html template.**

<https://restcountries.eu/rest/v2/all>

//create a request variable

var request = new XMLHttpRequest();

//open a connection

// three parameters- "request method","url",true

request.open("GET" , "https://restcountries.eu/rest/v2/all",true);

//send the request

request.send();

//load the response

request.onload = function(){

var data = JSON.parse(this.response);

//console.log(data);

var flag = [];

data.forEach(element => {

flag.push(element.flag);

});

console.log(flag);

}