Web Developer Test

# General Knowledge:

1. Describe a RESTful service.
2. Describe a SPA application.

# Vue.js

1. What are the differences between two-way data binding and one-way data binding?
2. What is Vuex? What problems does it try to solve?

# JavaScript

1. Write a function to remove duplicate items from an array of strings.
2. What is the output of the following, and explain why?

*let* object1 = { elem: 1 };  
*let* object2 = { elem: 1 };  
*let* object3 = object1;  
*console*.log(object1 === object2);  
*console*.log(object1 === object3);

1. Given the following

*function* wait(label) {  
 *return new Promise*(resolve => {  
 *setTimeout*(resolve(label), 1000);  
 });  
}  
*async function* test(){  
 *const* list = [wait('First'), wait('Second')];  
 *console*.log('Started function');  
 list.forEach(*async function*(x){  
 *console*.log('Started');  
 *console*.log(*await* x);  
 *console*.log('Finished');  
 });  
 *console*.log('Finished function');  
}

We want the output to look like

1. Started function
2. Started
3. First
4. Finished
5. Started
6. Second
7. Finished
8. Finished Function

What is the output and how would you fix the function to operate in the assumed manner?

# Performance

1. What is the Big O notation of the following functions?

*function* Problem1(num = 0){  
 *return* 0;  
}  
*function* Problem2(array = []) {  
 *for*(*let* i =0;i<array.length;i++){  
 }  
}  
*function* Problem3(array = []) {  
 *for*(*let* i =0;i<array.length;i++){  
 *for*(*let* k=0;k<array.length;k++){  
 }  
 }  
}  
  
*function* Problem4(object ={}, haskey="") {  
 *if*(haskey *in* object){  
 *return true*;  
 }  
 *return false*;  
}  
*function* Problem5(object ={}, haskey="") {  
 *return Object*.keys(object).includes(haskey)  
}

1. Which Method is more performant? Explain your reasoning.

*function* Method1(arr = [],search =""){  
 *return* arr.find(x=>x === search) !== *null*}  
*function* Method2(obj = {},search =""){  
 *return Object*.keys(obj).find(x=> x === search) !== *null*}  
*function* Method3(obj = {},search =""){  
 *return* (search *in* obj);  
}  
*function* Method4(arr = [],search =""){  
 *return* arr.includes(search)  
}

# SQL

You have three tables

* Card (card\_id, name)
* Deck (deck\_id,card\_id)
* Game (game\_id,deck\_id)

Write SQL queries to answer the following questions. You may use any SQL dialect you wish.

Please specify which dialect you are using i.e. (MySQL, sqlite3, PostgreSQL, etc.).

1. Insert a Card
2. Update a Card Name
3. Delete a Game
4. Find the most used Card in all Games.

# Programming

Please implement the following. This will be judged based on **Readability** and **Functionality**.

1. You have an object of account's given in the format:

*const* accountData = {  
 '33fb0cb9-b24e-4f11-b16c-ed9055a78459':{  
 'createdAt':1005,  
 'name':'Player1'  
 },  
 'e1e4624e-dd90-4009-9fc6-9cad750b8fda':{  
 'createdAt':79,  
 'name':'Player2'  
 },  
 '6d0153b1-9522-4a5b-b793-e6185b5e4dbe':{  
 'createdAt':'5000',  
 'name':'Player3'  
 },  
 '3d914cd1-5163-4678-b081-053f6a7fbd64':{  
 'createdAt':92,  
 'name':'Player4'  
 },  
}

1. Write a function to return the name of the biggest createdAt value.
2. Write a function to return the name of the smallest createdAt value.
3. Write a function to return an array of sorted accounts by createdAt (should be the smallest first)
4. Q. Simple Card Game.
5. Create a Deck
   * The deck should contain 52 unique Cards.
   * Cards are standard poker cards.
   * Suit Order
     + Hearts, Diamonds, Clubs then Spades
   * Card Values
     + Aces are 15 points.
     + Face cards (Jack, Queen, King) are 10 points.
     + Numbered cards are worth the value on the card (i.e. a 5 is 5 points).
   * The Deck should be able to be shuffled
6. Create a Player
   * The Player should have a Hand that can hold 5 Cards from the Deck.
   * The Player's Hand should be sorted by suit and then by point value
   * The Player's Hand should be able to be totaled by point value.
   * Players should be able to draw a card from the top of the deck.
   * Players should be able to print the cards in their hand to console
7. Program Flow
   * Initialize and Shuffle the Deck.
   * Create a random number of Player's between 2-6.
   * Allow each player to draw a Card from the Deck until all players have 5 cards.
   * Count the total points in each player's hand and print all players hands to console.
   * Print the Player with the most Point's (if there is a tie randomly pick one of the winning players).

# Quiz video game

## Description

You have to make a quiz application

## Requirements

### Front End Requirements

* The player can see one question at the time
* The player can see their score at the end
* The order of the questions and answer choices should be randomized
* The game should be a single-page application
* You can use Bootstrap, Materialize, or another framework as desired

### Back End Requirements

* Create a nodejs backend server to serve data to the front end
* You cannot edit the quizQuestions.txt file, instead create your own data structure to use.
* The client should not know what the correct answers are.
* The server should handle question verification, the client should send a potential answer to the server for the server to verify if that answer was correct.
* The server should respond with whether the selected answer was correct or not and the client should react based on the response.

### Example Quiz Layout

