Pre-Bootcamp -

Expectation: Make everyone to same plane

https://www.youtube.com/playlist?list=PL_9uM5be2amqfJBrXdUf0dn2fggYWvG9P

Mandatory Outcome : Solving Abso Begin using JS

Tasks to be done after Pre-Bootcamp:

1. CodeKata Absolute beginner aka AB
    a. Submit the entire AB set with all test cases passed. This is mandatory to take the assessment.
    b. **Bonus points :** If you are an expert then go ahead and push forward.
2. Tasks to be submitted in github repos with name "**GUVI_prebootcamp**"
    a. Create a github account → https://github.com/
    b. How to upload file to repo → How to upload code onto github repository | How to push code from local repo to remote repo
    c. Create a repo with name "GUVI_prebootcamp" - pre-bootcamp and upload your js. Every task should have an individual js file.
        i. https://medium.com/@reach2arunprakash/guvi-zen-class-find-the-culprits-and-nail-them-9ee6c67c44fb
        ii. https://medium.com/@reach2arunprakash/www-guvi-io-zen-4fa483a7d359
        iii. https://medium.com/@reach2arunprakash/www-guvi-io-zen-d395deec1373

Why do we need this pre-bootcamp?

Learners fall into three buckets.

1. Beginner - Have logical skills but yet to start in code
2. Inter - Can apply logic in code, loops , array
3. Adv - LDS and Adv-DS

Concepts to be in covered pre-bootcamp:

1. Intro to Console in browser and <script> tag
2. Intro to Problem solving using Javascript (Code Kata & GUVI Ide)
3. Basics of JS -
    a. Variables - Numbers , string , boolean

b. NAN & undefined
c. Type casting - String to number (int,float) - parseInt , parseFloat , + , Number , String to boolean
d. Printing - console.log();
e. Looping
    i. Structure of looping with solved problem
    ii. Nested looping with an example
f. Conditions
g. Arrays
h. Objects
i. Function - basics - Don't go in depth

4. Codekata - Lil extra - any missing pieces
5. Test Cases - Space at end or beginning , number format , single line print

Assessment pattern

1. MCQ & Coding

1. MCQs - only in the JS topics covered in Pre-Bootcamp
2. Coding - Code Kata
    a. 2 questions - Arrays , Maths and String - Mandatory to attend
    b. 1 q - DS - skip if you don't know

Session 1: Intro to code kata and JS

How to do CodeKata:
https://medium.com/@reach2arunprakash/guvi-codekata-javascript-8d0e88d35630

Reference materials for Javascript:

1. JS Course in GUVI - Get unlocked from **Arun V**
2. https://github.com/reach2arunprakash/javascript-101
3. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks

Start here:

Where to run the code as first steps??

https://www.guvi.in/ide

This is the code template for reading the input in JS from code kata

Task 1:

1. Copy the below Code Template and paste it into https://www.guvi.in/ide
2. Paste output in the chat

Code Template:

```
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
//Your code goes here … replace this line with your code logic

//end-here
});
```

Output:

Output:

```
Nil
```

Execution Time:

```
0.072s
```

Memory Used:

```
8328kb
```

Next steps:

1. Print variable values - hardcoded
2. Add 2 variable and print
3. Read a variable and print - Input
4. Read split variables
5. Read two var and add
6. Read two var and compare
7. Array
    a. 11 - single variable
    b. ['11', '23', '45'] --Array
8. Read and add multiline / Read array of numbers - normal and looping

        1 2 3
        4 5 6
        7 8 9
9. Space at end
10. Test Cases
11. GitHub

```
var a = 10;
a = 40;
console.log(a);
var b = 20;

console.log(a+b);
```

Looping - Nested

```
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 total = 0;

for(var i = 0 ; i < userInput.length ; i = i +1)
{
   var dummy = userInput[i].split(" ");
   console.log(dummy);

    for(var j = 0 ; j< dummy.length ; j = j+1)
    {
        total = total + +dummy[j]

    }

}

//var dummy = ["1","2","3"];

console.log(total);


//end-here
});
```

```
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 sum = 0 ;

for (i=0; i<3; i = i+1 )
{
 something = userInput[i].split(" ");
for(j = 0 ; j < 3; j++)
{
  sum = sum + +something[j];
}
}

 console.log(sum);

//start-here
/*
var sum =0;
for(var x=0;x < userInput.length;x++)
{
    z = userInput[x].split(" ");
    for (var i=0;i<z.length;i++)     {
        sum = sum+parseInt(z[i]);   }

}

console.log(sum)
*/
//end-here
});
```

Further common Issues:

Array traversal
End space .join

Number:

```
let age = 10;
let mark = 80.09;
```

String/ char

```
let name = "arun";
let sex = 'm';

boolean

let pass = True;
let pass = False;
```

Typecasting

Input:

```
10
hai
true
1,23, arun , a
```

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

//console.log(userInput);

let i =0 ;
for(i=0;i<userInput.length;i++)
{
    console.log(typeof(userInput[i]))
}
let intvar = parseFloat(userInput[0]);
let strvar = userInput[1];
let bvar = (userInput[2] == 'true');
let arrvar = userInput[3].split(",");
```

```
console.log (typeof(intvar));
console.log (typeof(strvar));
console.log (typeof(bvar));
console.log (bvar);

console.log (arrvar);
console.log (typeof(arrvar));


/*
let bvar = true;
let arrvar = [10,10.3,"a","arun",23,false]
let objvar = {"name":"arun","age":100,"city":"chennai"}

console.log(typeof(intvar),typeof(strvar),typeof(bvar));

let i=0;

for(i = 0 ;i< arrvar.length;i++)
{
        console.log(arrvar[i]);
}


console.log(intvar.toFixed(2));
console.log(bvar);
console.log(arrvar);
console.log(objvar);
*/
//console.log();

//end-here
});
```

Session 2:

Hoisting :

1. Hoisting is moving up
2. Move only the var not the value

1. var is hoisted & function scope
2. let is not hoisted & its block scope

Reverse

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

var str = userInput[0];
var reverseStr = "";
for(var i = str.length-1; i>= 0; i--)
{
 reverseStr += str[i];
}


//end-here
});
```

Single line print

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

```
});
inp.on("close", () => {

var str = userInput[0].split(" ");
var ss = [];
var zz = "";
for(var i=0;i<str.length;i++)
{
   ss.push(str[i])
   zz+=str[i] + " "
   //console.log(str[i]);
}
console.log(ss.join(" "));
console.log(zz.trim());

//end-here
});
```

Objects:

Objects - JSON format --> K:V
JavaScript Object Notation

K:V
JSON
hashtable
hashmap
dict

How will to create a contact details code?

```
let name = [ "Arun","prakash","guvi"];
let number = [91768,123123,91764];
console.log(number[name.indexOf("Arun")]);

let details = { "Arun": 91768,"prakash": [123123,34534,435345],"guvi" :91764 }
console.log(details)
```

Class Task : Create a Car Object

```
{
   "brand1123": "BMW",
   "color":"icewater",
   "make":"icewater",
   "year":"icewater",
   "reported":"icewater",
   "wheels":3,
   "stepinie":4
}
```

Create array of Car Object

```
let cars = [{
   "brand1123": "BMW",
   "color":"icewater",
   "make":"icewater",
   "year":"icewater",
   "reported":"icewater",
   "wheels":3,
   "stepinie":4
},
{
   "brand1123": "Audi",
   "color":"icewater",
   "make":"icewater",
   "year":"icewater",
   "reported":"icewater",
   "wheels":3,
   "stepinie":4
}
,

{
   "brand1123": "Rolls",
   "color":"icewater",
   "make":"icewater",
   "year":"icewater",
   "reported":"icewater",
   "wheels":3,
   "stepinie":4
```

```
}
]
```

Cars
cars[0]
cars[1]["brand1123"]

```
var library = [
     {
          title : "Javascript",
          price : 500,
          readers : [
               {
                    name : "Person 1",
                    count : 300
               },
               {
                    name : "Person 2",
                    count : 400
               }
          ],
          authorDetails : {
               name : "Raj",
               age : 40
          }
     },
     {
          title : "Nodejs",
          price : 600,
          readers : [],
          authorDetails : {
               name : "Raj",
               age : 40
          }
     }
]
```

Technical Specifications

Dimensions			
Overall length		mm	3,500
Overall width		mm	1,600
Overall height		mm	1,490
Wheelbase		mm	2,360
Track width	Front	mm	1,405
	Rear	mm	1,400
Minimum turning radius		m	4.5
Minimum ground clearance		mm	170
Capacities			
Seating capacity		persons	5
Fuel tank capacity		litres	35
Engine			
Type			KB-Series
Number of cylinders			3
Number of valves			12
Capacity		cc/cm^3	998
Bore x stroke		mm	73.0 x 79.5

Compression ratio			10:1
Maximum power		PS/rpm	67/6,200
Maximum torque		Nm/rpm	90/3,500
Fuel distribution			Multipoint Injection
Transmission			
Type			5-speed MT
Chassis			
Steering			Rack & Pinion, Power assisted
Brakes	Front		Ventilated discs
	Rear		Drums
Suspension	Front		MacPherson strut & coil spring
	Rear		Isolated trailing link & coil spring
Shock absorbers			Gas filled
Tyre (tubeless)			155/80R13
Weights			
Kerb weight (min. with full options)		kg	860-880
Gross vehicle weight		kg	1,320

Count duplicates:

```
let elem = [12,12,12,34,34,45,45,56,67,67,78,78,78,78,78]

let count = {};

for(i=0;i<elem.length;i++)
{

  if (count[elem[i]] === undefined)
  {
    count[elem[i]] = 1;
  }
  else
  {
    count[elem[i]] = count[elem[i]] + 1
  }

}
```

Problems to Solve:

1. 2D array sum
2. Sum of each row from a 2D array and print in a single row
3. In String, remove vowels
4. Sort the array elements in descending order based on the number of 1’s in its binary representation.
5. Frequency sort using objects.