

TASK 1

1)Diff btw HTTP1.1 vs HTTP2?

Because HTTP/1.1 relies on the transport layer to avoid buffer overflow, each new TCP connection requires a separate flow control mechanism. **HTTP/2**, however, multiplexes streams within a single TCP connection, and will have to implement flow control **in a** different manner.

- HTTP2 is binary, instead of textual
- HTTP2 is fully multiplexed, instead of ordered and blocking
- HTTP2 can, therefore, use one connection for parallelism
- HTTP2 uses header compression to reduce overhead
- HTTP2 allows servers to “push” responses proactively into client caches
- HTTP2 Loads faster than HTTP 1 due pipelining in HTTP1.1

2)HTTP version History?

| Year | HTTP Version |
|------|--------------|
| 1991 | 0.9 |
| 1996 | 1.0 |
| 1997 | 1.1 |
| 2015 | 2.0 |

| | |
|--------------|-----|
| Draft (2020) | 3.0 |
|--------------|-----|

3)Diff btw browser js vs node js?

S.
No

Javascript

NodeJS

1. Javascript is a programming language that is used for writing scripts on the website.

NodeJS is a Javascript runtime environment.

2. Javascript can only be run in the browsers.

NodeJS code can be run outside the browser.

3. It is basically used on the client-side.

It is mostly used on the server-side.

- | | | |
|----|---|--|
| 4. | Javascript is capable enough to add HTML and play with the DOM. | Nodejs does not have capability to add HTML tags. |
| 5. | 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. |
| 6. | Javascript is used in frontend development. | Nodejs is used in server-side development. |
| 7. | Some of the javascript frameworks are RamdaJS, TypedJS, etc. | Some of the Nodejs modules are Lodash, express etc. These modules are to be imported from npm. |
| 8. | It is the upgraded version of ECMA script that uses Chrome's V8 engine written in C++. | Nodejs is written in C, C++ and Javascript. |

4)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.

TASK 2

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.

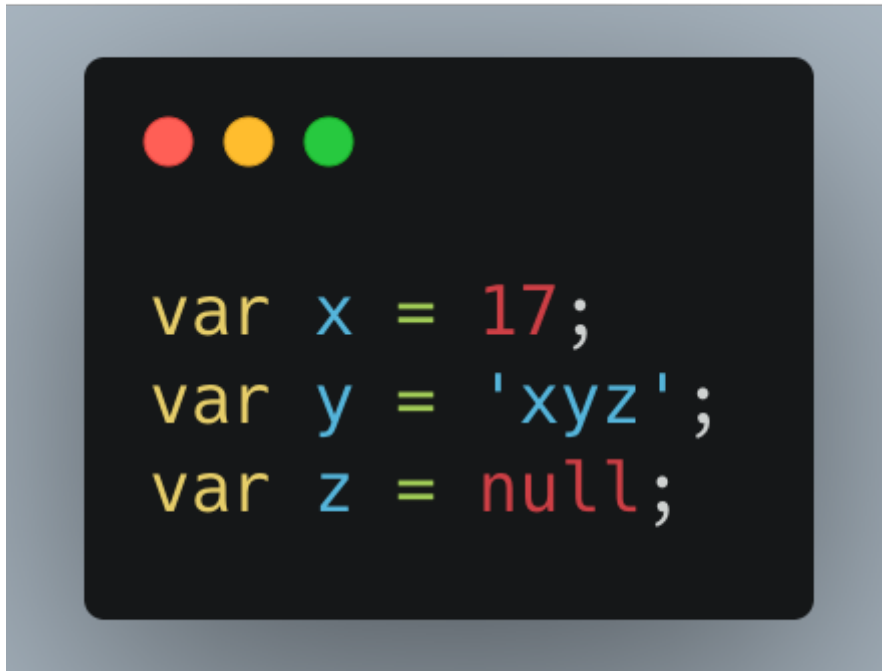


Figure 1: Variable assignment

In Figure 1, `x` contains value `17`, `y` contains `'xyz'`.

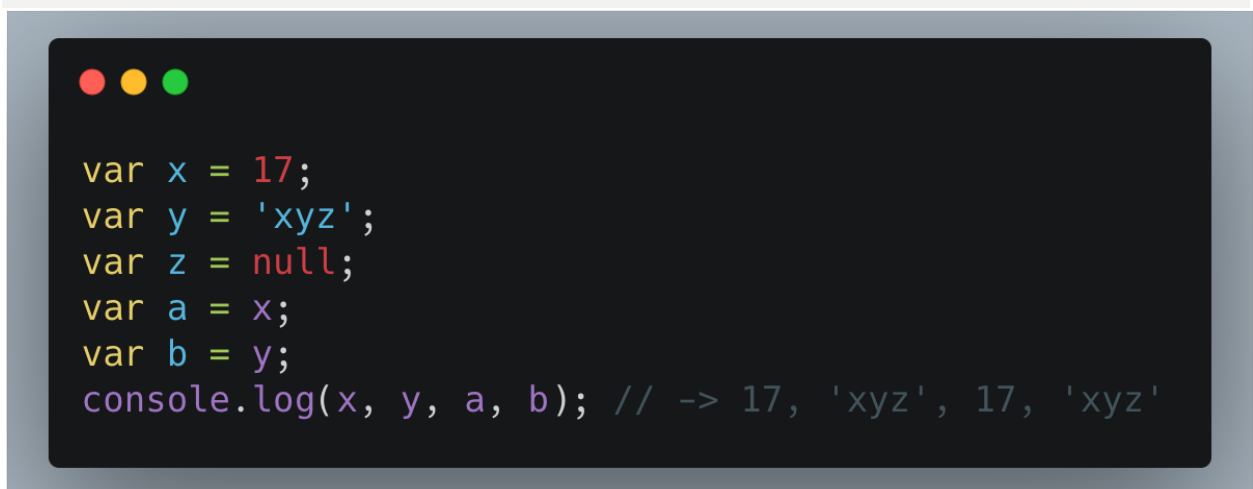
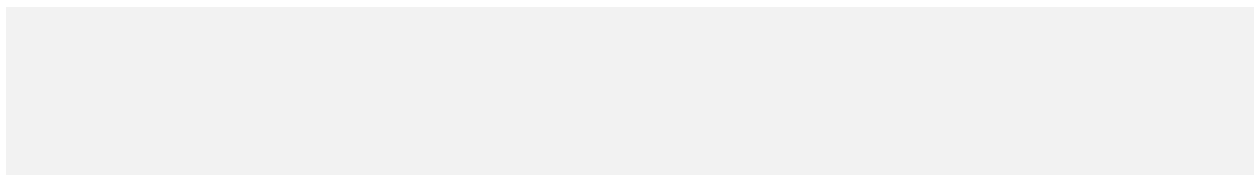


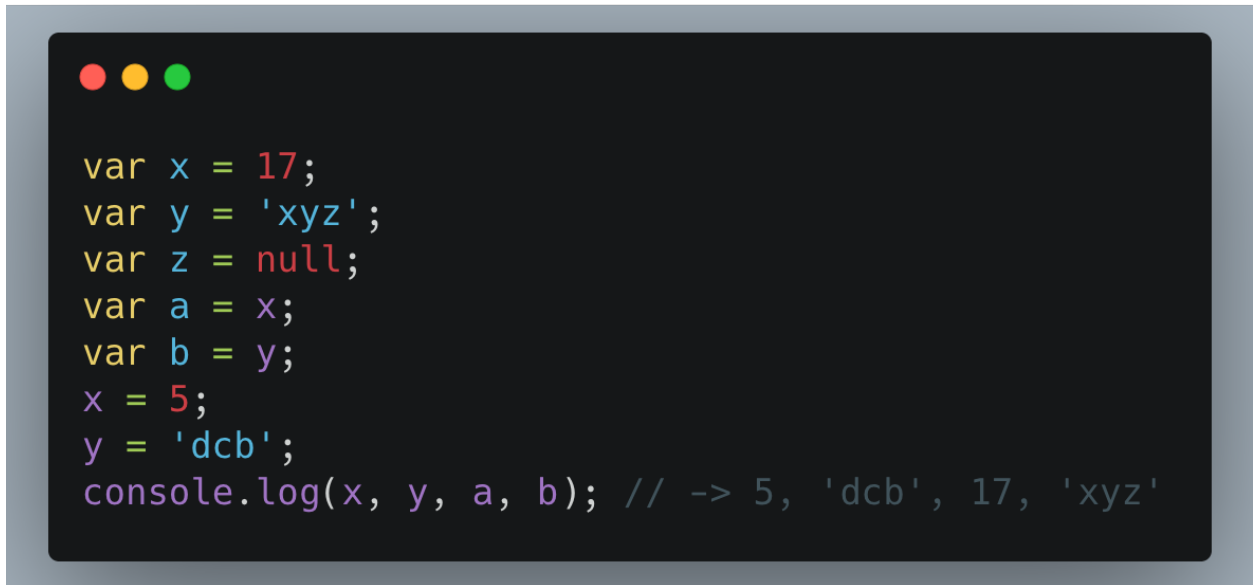
Figure 2: Variables are copied

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 place in one does not affect the other.





```
var x = 17;
var y = 'xyz';
var z = null;
var a = x;
var b = y;
x = 5;
y = 'dcb';
console.log(x, y, a, b); // -> 5, 'dcb', 17, 'xyz'
```

Figure 3: Variables are independent of each 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'
```

Figure 4: Copy by reference

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 as well.

3)How to copy by value a composite data type (array+objects).

One of the ways of copying the composite data type by value is achieved with the help of spread operator (...).

```
> let array1=[2,4,6,8,10];
  console.log("Original Array:",array1)
  let array2=[...array1];
  array1[0]=0;
  let array3=array1;
  console.log("After Updating original array:",array1);
  console.log("Copy by Value using spread operator:",array2);
  console.log("Copy by reference",array3);
```

| | |
|--------------------------------------|------------------------|
| Original Array: | ▶ (5) [2, 4, 6, 8, 10] |
| After Updating original array: | ▶ (5) [0, 4, 6, 8, 10] |
| Copy by Value using spread operator: | ▶ (5) [2, 4, 6, 8, 10] |
| Copy by reference | ▶ (5) [0, 4, 6, 8, 10] |

CopyByValue using SpreadOperator for Composite data type

In the above image, the array2 has the elements of array1 copied into it. Any changes made to array1 will not be reflected in array2 and vice versa.

In the same program, array3 which is copied as a reference had its value changed and the changes made in one array would reflect in the other array which in most cases is undesirable.

Hence with the help of **Spread operator(...)** ,we can copy by value for composite data types.