JavaScript: Task 1

HTTP/1.1

Programs like gzip have long been used to compress the data sent in HTTP messages, especially to decrease the size of CSS and JavaScript files. The header component of a message, however, is always sent as plain text.

HTTP/2

One of the themes that has come up again and again in HTTP/2 is its ability to use the binary framing layer to exhibit greater control over finer detail. The same is true when it comes to header compression. HTTP/2 can split headers from their data, resulting in a header frame and a data frame.

HTTP History

The term hypertext was coined by Ted Nelson in 1965 in the Xanadu Project, which was in turn inspired by Vannevar Bush's 1930s vision of the microfilm-based information retrieval and management "memex" system described in his 1945 essay "As We May Think". Tim Berners-Lee and his team at CERN are credited with inventing the original HTTP, along with HTML and the associated technology for a web server and a text-based web browser.

Year	**HTTP Version**
1996	1.0
1997	1.1
2015	2.0
Draft (2020)	3.0

Difference between Node.JS and Javascript

Javascript is a programming language that is used for writing scripts on the website.	NodeJS is a Javascript runtime environment.
Javascript can only be run in the browsers.	NodeJS code can be run outside the browser.
It is basically used on the client-side.	It is mostly used on the server-side.
Javascript is capable enough to add HTML and play with the DOM.	Nodejs does not have capability to add HTML tags.
Javascript can run in any browser engine as like JS core in safari and Spidermonkey in Firefox.	Nodejs can only run in V8 engine of google chrome.

What happens when you type a URL in the address bar in the browser?

1. You enter a URL into a web browser
2. The browser looks up the IP address for the domain name via DNS
3. The browser sends a HTTP *request* to the server
4. The server sends back a HTTP *response*
5. The browser begins rendering the HTML
6. The browser sends requests for additional objects embedded in HTML (images, css, JavaScript) and repeats steps 3-5.
7. Once the page is loaded, the browser sends further async requests as needed.

JavaScript: Task 2

1.HTML and Script.JS file and run a for loop on the data and print all the country names in console.

script.js

```
var request= new XMLHttpRequest();//request variable
request.open('GET','https://restcountries.eu/rest/v2/all',true)
request.send();//send the request
request.onload =function (){  //anonymous function
var data= JSON.parse(this.response); //process and load the response
var sum=0;
for(let i=0;i<data.length;i++)
{
    console.log(data[i].name); //Fetching the name of every country

}
}
```

Index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>sample api data</title>
</head>
<body>
   <script src="script.js"></script>
</body>
</html>
```

2.Try the rest countries api. Extract and print the total population of all the countries in the console. use the html template.

script.js

```javascript
var request= new XMLHttpRequest();//request variable
request.open('GET','https://restcountries.eu/rest/v2/all',true)
request.send();
request.onload =function (){
var data= JSON.parse(this.response); var sum=0;
for(let i=0;i<data.length;i++)
{
        sum= sum+data[i].population;
}
console.log("Total Population"+" "+ sum);
}
```

Index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>sample api data</title>
</head>
<body>
   <script src="script.js"></script>
</body>
</html>
```

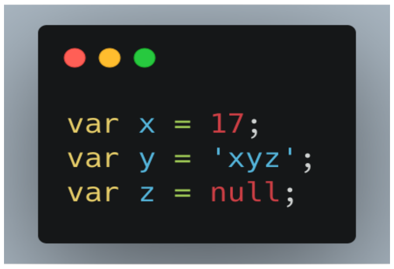
Output:

Total Population 7349137231

3.Write a write up on Difference between copy by value and copy by reference?

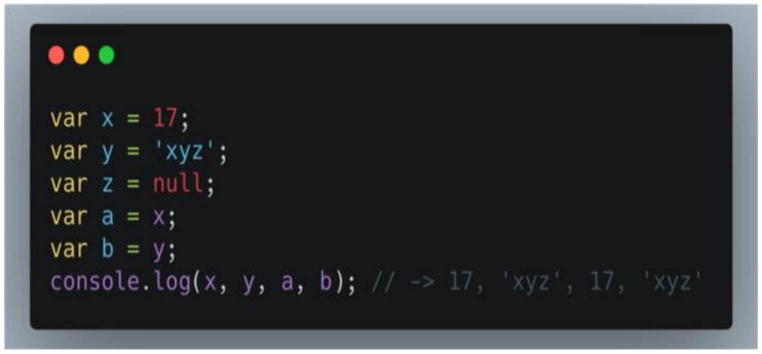
Copy by value

In a primitive data-type when a variable is assigned a value we can imagine that a box is created in the memory. This box has a sticker attached to it i.e. the variable name. Inside the box the value assigned to the variable is stored.



```
var x = 17;
var y = 'xyz';
var z = null;
```

In Figure 1, ‘x’ contains value 17, ‘y’ contains ‘xyz’.



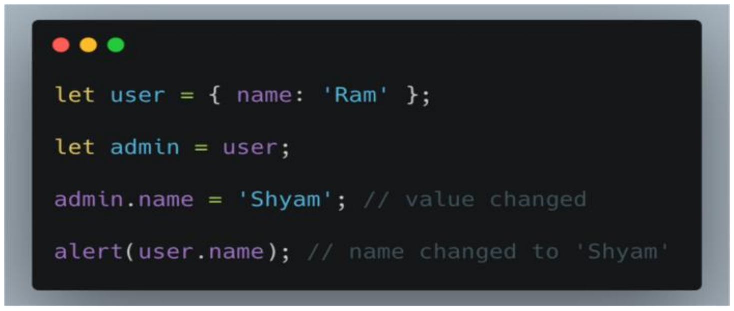
```
var x = 17;
var y = 'xyz';
var z = null;
var a = x;
var b = y;
console.log(x, y, a, b); // -> 17, 'xyz', 17, 'xyz'
```

In the Figure 2; the **values** in the boxes ‘x’ and ‘y’ are copied into the variables ‘a’ and ‘b’.

At this point of time both ‘x’ and ‘a’ contains the value 17. Both ‘y’ and ‘b’ contains the value ‘xyz’. However, an important thing to understand here is that even though ‘x’ and ‘a’ as well as ‘y’ and ‘b’ contains the same value they are not connected to each other. It is so because the values are directly copied into the new variables.Changes taking palace in one does not affect the other.

Copy by reference

In case of a non-primitive data-type the values are not directly copied. When a non-primitive data-type is assigned a value a box is created with a sticker of the name of the data-type. However, the values it is assigned is not stored directly in the box. The language itself assigns a different memory location to store the data. The address of this memory location is stored in the box created.



```
let user = { name: 'Ram' };

let admin = user;

admin.name = 'Shyam'; // value changed

alert(user.name); // name changed to 'Shyam'
```

In the Figure 3, when the value of admin is changed it automatically changes the value of user as well.

This happens because both ‘user’ and ‘admin’ are storing the address of the memory location. And when one changes the values in the allocated memory it is reflected in the other as well.

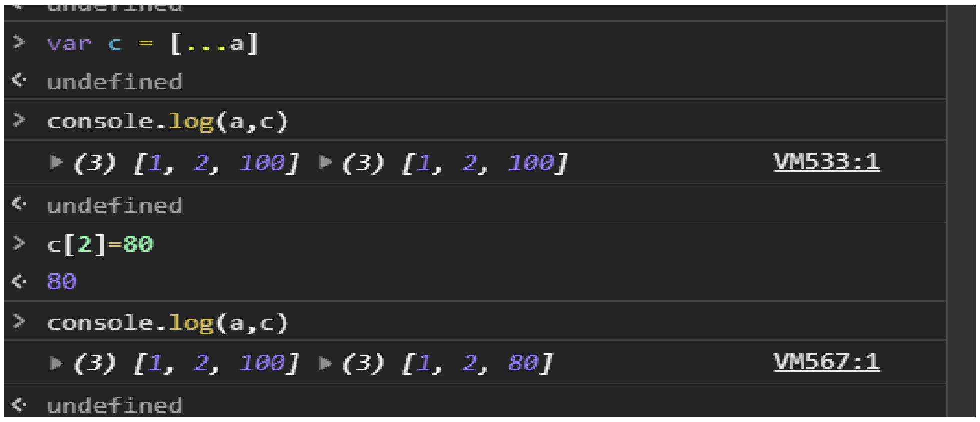
We can further elaborate it we can say that; copy by reference is like having two keys of the same room shared between ‘admin’ and ‘user’. If one of them alters the arrangement of the room the other would experience it ads well.

4.How to copy by value a composite data type(array+object)?

There are 3 ways to copy by value for composite data types.

1. Using Spread (...)

Spread operator allows an iterable to expand in places where 0+ arguments are expected. It is mostly used in the variable array where there is more than 1 values are expected. It allows us the privilege to obtain a list of parameters from an array.Using spread will clone your object. Note this will be a shallow copy.

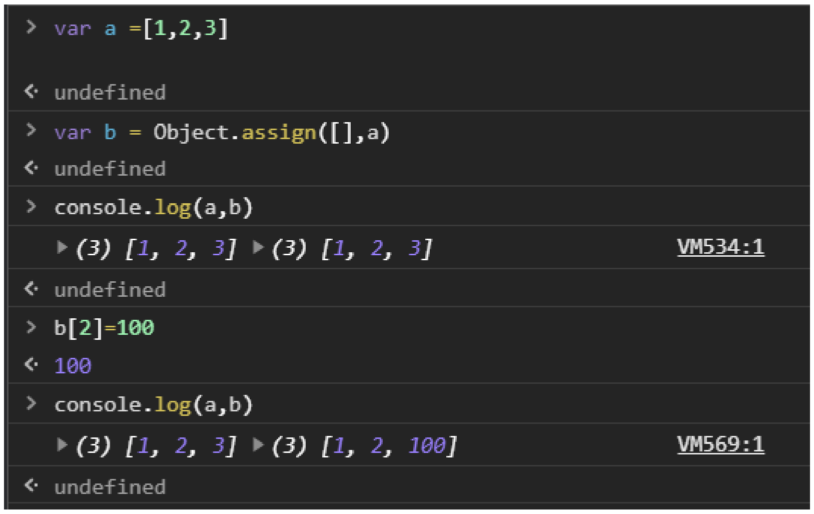


```
> var c = [...a]
<· undefined
> console.log(a,c)
   ▶(3) [1, 2, 100] ▶(3) [1, 2, 100]            VM533:1
<· undefined
> c[2]=80
<· 80
> console.log(a,c)
   ▶(3) [1, 2, 100] ▶(3) [1, 2, 80]             VM567:1
<· undefined
```

In the above example when copied variable value is changed but original variable value remain same .

2. Using Object.assign()

The Object.assign() method copies all enumerable own properties from one or more source objects to a target object. It returns the target object. Note this will be a shallow copy.



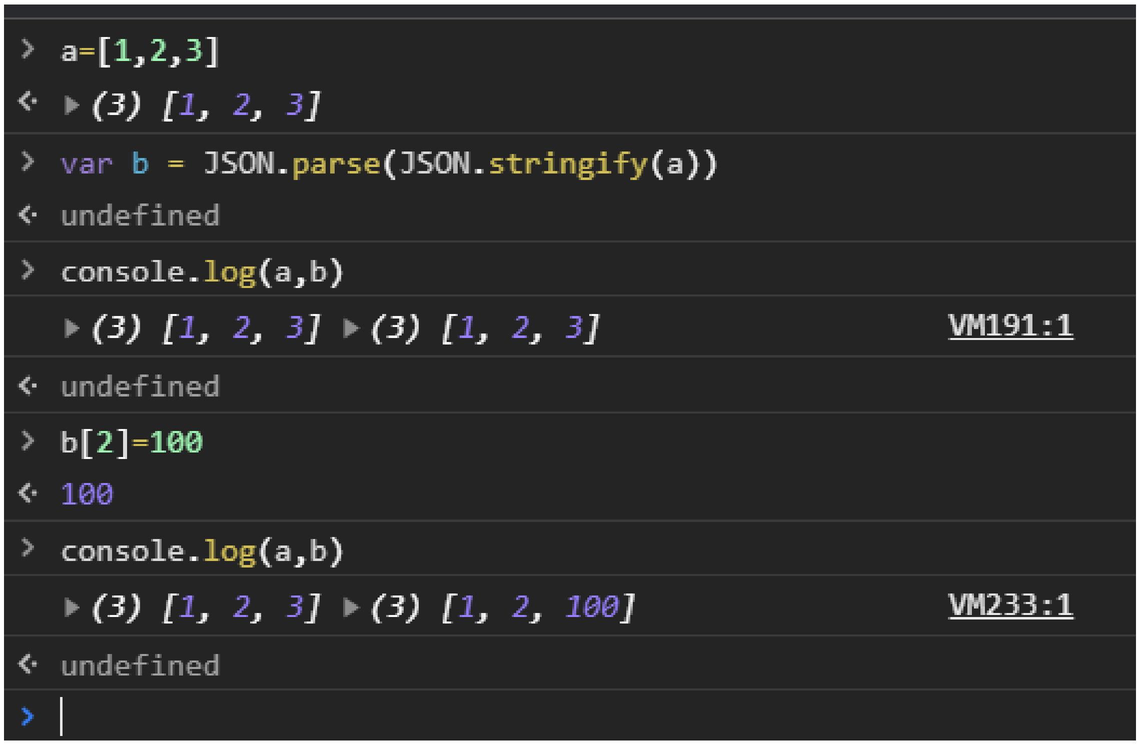
```
> var a =[1,2,3]

< undefined
> var b = Object.assign([],a)
< undefined
> console.log(a,b)
   ▸(3) [1, 2, 3] ▸(3) [1, 2, 3]                VM534:1
< undefined
> b[2]=100
< 100
> console.log(a,b)
   ▸(3) [1, 2, 3] ▸(3) [1, 2, 100]              VM569:1
< undefined
```

Note the empty [] as the first argument, this will ensure you don't mutate the original object

3. Using JSON.parse() and JSON.stringify()

The JSON object, available in all modern browsers, has two useful methods to deal with JSON-formatted content: parse and stringify. JSON.parse() takes a JSON string and transforms it into a JavaScript object. JSON.stringify() takes a JavaScript object and transforms it into a JSON string.Using JSON.parse() and JSON.stringify() for copy performs deep copy.



```
> a=[1,2,3]
<· ▸ (3) [1, 2, 3]
> var b = JSON.parse(JSON.stringify(a))
<· undefined
> console.log(a,b)
  ▸ (3) [1, 2, 3] ▸ (3) [1, 2, 3]                         VM191:1
<· undefined
> b[2]=100
<· 100
> console.log(a,b)
  ▸ (3) [1, 2, 3] ▸ (3) [1, 2, 100]                       VM233:1
<· undefined
> |
```

The deep copy is a true copy for nested objects. Shallow copy copies only reference in case of nested objects.

5.JavaScript Practice problems in JSON(Objects) and List

```
var cat = {
 name: 'Fluffy',
 activities: ['play', 'eat cat food'],
 catFriends: [
 {
 name: 'bar',
 activities: [‘be grumpy’, ‘eat bread omblet’],
 weight: 8,
 furcolor: ‘white’
 },
 {
 name: ‘foo’,
 activities: [‘sleep’, ‘pre-sleep naps’],
 weight: 3
 }
 ]
}
console.log(cat);
```

Basic Tasks to play with JSON:

Add height and weight to Fluffy:

cat.weight=5;

cat.height=4;

Fluffy name is spelled wrongly. Update it to Fluffyy:

cat.name="fluffyy";

List all the activities of Fluffyy’s catFriends.

```
for(let i in cat.catFriends)
{
 console.log(cat.catFriends[0].activities);
}
```

Print the catFriends names:

```
for(let i in cat.catFriends)
{
   console.log(cat.catFriends[i].name);
}
```

Print the total weight of catFriends

```
sum=0;
for(let i in cat.catFriends)
{
  sum=sum+cat.catFriends[i].weight;
}
console.log(sum);
```

Print the total activities of all cats

```
sum=0;
for(let i in cat.catFriends)
{
  sum=sum+cat.catFriends[i].activities.length;
}
console.log(sum)
```

Add 2 more activities to bar & foo cats

```
cat.catFriends[0].activities[2]="sleep";
cat.catFriends[0].activities[3]="pre-sleep naps";
cat.catFriends[1].activities[2]="be grumpy";
cat.catFriends[1].activities[3]="eat bread omblet";
console.log(cat.catFriends[0].activities);
console.log(cat.catFriends[1].activities);
```

Update the fur color of bar

```
cat.catFriends[0].furcolor="black";

console.log(cat.catFriends[0].furcolor);
```

Iterating with JSON object's Values

```
var myCar = {
make: ‘Bugatti’,
model: ‘Bugatti La Voiture Noire’,
year: 2019,
accidents: [
{
```

```
date: ‘3/15/2019’,
damage_points: ‘5000’,
atFaultForAccident: true
},
{
date: ‘7/4/2022’,
damage_points: ‘2200’,
atFaultForAccident: true
},
{
date: ‘6/22/2021’,
damage_points: ‘7900’,
atFaultForAccident: true
}
]
```

1. Loop over the accidents array. Change atFaultForAccident from true to false.

```
for(let i in myCar.accidents)
  {
  myCar.accidents[i].atFaultForAccident=false;
  }
  console.log( myCar.accidents);
```

2. Print the dated of my accidents

```
  for(let i in myCar.accidents)
  {
  console.log(myCar.accidents[i].date)
  }
```

Write a function called “printAllValues” which returns an newArray of all the input object’s values

```
var object = {name: "RajiniKanth", age: 33, hasPets : false};
function printAllValues(obj) {
 console.log(Object.values(object));
}
printAllValues();
```

Write a function called “printAllKeys” which returns an newArray of all the input object’s keys.

```
var object = {name: "RajiniKanth", age: 33, hasPets : false};
function printAllValues(obj) {
 console.log(Object.keys(object));
}
printAllValues();
```

Write a function to return the list of characters below 20 age

```
var students = [
   {name: 'Siddharth Abhimanyu', age: 21},
   { name: 'Malar', age: 25},
   {name: 'Maari',age: 18},
   {name: 'Bhallala Deva',age: 17},
   {name: 'Baahubali',age: 16},
   {name: 'AAK chandran',age: 23},
   {name:'Gabbar Singh',age: 33},
   {name: 'Mogambo',age: 53},
   {name: 'Munnabhai',age: 40},
   {name: 'Sher Khan',age: 20},
   {name: 'Chulbul Pandey',age: 19},
   {name: 'Anthony',age: 28},
   {name: 'Devdas',age: 56}
   ];
  function returnMinors(arr)
  {
      for(let i in students)
      {
      if(students[i].age<=20)
      {
       console.log( students[i].name);
      }
      }

  }
  console.log(returnMinors(students));
```

Write a function called “convertObjectToList” which converts an object literal into an array of arrays.Write a function called “convertObjectToList” which converts an object literal into an array of arrays.

```
var object = {name: 'ISRO', age: 35, role: 'Scientist'};
function convertListToObject(obj) {
 // your code here
 console.log(Object.entries(object));
}
convertListToObject();
```

Write a function ‘transformFirstAndLast’ that takes in an array, and returns an object with:

```
var arr = ['GUVI', 'I', 'am', 'a geek'];
function transformFirstAndLast(arr) {
   let newObject = {};
   let arrLength = arr.length;
   newObject[arr[0]] = arr[arrLength-1];
   return newObject;;
}
console.log(transformFirstAndLast(arr))
```

Write a function called “transformGeekData” that transforms some set of data from one format to another.

```
var arr= [[['firstName', 'Vasanth'], ['lastName', 'Raja'], ['age', 24], ['role', 'JSWizard']],
[['firstName', 'Sri'], ['lastName', 'Devi'], ['age', 28], ['role', 'Coder']]];
function transformEmployeeData(arr) {
   var transformEmployeeList = [];

   for(let i = 0 ; i < arr.length ; i++){
      let myobject = {};
      for(let j=0 ; j<arr[i].length ; j++){
         myobject[arr[i][j][0]] = arr[i][j][1] ;
      }
      transformEmployeeList.push(myobject);
   }
   return transformEmployeeList;
  }
  console.log(transformEmployeeData(arr));
```

Write a function “fromListToObject” which takes in an array of arrays, and returns an object with each pair of elements in the array as a key-value pair.

```
var arr = [['make', 'Ford'], ['model', 'Mustang'], ['year', 1964]];
function fromListToObject(arr) {
   var newObject = {};
```

```
    for(let i=0;i<arr.length;i++)
    {
   newObject[arr[i][0]]=arr[i][1];
    }

  return newObject ;
   }

  console.log(fromListToObject(arr));
```

Write an “assertObjectsEqual” function from scratch.
Assume that the objects in question contain only scalar values (i.e., simple values like strings or numbers).
It is OK to use JSON.stringify().
Note: The examples below represent different use cases for the same test. In practice, you should never have multiple tests with the same name.

```
var expected = {foo: 5, bar: 6};
var actual = {foo: 5, bar: 9}
function assertObjectsEqual(actual, expected, testName){
    let actualString = JSON.stringify(actual);
    let expectedString = JSON.stringify(expected);
    if(actualString!==expectedString){
        console.log("Failed [" +testName+ "] Expected" + expectedString + "but got" +
actualString);
    }
    else
        testName = console.log("Passed");
    return testName;

}
assertObjectsEqual(actual, expected, 'detects that two object are equal')
```