1) **What is the difference between window, screen, and document in Javascript?**

*The **window** object is the global object. It is on the top of the hierarchy and consists of a number of other objects/properties associated with it. It actually represents the browser window.*

*The **Document** is the content that is loaded on to the webpage. The HTML page is converted into a Document Object Model Tree by the browser.*

*On the other hand, the **Screen** is a property of the window object and it has properties related to the user's display.*

2) **JSON task**

https://medium.com/@reach2arunprakash/guvi-zen-code-sprint-javascript-practice-problems-in-json-objects-and-list-49ac3356a8a5

Problem 0 : Part A

Playing with JSON object's Values:

```
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:

Add height and weight to Fluffy

```
cat.height = 25;
cat.weight = 4;
```

Fluffy name is spelled wrongly. Update it to Fluffyy

```
cat.name = 'Fluffyy';
```

List all the activities of Fluffyy's catFriends.

```
function activity()
{
var b= [];
for(var i = 0; i<cat.catFriends.length; i++)
        b.push(cat.catFriends[i].activities)
return b;
}

console.log(activity())
```

Print the catFriends names.

```
function names()
{
var b = [];
for(var i = 0; i<cat.catFriends.length; i++)
        b.push(cat.catFriends[i].name);
return b;
}
console.log(names())
```

Print the total weight of catFriends

```
function totalWeight()
{
var b = [];
var c;
for(var i = 0; i<cat.catFriends.length; i++)
       b.push(cat.catFriends[i].weight);
for(var j = 0; j<b.length; j++)
      c= c + parseInt(b[j]);
return c;
}
console.log(totalWeight())
```

Print the total activities of all cats (op:6)

```
function allActivity()
{
var b= [];
b.push(cat.activities);
for(var i = 0; i<cat.catFriends.length; i++)
       b.push(cat.catFriends[i].activities)
return b;
}

console.log(allActivity())
```

Add 2 more activities to bar & foo cats

```
function addActivity()
{
for(var i = 0; i<cat.catFriends.length; i++)
       cat.catFriends[i].activities.push('purr');
       cat.catFriends[i].activities.push('scratch');

return cat;
}

console.log(addActivity())
```

Update the fur color of bar

```
cat.catFriends[0].furcolor = 'brown';
```

Problem 0 : Part B (15 mins):

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.

```
function update()
{
   for(var i = 0; i<myCar.accidents.length; i++)
        myCar.accidents[i].atFaultForAccident = false;
   return myCar
}
console.log(update())
```

2. Print the dates of my accidents

```
function dateofAccs()
{
  var b = [];
```

```
    for(var i = 0; i<myCar.accidents.length; i++)
        b.push(myCar.accidents[i].date);
   return b;
}
console.log(dateofAccs());
```

Real challenges starts here

1. Parsing an JSON object's Values:

Write a function called "printAllValues" which returns an newArray of all the input object's values.

```
var pup = {name: "Bruno", age: 1.5, lovesFood: true};
function printAllValues()
{
  return (Object.values(pup));
}
```

Output: ["Bruno", 1.5, true]

2. Parsing an JSON object's Keys:

Write a function called "printAllKeys" which returns an newArray of all the input object's keys.

```
var pup = {name: "Bruno", age: 1.5, lovesFood: true};
function printAllKeys()
{
  return (Object.keys(pup));
}
```

Output: [name, age, lovesFood]

3. **Parsing an JSON object and convert it to a list:**

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

```
var fun = {university: "DTU", age: 28, subject: "Space Physics" };
function convertObjectToList()
{
    let cList = Object.entries(fun);
    return cList;
}
```

Output:

0: Array ["university", "DTU"]

1: Array ["age", 28]

2: Array ["subject", "Space Physics"]

length: 3

4. **Parsing a list and transform the first and last elements of it:**

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

```
function transformFirstAndLast() {
var foo = [];
var jump = [];
foo.push(fun[0]);
foo.push(fun[fun.length - 1]);
jump.push(foo);
let blu = Object.fromEntries(jump);
return blu;
}
console.log(transformFirstAndLast());
```

Output: Object { Love: "Yourself" }

5. **Parsing a list of lists and convert into a JSON object:**

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 = [["Chrome","v8"],["Mozilla","SpiderMonkey"],["Safari","Nitro"]];
function fromListToObject()
{
var newObj = Object.fromEntries(arr);
return newObj;
}
console.log(fromListToObject());
```

Output: { Chrome: "v8", Mozilla: "SpiderMonkey", Safari: "Nitro" }

6. **Parsing a list of lists and convert into a JSON object:**

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

```
var ss = [["name", "sun"], ["isStar", true], ["name", "moon"], ["isStar", false]];
var a = [];
var b = [];
for (var j = 1; j< ss.length; j++)
      if (ss[j][0] == ss[0][0])
         for (var i =0;i<j; i++)
               a.push(ss[i]);
for (var l = 0; l< ss.length; l++)
     if (ss[l] !== a[l])
          b.push(ss[l]);
Object.fromEntries(a);
Object.fromEntries(b);
var c = [];
c.push(a);
c.push(b);
console.log(c);
```

Output: **[**{ name: "sun", isStar: true }, { name: "moon", isStar: false }]

7. Parsing two JSON objects and Compare:

```
var expected = {foo: 5, bar: 6};
var actual = {foo: 5, bar: 6}
var exd = JSON.stringify(expected);
var act = JSON.stringify(actual);
function assertObjectsEqual(ac, ex)
{
   if (ac == ex)
     return (“Passed”);
   else
      return (“Failed”);
}
console.log(assertObjectsEqual(act, exd));
```

Output: “Passed”

8. Parsing JSON objects and Compare:

```
var a = [{q: "what's your pet's name?", a: "Aaran"}, {q: "which is your dream place?", a:
"Antarctica"}];

function check(qt, as)
{
    for(var i = 0; i<a.length; i++)
          if(qt == a[i].q)
               if(as == a[i].a)
                    return true;
          else
               var c = 0;
    if (c === 0)
       return false;
}

var qb = “which is your dream place?”;
var ad = “Antarctica”;
console.log(check(qb,ad));
```

Output: true

9. **Parsing JSON objects and Compare:**

```
var y = [{name: "g", age: 23}, {name: "p", age: 17}, {name: "i", age: 5}];
function kids()
{
   var b = [];
   for (var i = 0; i< y.length; i++)
        if(y[i].age < 20)
           b.push(y[i].name);
   return b;
}
console.log(kids());
```

Output: [‘p’, ’i’]

3) Error handling in XMLHttpRequest

When the XMLHttpRequest fails due to some error, *XMLHttpRequestEventTarget.onerror*, is the function that has to be called.

Syntax:

XMLHttpRequest.onerror = callback;

Where callback will be the function that has to be executed when the error occurs.

This way of error handling is compatible with major browsers such as Chrome, Firefox, etc,.

4) Convert the ppt into a writeup
https://www.slideshare.net/nzakas/enterprise-javascript-error-handling-presentation/25-C_ommunication_E_rrors_Invalid

Developers/ designers should take into account all kinds of probable errors and design a system with minimal errors. They should design it in such a way that even if an error is encountered, it should be detected easily and have least consequences.
(As described by Donald A. Norman in The Design of Everyday Things)

Rules to keep in mind:

1) Assume your code will fail: Example- Think like what if the destination/source is null?
2) Log errors to the server instead of client.
3) The developer should handle the errors through the code instead of the browser. Example: One can use the “Try-Catch” method or window.onerror method.

The error propagates in this order: Detection -> Try-Catch -> window onerror -> Browser error

4) Look through the code and identify where the error might occur.

Types of errors: Type coercion, data type, communication

5) Throw your own errors and check
6) *Important:* Distinguish between fatal and non-fatal errors.
7) Provide a debug mode.

5) Find the fix for the error slide

i) Uncaught TypeError: Cannot read property- This occurs when the value to be operated on is not compatible with the type expected by the browser. This type of error is missed by the try-catch method.

ii) TypeError: undefined is not an object- This occurs when a property is accessed or a method is called on an undefined object.

iii) TypeError: null is not an object: This might occur if you try using a DOM element in the program before the element is loaded since, the DOM API would return null for the object references that are blank.

iv) (unknown): script error- This error occurs when the JS code is hosted on a different domain. To fix, we can set Access-Control-Allow-Origin: * on the JS file, and crossorigin="anonymous" on the <script> tag.

v) TypeError: Object doesn’t support property- This occurs when we try to use a method or property that the specified object does not support.

vi) TypeError: undefined is not a function- This type of error occurs when attempting to call a value like a function, when it is not a function.

6) How to compare two JSON have the same properties without order?
var obj1 = {"name":"GUVI","class":"FS"};

var obj2 = {"class":"FS","name":"GUVI"};

Using _.isEqual method

Syntax: _.isEqual(JSONObject1, JSONObject2);

7) Mandatory Tasks - Zen tasks

i) UPLOAD TO GIT: WARMUP TASKS:

https://github.com/Gaya3-bytes/LearningJS/blob/master/basicFunctionTasks

ii) Do the below programs in anonymous function & IIFE

Print odd numbers in an array

```
var oddNumbers = function(arr) { var c = arr.filter(x => if(x%2 !== 0){return x;}) return c;);}
oddNumbers(arr)

(function() { var c = arr.filter(x => if(x%2 !== 0){return x;}) return c;); })(arr)
```

Convert all the strings to title caps in a string array

```
var ss = function(str)
{
  var a1 = str.split(' ');
  var array1 = [];

  for(var x = 0; x < a1.length; x++){
      array1.push(a1[x].charAt(0).toUpperCase()+a1[x].slice(1));
  }
  return array1.join(' ');
}
ss("the quick brown fox"));



var ss = function(str)
{
  var a1 = str.split(' ');
  var array1 = [];
```

```
  for(var x = 0; x < a1.length; x++){
      array1.push(a1[x].charAt(0).toUpperCase()+a1[x].slice(1));
  }
  return array1.join(' ');
}("the quick brown fox"));
```

Sum of all numbers in an array

```
var sum =  function(arr)
{
  Var s = 0;
   for (var i = 0; i<arr.length; i++)
        s += arr[i];
  Return s;
}
sum(arr)
```

```
var sum =  function(arr)
{
  Var s = 0;
   for (var i = 0; i<arr.length; i++)
        s += arr[i];
  Return s;
}(arr)
```

Return all the prime numbers in an array

```
Function prime(n)
{
  for(var i = 0; i<n; i++)
       if(n%i == 0)
         break;
        else
          return n;
}

var p = function(arr) { var c = arr.filter(x => prime(x));return c; }
p(arr);

var p = function(arr) { var c = arr.filter(x => prime(x));return c; }(arr)
```

Return all the palindromes in an array

```
function palindrome(str) {
var lows = str.toLowerCase()
var rs = lows.split('').reverse().join('');
if( rs === lows)
  return str;
}

var p = function(arr) { var c = arr.filter(x => palindrome(x));return c; }
p(arr);

var p = function(arr) { var c = arr.filter(x => palindrome(x));return c; }(arr);
```

Return median of two sorted arrays of same size

```
var a = [1, 4, 5,7];
var b = [4, 7, 9,8];
var c = a.concat(b);
c.sort((x,y) => x-y );
function isEven(n)
{
  if(n%2 === 0)
    return true;
  else
     return false;
}
var median = function(c) { if(isEven) { var h = (c[c.length/2] + c[c.length/2 + 1])/2;} else { h =
c[Math.ceil(c.length/2)] } return h;}
median(c);

var median = function(c) { if(isEven) { var h = (c[c.length/2] + c[c.length/2 + 1])/2;} else { h =
c[Math.ceil(c.length/2)] } return h;}(c);
```

Remove duplicates from an array

```
var dup = function(arr) { var c = []; for(var i = 0; i<arr.length; i++) { for (varj = 0; j<arr.length; j++) {
if((arr[i] == arr[j]) && (i != j)) { arr.splice(j, 1); } } } return arr; }
dup(arr);
```

```
var dup = function(arr) { var c = []; for(var i = 0; i<arr.length; i++) { for (varj = 0; j<arr.length; j++) {
if((arr[i] == arr[j]) && (i != j)) { arr.splice(j, 1); } } } return arr; }(arr);
```

Rotate the array k times and return the array

```
var rotate = function( ar , k ){
while( k-- )
{
   var b = ar.shift();
   ar.push( b );
 }
return ar;
}
rotate(ar, k);

var rotate = function( ar , k ){
while( k-- )
{
   var b = ar.shift();
   ar.push( b );
}
return ar;
}(ar, k);
```