**NAME:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ DATE:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

***Note:*** This exercise tests the understanding of following topics in ES6:

1. Arrow functions
2. Classes
3. Modules
4. Promises

# Arrow functions

* 1. The array below contains an array of objects, each represent a rectangle. Using Array’s ‘forEach()’ method, calculate the area of each rectangle and store it in a new array called ‘areas’. The area of a rectangle can be calculated as ‘rectangle.length \* rectangle.width’. Use arrow function syntax for the callback function to ‘forEach()’ method. Use object destructuring to unpack the ‘length’ and ‘width’ values from the rectangle object that gets passed as parameter to the callback function.

const rectangles = [

{ length: 10, width: 30 },

{ length: 20, width: 90 },

{ length: 54, width: 32 }

];

const areas = [];

* 1. The 'customers' variable holds an array of arrays, where each array represents a single customer. Convert this array of arrays into an array of objects, where each object has the keys 'name', 'phone', 'email' and 'city' and assign the result to 'customersAsObject'. Use array destructuring and the Array’s ‘map()’ method. Use arrow function syntax for the callback function to ‘map()’ method.

const customers = [

['Hari', '98860-98860', 'hari@abc.com', 'Bengaluru'],

['Ram', '98860-98861', 'ram@abc.com', 'Chennai'],

['Shiv', '98860-98862', 'shiv@abc.com', 'Mumbai'],

['Krish', '98860-98863', 'krish@abc.com', 'Delhi']

];

const customersAsObject = [];

# Classes

* Create a ‘Person’ class
  + Include a ‘name’ property
  + Include a constructor method.
    - Let the constructor take ‘name’ as input parameter
    - Initialize the ‘name’ property using input parameter
  + Include a ‘showDetails()’ method
    - Display the value of ‘name’ property within the method
* Create an ‘Employee’ class
  + Include a ‘department’ property
  + Include a constructor method
    - Let the constructor taken ‘name’ and ‘department’ as input parameters
    - Initialize ‘name’ property through the super class constructor
    - Initialize the ‘department’ property using input parameter
  + Include a ‘showDetails()’ method
    - Display the value of ‘name’ and ‘department’ properties within the method
* Create an instance of ‘Person’ class. Invoke ‘showDetails()’ method
* Create an instance of ‘Employee’ class. Invoke ‘showDetails()’ method
* Run the code and check if things are working as expected

# Modules

* Create a JavaScript module for Books. Name it as ‘books.js’
  + Create an array for storing ‘Book’ instances. Name it as ‘books’. Do not make this array visible to other files
  + Export a ‘Book’ class
    - Include ‘title’ and ‘author’ properties in the class
    - Include a constructor method that takes ‘title’ and ‘author’ input parameters
    - Initialize the properties using input parameters
  + Export ‘addBook()’ function
    - Let this function take an instance of ‘Book’ class as input parameter
    - Include code to add the book instance to ‘books’ array
  + Export ‘showBooks()’ function
    - Include code to iterate through ‘books’ array and display individual book details as a list
* Create a JavaScript module for Members. Name it as ‘members.js’
  + Create an array for storing ‘Member’ instances. Name it as ‘members’. Do not make this array visible to other files
  + Export a ‘Member’ class
    - Include ‘name’ and ‘email’ properties in the class
    - Include a constructor method that takes ‘name’ and ‘email’ input parameters
    - Initialize the properties using input parameters
  + Export ‘addMember()’ function
    - Let this function take an instance of ‘Member’ class as input parameter
    - Include code to add the member instance to ‘members’ array
  + Export ‘showBooks()’ function
    - Include code to iterate through ‘members’ array and display individual member details as a list
* Create a JavaScript file to consume ‘books’ and ‘members’ modules created above, for e.g., name it as ‘index.js’
  + Import ‘Book’, ‘addBook’ and ‘showBooks’ from ‘books.js’ module
  + Import ‘Member’, ‘addMember’ and ‘showMembers’ from ‘members.js’ module
  + Create couple of instances of ‘Book’ class
  + Call ‘addBook()’ function to add the instances to book catalog
  + Create couple of instances of ‘Member’ class
  + Call ‘addMember()’ function to add the instances to member list
  + Call ‘showBooks()’ function to display the book catalog
  + Call ‘showMembers()’ function to display the member list
* Run the code and check if things are working as expected

# Promises

Refactor the following code.

* ‘getUser()’ function currently uses a callback to communicate with the caller. Modify the function to return a promise object
* The caller should use ‘then()’ and ‘catch()’ methods of the promise object to handle success and error scenarios
* Use arrow function syntax for all function definitions and callbacks

function getUser(id, callback) {

console.log('getUser() begin');

if (isNaN(id)) {

callback('User Id should be a number');

return;

}

setTimeout(function () {

callback(null, {

id: id,

name: 'Hari'

});

console.log('getUser() end');

}, 3000);

}

console.log('Begin');

getUser(1, function (error, user) {

if (error) {

console.log('Error:', error);

return;

}

console.log('user:', user);

});

console.log('End');