Problem 0 : Part A (15 mins):

Playing with JSON object’s Values:

Fluffy sorry, Fluffyy is my fav cat and it has 2 catFriends  
Write a code to get the below details of Fluffyy so that  
I can take him to vet.

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**

1. Add height and weight to Fluffy

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  
 }  
 ]  
}

cat.height="50cm"+;

cat.weight="5kg";

console.log(cat);

1. Fluffy name is spelled wrongly. Update it to Fluffyy

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  
 }  
 ]  
}

cat.name="Fluffyy";

console.log(cat);

1. List all the activities of Fluffyy’s catFriends.

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  
 }  
 ]

}

var bat=cat.catFriends.map(function ddd(a){

return a.activities });

console.log(bat);

1. Print the catFriends names.

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  
 }  
 ]

}

cat.filter(function foo(a){

return a.name!="Fluffy";

}).map(function poo(a){

return a.activities;

});

var bat=cat.catFriends.map(function ddd(a){

return a.name });

console.log(bat);

1. Print the total weight of catFriends

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  
 }  
 ]  
}

var bat=cat.catFriends.map(function ddd(a){

return a.weight });

console.log(bat[0]+bat[1]);

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

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  
 }  
 ]  
}

var bat=cat.catFriends.map(function ddd(a){

return a.activities });

console.log(cat.activities,bat);

1. Add 2 more activities to bar & foo cats

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  
 }  
 ]  
}

cat.catFriends[0].activities = ["be grumpy", "eat bread omblet","eat","sleep"];

cat.catFriends[1].activities = ["be grumpy", "eat bread omblet","jump","sleep"];

console.log(cat.catFriends[0].activities,cat.catFriends[1].activities);

1. Update the fur color of bar

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  
 }  
 ]  
}

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

console.log(cat);

Problem 0 : Part B (15 mins):

Iterating with JSON object’s Values

Above is some information about my car. As you can see, I am not the best driver.  
I have caused a few accidents.  
Please update this driving record so that I can feel better about my driving skills.

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.

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  
 }  
 ]  
}

for(i=0;i<myCar.accidents.length;i++)

{

myCar.accidents[i].atFaultForAccident= false;

}

console.log(myCar);

1. Print the dated of my accidents

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  
 }  
 ]  
}

for(i=0;i<myCar.accidents.length;i++)

{

console.log(myCar.accidents[i].date);

}

**Real challenges starts here**

:bowtie:

**Problem 1 (5 mins):**

**Parsing an JSON object’s Values:**

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

Input (Object):

var object = {name: “RajiniKanth”, age: 33, hasPets : false};  
Output:

[“RajiniKanth”, 33, false]

**Sample Function proto:**

var obj = {name : “RajiniKanth”, age : 33, hasPets : false};function printAllValues(obj) {  
 // your code here

console.log(Object.values(obj));  
}

Problem 2(5 mins) :

Parsing an JSON object’s Keys:

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

Example Input:  
{name : ‘RajiniKanth’, age : 25, hasPets : true}  
Example Output:  
[‘name’, ‘age’, ‘hasPets’]

**Sample Function proto:**

function printAllKeys(obj) {  
 // your code here

console.log(Object.keys(obj));  
}

Problem 3( 7–9 mins):

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.  
Input (Object):  
var object = {name: “ISRO”, age: 35, role: “Scientist”};  
Output:  
[[“name”, “ISRO”], [“age”, 35], [“role”, “Scientist”]]

**Sample Function proto:**

var obj = {name: “ISRO”, age: 35, role: “Scientist”};  
function convertListToObject(obj) {

console.log(Object.entries(obj));

// your code here  
}

Problem 4( 5 mins):

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:  
1) the first element of the array as the object’s key, and  
2) the last element of the array as that key’s value.  
Input (Array):  
var array = [“GUVI”, “I”, “am”, “Geek”];  
Output:  
var object = {  
GUVI : “Geek”  
}

**Sample Function proto:**

var arr = [“GUVI”, “I”, “am”, “a geek”];function transformFirstAndLast(arr) {  
 var arr1 = [[arr[0],arr[3]]];

const newobject = Object.fromEntries(arr1);

console.log(newobject);  
 return newObject;  
}

Problem 5 ( 7 -9 mins):

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.  
Input (Array):  
var array = [[“make”, “Ford”], [“model”, “Mustang”], [“year”, 1964]];  
Output:  
var object = {  
make : “Ford”  
model : “Mustang”,  
year : 1964  
}

**Sample Function proto:**

var arr = [[“make”, “Ford”], [“model”, “Mustang”], [“year”, 1964]];function fromListToObject(arr) {  
 var newObject = {};  
 newObject = Object.fromEntries(arr);

console.log(obj);  
 return newObject;  
}

Problem 6 (10 mins):

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.

Input (Array):  
var array = [[[“firstName”, “Vasanth”], [“lastName”, “Raja”], [“age”, 24], [“role”, “JSWizard”]], [[“firstName”, “Sri”], [“lastName”, “Devi”], [“age”, 28], [“role”, “Coder”]]];  
Output:  
[  
{firstName: “Vasanth”, lastName: “Raja”, age: 24, role: “JSWizard”},  
{firstName: “Sri”, lastName: “Devi”, age: 28, role: “Coder”}  
]

**Sample Function proto:**

var arr= [[[“firstName”, “Vasanth”], [“lastName”, “Raja”], [“age”, 24], [“role”, “JSWizard”]], [[“firstName”, “Sri”], [“lastName”, “Devi”], [“age”, 28], [“role”, “Coder”]]];

function transformEmployeeData(arr) {

const obj = Object.fromEntries(arr[0]);

const obj1 = Object.fromEntries(arr[1]);

var tranformEmployeeList =[obj,obj1];

console.log(arr1);  
 //Your code  
   
 return tranformEmployeeList;  
}

Problem 7 (10 — 20 mins):

Parsing two JSON objects and Compare:

Read this : <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify>

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.  
Success Case:  
Input:  
var expected = {foo: 5, bar: 6};  
var actual = {foo: 5, bar: 6}  
assertObjectsEqual(actual, expected, ‘detects that two objects are equal’);  
Output:  
Passed  
Failure Case:  
Input:var expected = {foo: 6, bar: 5};  
var actual = {foo: 5, bar: 6}  
assertObjectsEqual(actual, expected, ‘detects that two objects are equal’);  
Output:  
FAILED [my test] Expected {“foo”:6,”bar”:5}, but got {“foo”:5,”bar”:6}

var expected = {foo: 5, bar: 6};  
var actual = {foo: 5, bar: 6}

function assertObjectsEqual(actual, expected, testName){

const obj1 = JSON.stringify(actual);

const obje1 = JSON.stringify(expected);

if(obj1==obje1)

console.log("passed");

else

{

console.log("FAILED [my test] Expected "+obj1+"but got"+obje1);

}  
   
}

Problem 8(10 mins):

Parsing JSON objects and Compare:

I have a mock data of security Questions and Answers. You function should take the object and a pair of strings and should return if the quest is present and if its valid answer

var securityQuestions = [  
 {  
 question: “What was your first pet’s name?”,  
 expectedAnswer: “FlufferNutter”  
 },  
 {  
 question: “What was the model year of your first car?”,  
 expectedAnswer: “1985”  
 },  
 {  
 question: “What city were you born in?”,  
 expectedAnswer: “NYC”  
 }  
]

function chksecurityQuestions(securityQuestions,question) {  
  
 var bat=securityQuestions.filter(function ddd(a){

return a.expectedAnswer===answers;}).map(function sss(a){

return true;

});

return(bat);  
}

//Test case1:var ques = “What was your first pet’s name?”;  
var ans = “FlufferNutter”;

var status = chksecurityQuestions(securityQuestions, ques, ans);

console.log(status); // true

//Test case2:var ques = “What was your first pet’s name?”;  
var ans = “DufferNutter”;

var status = chksecurityQuestions(securityQuestions, ques, ans);

console.log(status); // flase

Problem 9(20 mins):

Parsing JSON objects and Compare:

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)  
{

var res = arr.filter(function foo(a){

return a.age<20;

}).map(function foo(a){

return a.name;

});

Return res

}

console.log(returnMinors(students));