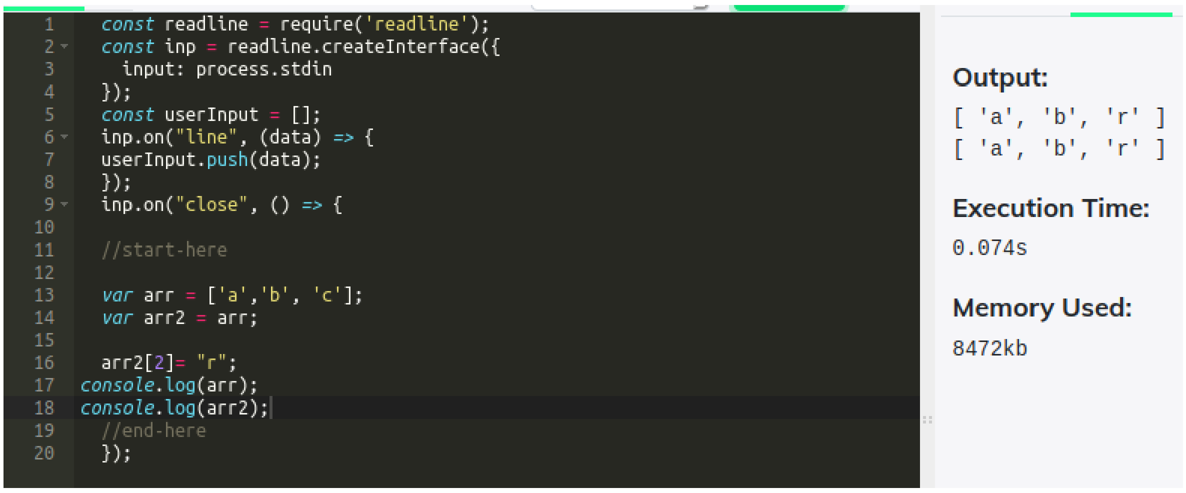
day 2 (06/11/2020)

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

sol. Copy by value method is only effective for primitive datatype. but when we consider composite datatype values are copied by copy of reference. Example to understand better.



```javascript
const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
userInput.push(data);
});
inp.on("close", () => {

//start-here

var arr = ['a','b', 'c'];
var arr2 = arr;

arr2[2]= "r";
console.log(arr);
console.log(arr2);
//end-here
});
```

Output:

```
[ 'a', 'b', 'r' ]
[ 'a', 'b', 'r' ]
```

Execution Time:

`0.074s`

Memory Used:

`8472kb`

If we consider above example arr and arr2 share the same location for array. So if we make any changes to the location it will change both the values, which you wont be able to find in primitive datatype.

2) why there is a difference in behaviour for copying contents in primitive and non primitive type?

sol. Major difference copying content from primitive datatype and non-primitive datatype is copy by value will fail if there is huge amount of data (in millions) present in either in array or object.

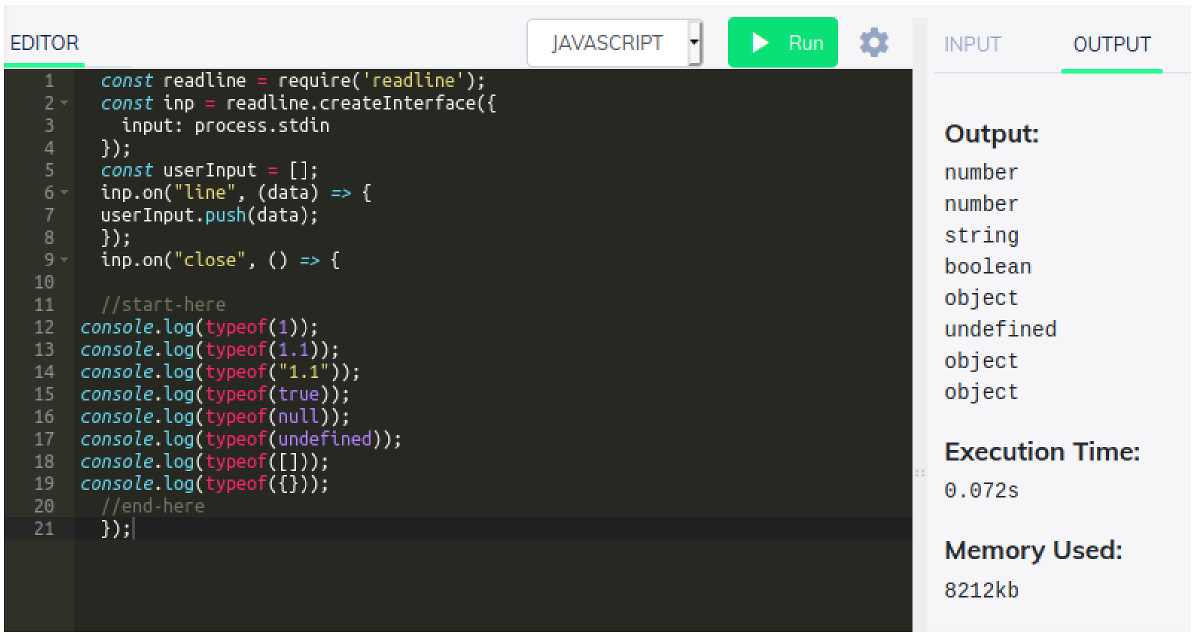
So copy of reference will work efficiently, because changes would made in the memory location and that same location is shared by other variables, where in copy by value method we have declare value for every changes.

---

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

- typeof(1)
- typeof(1.1)
- typeof("1.1")
- typeof(true)
- typeof(null)
- typeof(undefined)
- typeof([])
- typeof({})

sol.



```
const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
userInput.push(data);
});
inp.on("close", () => {

  //start-here
console.log(typeof(1));
console.log(typeof(1.1));
console.log(typeof("1.1"));
console.log(typeof(true));
console.log(typeof(null));
console.log(typeof(undefined));
console.log(typeof([]));
console.log(typeof({}));
  //end-here
  });
```

Output:

```
number
number
string
boolean
object
undefined
object
object
```

Execution Time:

```
0.072s
```

Memory Used:

```
8212kb
```

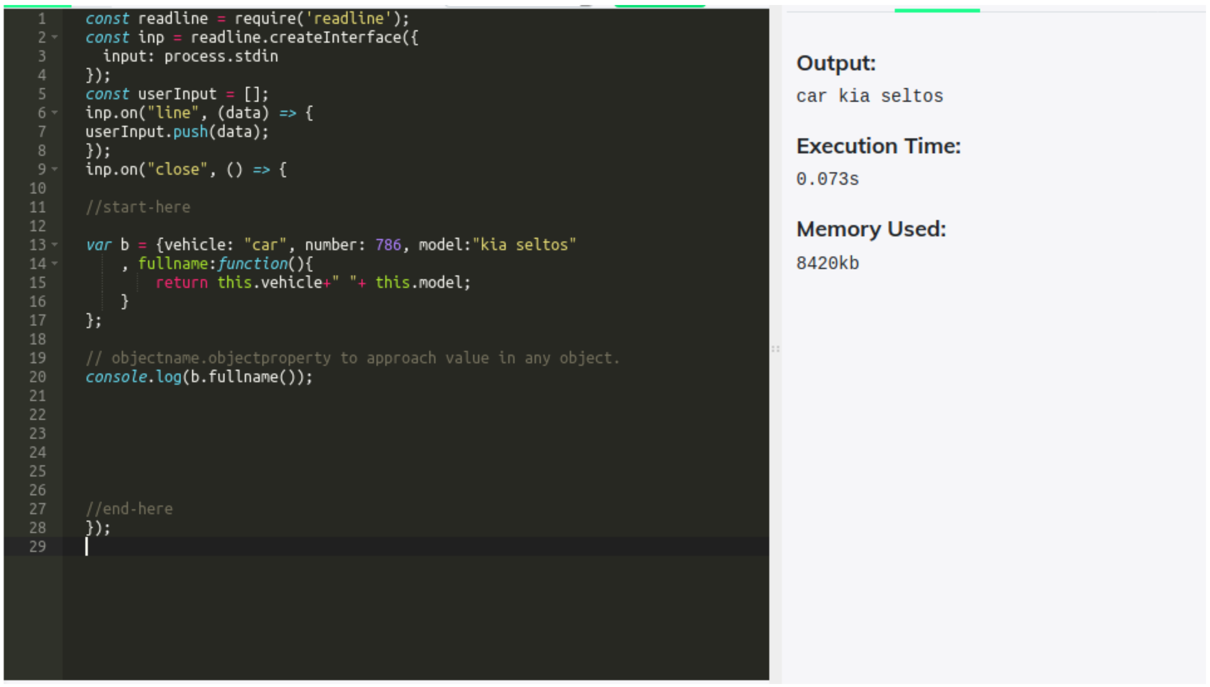
4) Write a blog about objects and its internal representation in Javascript?

sol. Objects:

It consist of key value pair and mostly everything in javascript is an object.

- Booleans can be objects
- number can be object
- string can be object
- dates can be object
- math are always objects
- Array,function and regular expression are always objects.

objects can also have methods. Methods are action which are performed in objects.



```
const readline = require('readline');
const inp = readline.createInterface({
  input: process.stdin
});
const userInput = [];
inp.on("line", (data) => {
userInput.push(data);
});
inp.on("close", () => {

//start-here

var b = {vehicle: "car", number: 786, model:"kia seltos"
    , fullname:function(){
        return this.vehicle+" "+ this.model;
    }
};

// objectname.objectproperty to approach value in any object.
console.log(b.fullname());






//end-here
});
```

Output:

```
car kia seltos
```

Execution Time:

```
0.073s
```

Memory Used:

```
8420kb
```

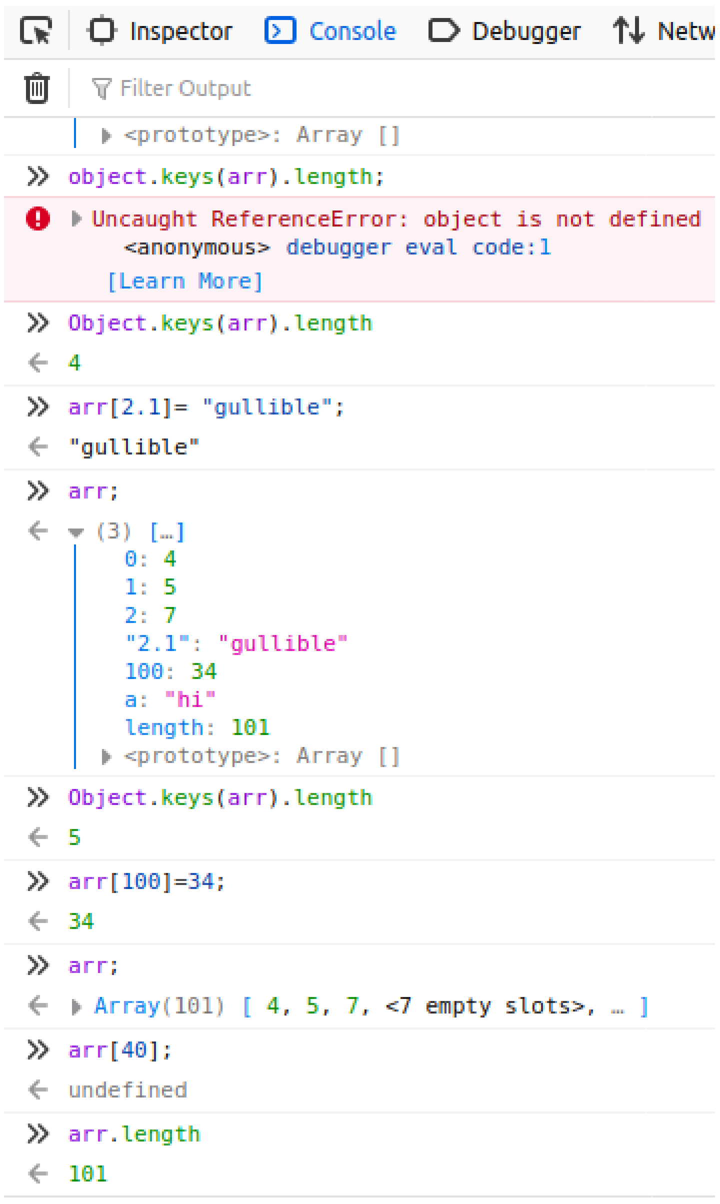
- Do not declare strings, number, booleans as objects. when we make a new variable in it is declared as object.
- all the objects are copied by copy of reference

Object internals: Array is also an JSON (java script object notation)

- everything is JSON object except primitves.
- A javascript array is exclusively numerically indexed.

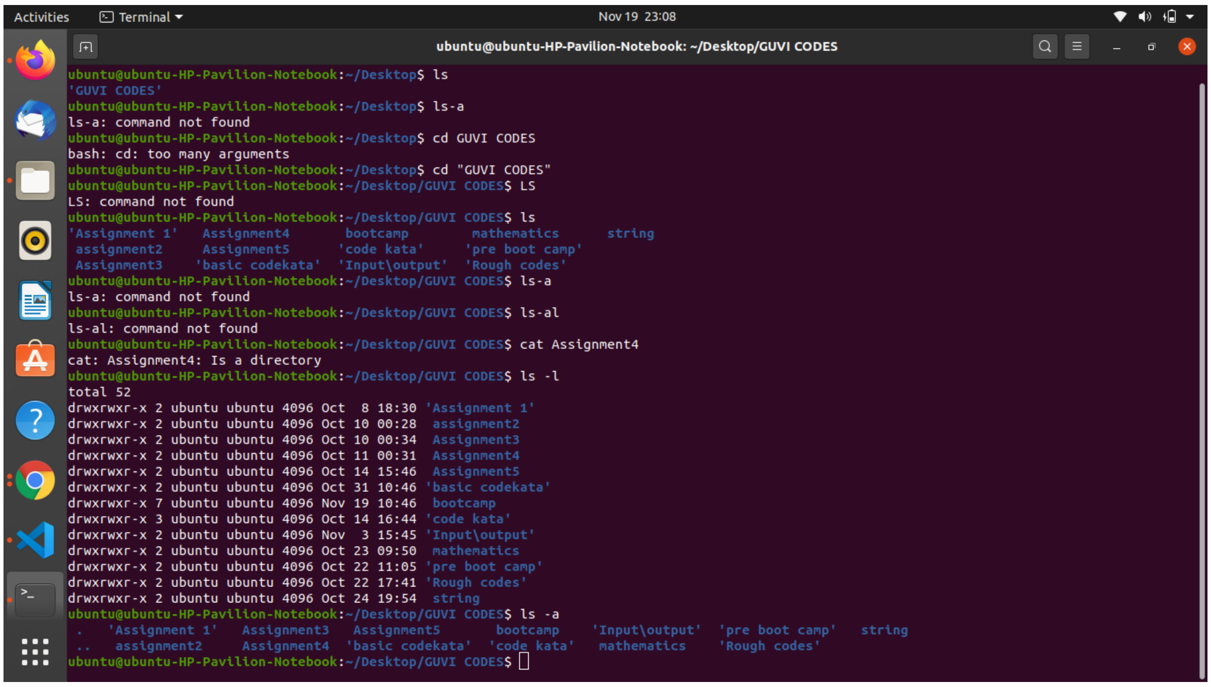
- javascript arrays cannot have "string indexes.
- when you set a "string index", you're setting a property of the array(object).

Example for understanding how decimal are converted to string and treated like properties.

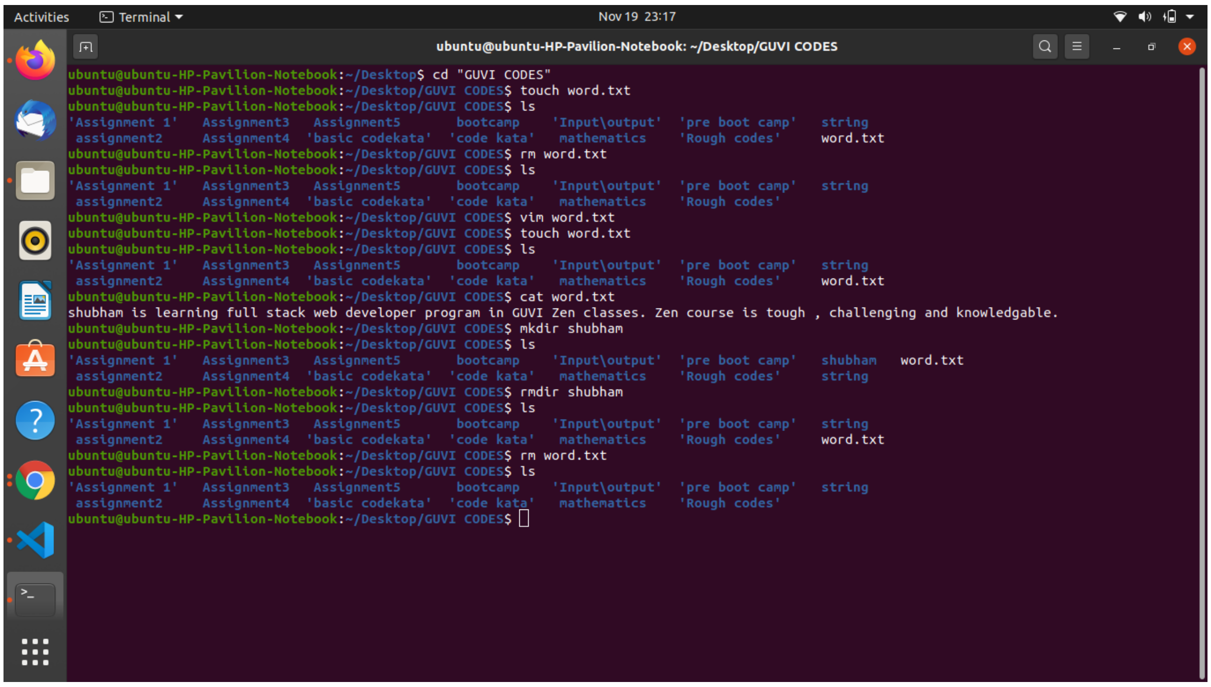


```
  ▸ <prototype>: Array []
>> object.keys(arr).length;
❗ ▸ Uncaught ReferenceError: object is not defined
      <anonymous> debugger eval code:1
    [Learn More]
>> Object.keys(arr).length
← 4
>> arr[2.1]= "gullible";
← "gullible"
>> arr;
← ▾ (3) […]
      0: 4
      1: 5
      2: 7
      "2.1": "gullible"
      100: 34
      a: "hi"
      length: 101
    ▸ <prototype>: Array []
>> Object.keys(arr).length
← 5
>> arr[100]=34;
← 34
>> arr;
← ▸ Array(101) [ 4, 5, 7, <7 empty slots>, … ]
>> arr[40];
← undefined
>> arr.length
← 101
```

5)



```
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop$ ls
'GUVI CODES'
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop$ ls-a
ls-a: command not found
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop$ cd GUVI CODES
bash: cd: too many arguments
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop$ cd "GUVI CODES"
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ LS
LS: command not found
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ ls
'Assignment 1'   Assignment4       bootcamp        mathematics      string
 assignment2     Assignment5      'code kata'     'pre boot camp'
 Assignment3    'basic codekata'  'Input\output'  'Rough codes'
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ ls-a
ls-a: command not found
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ ls-al
ls-al: command not found
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ cat Assignment4
cat: Assignment4: Is a directory
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ ls -l
total 52
drwxrwxr-x 2 ubuntu ubuntu 4096 Oct  8 18:30 'Assignment 1'
drwxrwxr-x 2 ubuntu ubuntu 4096 Oct 10 00:28  assignment2
drwxrwxr-x 2 ubuntu ubuntu 4096 Oct 10 00:34  Assignment3
drwxrwxr-x 2 ubuntu ubuntu 4096 Oct 11 00:31  Assignment4
drwxrwxr-x 2 ubuntu ubuntu 4096 Oct 14 15:46  Assignment5
drwxrwxr-x 2 ubuntu ubuntu 4096 Oct 31 10:46 'basic codekata'
drwxrwxr-x 7 ubuntu ubuntu 4096 Nov 19 10:46  bootcamp
drwxrwxr-x 3 ubuntu ubuntu 4096 Oct 14 16:44 'code kata'
drwxrwxr-x 2 ubuntu ubuntu 4096 Nov  3 15:45 'Input\output'
drwxrwxr-x 2 ubuntu ubuntu 4096 Oct 23 09:50  mathematics
drwxrwxr-x 2 ubuntu ubuntu 4096 Oct 22 11:05 'pre boot camp'
drwxrwxr-x 2 ubuntu ubuntu 4096 Oct 22 17:41 'Rough codes'
drwxrwxr-x 2 ubuntu ubuntu 4096 Oct 24 19:54  string
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ ls -a
 .   'Assignment 1'   Assignment3   Assignment5       bootcamp     'Input\output'  'pre boot camp'   string
 ..   assignment2     Assignment4  'basic codekata'  'code kata'    mathematics    'Rough codes'
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ 
```



```
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop$ cd "GUVI CODES"
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ touch word.txt
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ ls
'Assignment 1'   Assignment3   Assignment5       bootcamp     'Input\output'  'pre boot camp'   string
 assignment2     Assignment4  'basic codekata'  'code kata'    mathematics    'Rough codes'     word.txt
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ rm word.txt
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ ls
'Assignment 1'   Assignment3   Assignment5       bootcamp     'Input\output'  'pre boot camp'   string
 assignment2     Assignment4  'basic codekata'  'code kata'    mathematics    'Rough codes'
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ vim word.txt
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ touch word.txt
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ ls
'Assignment 1'   Assignment3   Assignment5       bootcamp     'Input\output'  'pre boot camp'   string
 assignment2     Assignment4  'basic codekata'  'code kata'    mathematics    'Rough codes'     word.txt
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ cat word.txt
shubham is learning full stack web developer program in GUVI Zen classes. Zen course is tough , challenging and knowledgable.
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ mkdir shubham
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ ls
'Assignment 1'   Assignment3   Assignment5       bootcamp     'Input\output'  'pre boot camp'   shubham   word.txt
 assignment2     Assignment4  'basic codekata'  'code kata'    mathematics    'Rough codes'     string
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ rmdir shubham
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ ls
'Assignment 1'   Assignment3   Assignment5       bootcamp     'Input\output'  'pre boot camp'   string
 assignment2     Assignment4  'basic codekata'  'code kata'    mathematics    'Rough codes'     word.txt
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ rm word.txt
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ ls
'Assignment 1'   Assignment3   Assignment5       bootcamp     'Input\output'  'pre boot camp'   string
 assignment2     Assignment4  'basic codekata'  'code kata'    mathematics    'Rough codes'
ubuntu@ubuntu-HP-Pavilion-Notebook:~/Desktop/GUVI CODES$ 
```

6) window object:

it is default and supported by all browser, all global Javascript object, functions and variable automatically become members of the window object. Define browser specific condition. Ex: As alert is present in window.

document object:

it represent your web page, when ever you want to access any HTML page, you always start with accessing the document object.

Screen object:

It contain information of the viewers screen. It return height , width and color depth of the screen and return the colour resolution of the screen.