



## Full Stack Developer - Take home exercise

### Question 1 - Build the following

#### Part one

The final aim of this exercise is to list containers of different users in the browser; **displayed one after the other** and where the **admin keeps scrolling** to check each. This is how the final prototype should look like:

Sort By:

Filter by:

150x150

**Kaufman Frank**  
59 years old  
Male | Employee  
  
biography:  
*Aliquip labore do aute magna in magna aute cupidatat ad ad ipsum ipsum anim*  
  
547 Croton Loop, Harold, Palau, 2653  
+1 (932) 408-3778

Balance

\$ 2,995.90

laboris | voluptate |  
pariatur

150x150

Same as above;  
  
the next user's info will be displayed in here...

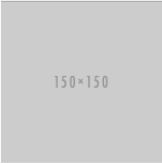
Balance

\$ etc..

..etc..

..More container etc...

- You are provided with two `.json` files: `data.json` and `mock_db_data.json`.
- You are required to use **AngularJS (v1)** on the front and **Node.js (with express framework)** on the back end.
- **No** database is required. You will be using the `mock_db_data.json` to simulate getting data from a NoSQL database.
- Each user container represents a **re-usable component (ie custom directive)**. And will look like the following (next page):



## Kaufman Frank

59 years old

Male | Employee

biography:  
*Aliquip labore do aute magna in magna aute cupidatat ad ad ipsum ipsum anim*

547 Croton Loop, Harold, Palau, 2653  
+1 (932) 408-3778

Balance

\$ 2,995.90

laboris | voluptate |  
pariatur

You fill a container based on json data received, like the following:

```
{
  _id: "57d46d526480ebf541389589",
  balance_cad: "2,955.90",
  picture: "http://placeholder.it/150x150",
  name: "Kaufman Frank",
  gender: "male",
  phone: "+1 (932) 408-3778",
  address: "547 Croton Loop, Harold, Palau, 2653",
  bio: "Aliquip labore do aute magna in magna aute cupidatat ad ad ipsum ipsum anim.",
  date_of_birth: "1957-02-02",
  tags: [
    "laboris",
    "voluptate",
    "pariatur"
  ],
  category: "employee"
},
```

- Your application on load should get data using a rest call to you server (the server will use the **data.json** file). You will then display the containers using this retrieved data.
- You **must** style (css) your prototype. No need for extra fancy styling, keep it simple however it should be made from **scratch** (Bootstrap, Foundation or others are not allowed). You can use flexbox for aligning, if you are familiar with it.

## Part two

Add two dropdowns: (check image above)

- One to sort containers by either 'none' or 'name'
  - \* Should have 2 items: 'none' and 'name'
  - \* When 'none' selected no sorting happens
  - \* 'none' is loaded by default on page load
- One to filter items out based on their respective 'category': *employee, investor, founder, board\_member*
  - \* Should have 5 items: 'none', 'employee', 'investor', 'founder', 'board\_member' (**remember; these are gotten from the json data response that your app receives**)
  - \* When 'none' selected no filtering happens
  - \* 'none' is loaded by default on page load

**Note:** Make the bar that contains the dropdowns fixed (ie it will always appear while containers are being scrolled)

## Part three

The admin should be able to click on the image. On click, your application should send a server request appending the container's '\_id' to the url. Something like the following:

**/yourapi/ 57d46d52f728f5230c46c676**

'57d46d52f728f5230c46c676' is the '\_id' of the item

- Your server should get the data from **mock\_db\_data.json** for this task (it has 1 json object, this same object will be used for all containers for simplicity). You can mock this json data into a javascript variable for simplicity.
- Append the response data to its specific container on the UI when received.

***Please note it is up to you to consider all optimizations, structuring, architecture and which Angular providers to use for this prototype; then decide the best approaches accordingly. These will be used for assessment.***

***Please note the app should be built from scratch. Though you are welcome to use existing angular directives in few very specific cases to optimize the performance of the app.***

***Please note that no unit testing is required for this prototype.***

## Question 2 (Bonus)

**How would you refactor the following angular code?**

```
angular.module(app, [])
.controller('appCtrl', function ($http, $q) {
  function parseFeed(id) {
    var deferred = $q.defer();
    $http.get('/json/' + id)
    .success(function (balance) {
      $http.get('/json/' + id)
      .success(function (range) {
        var response = { balance: balance, range: range };
        deferred.resolve(response);
      });
    });
    return deferred.promise;
  }
  parseFeed(3);
});
```