ASSIGNMENT 2

Question # 4

**React**

React is a library that deals with views.

**AngularJS**

AngulaJS is a full-fledged framework.

ADVANTAGES

**REACT**

* Simplifies your concerns. You are directly mapping your data into DOM elements
* No need to track what fields/data is being changed. All changes are treated the same.
* Performance is less affected by how much data is needed to generate the dom.

**ANGULAR**

Performance is less linked to the size of your view (processing through more nodes is less laborious then managing a virtual representation of the view and performing diffs).

DISADAVANTAGES

**REACT**

* Less resources and experienced people using React (simply because it's newer).
* Doing diffs and recomputing the DOM can be less efficient then dirty checking.
* Doing diffs and recomputing the DOM can be less efficient then dirty checking.

**ANGULARJS**

* Lots of domain specific language use which is hard to perform static analysis on.
* Need to track how all data changes will propagate through your application.

Question # 5

**VUE.JS**

Vue.js is an open-source JavaScript framework for building user interface and single-page applications.

Vue.js features an incrementally adoptable architecture that focuses on declarative rendering and component composition. Advanced features required for complex applications such as routing, state management and build tooling are offered via officially maintained supporting libraries and packages

**COMPARISON WITH REACT AND ANGULARJS**

* React and Vue both excel at handling dumb components: small, stateless functions that receive an input and return elements as output.
* React focuses on the use of JavaScript ES6. Vue uses Javascript ES5 or ES6. Angular relies on **TypeScript.**
* Angular is a framework rather than a library because it provides strong opinions as to how your application should be structured. React and Vue, on the other hand, are universally flexible. Their libraries can be paired to all kinds of packages

Question # 6

DIFFERENT  
Angular **is** based on TypeScript while AngularJS **is** based on JavaScript. TypeScript is a superset of ES6 and it’s backward compatible with ES5. Angular has also benefits of ES6 like: lambda operators, iterators or reflection’s mechanism.

**SCOPE AND CONTROLLERS**

AngularJS uses terms of scope and controller. To scope a variable you can add many variables that will be visible in View as well as in Controller. AngularJS has also a concept of rootScope. Variables in rootScope are available on all throughout application. Angular does not have a concept of scope or controllers.

**TEMPLATE ENGINE**

AngularJS has many directives and every developer can also specify custom new directive. Angular also has standard directives, but they are used in a bit different way. For example: ng-model in AngularJS means that you want to create two-way binding. If you want to create one-way binding, you should use ng-bind. Angular occurs only ngModel, but if you would write it only in: “[ ]”,you’ll get one-way binding. If you want to create two-way binding you must write it in: “[( )]”. We have to write it this way because of the fact that “[ ]” is used to property binding and “( )” is used to event binding.

Question # 7

**LINTING**

Linting is the process of running a program that will analyze code for potential errors.

**JSLINT**

JSLint is a static code analysis tool used in software development for checking if JavaScript source code complies with coding rules. It is provided primarily as a web application through jslint.com, but there are also command-line adaptations

**ESLINT**  
 The primary reason ESLint was created was to allow developers to create their own linting rules. ESLint is designed to have all rules completely pluggable. The default rules are written just like any plugin rules would be. They can all follow the same pattern, both for the rules themselves as well as tests. While ESLint will ship with some built-in rules to make it useful from the start, you’ll be able to dynamically load rules at any point in time.

Question #1

**MongoDB vs Mongoose**

 MongoDB is a NoSQL database system which stores data in the form of BSON documents. On the other hand, Mongoose is an **Object modeling tool** for MongoDB.

Data in MongoDB has a flexible schema documents in the same collection. They do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data

**CRUD**

CRUD operations refer to the basic Insert, Read, Update and Delete operations.

These terms are the foundation for all interactions with the database.

**CREATE OPERATIONS**

Create or insert operations add new [documents](https://docs.mongodb.com/manual/core/document/#bson-document-format) to a [collection](https://docs.mongodb.com/manual/core/databases-and-collections/#collections). If the collection does not currently exist, insert operations will create the collection.

MongoDB provides the following methods to insert documents into a collection:

db.collecton.insertOne()

db.collection.insertMany()



**READ OPERATIONS**

Read operations retrieves [documents](https://docs.mongodb.com/manual/core/document/#bson-document-format) from a [collection](https://docs.mongodb.com/manual/core/databases-and-collections/#collections); i.e. queries a collection for documents. MongoDB provides the following methods to read documents from a collection:

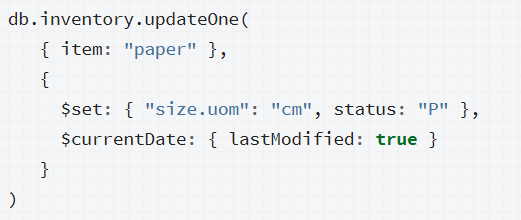
[db.collection.find()](https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find)



**UPDATE OPERATIONS**

Update operations modify existing [documents](https://docs.mongodb.com/manual/core/document/#bson-document-format) in a [collection](https://docs.mongodb.com/manual/core/databases-and-collections/#collections). MongoDB provides the following methods to update documents of a collection:

[db.collection.updateOne()](https://docs.mongodb.com/manual/reference/method/db.collection.updateOne/#db.collection.updateOne)



**DELETE OPERATIONS**

Delete operations remove documents from a collection. MongoDB provides the following methods to delete documents of a collection:

[db.collection.deleteOne()](https://docs.mongodb.com/manual/reference/method/db.collection.deleteOne/#db.collection.deleteOne)

[db.collection.deleteMany()](https://docs.mongodb.com/manual/reference/method/db.collection.deleteMany/#db.collection.deleteMany)



Question #2

**POST vs PUT**

The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. In other words, POST is used to create.

The PUT method requests that the enclosed entity be stored under the supplied Request-URI. If the Request-URI refers to an already existing resource, the enclosed entity SHOULD be considered as a modified version of the one residing on the origin server. If the Request-URI does not point to an existing resource, and that URI is capable of being defined as a new resource by the requesting user agent, the origin server can create the resource with that URI.

**DIFFERENCE:**

The fundamental difference between the POSTand PUT requests is reflected in the different meaning of the Request-URI. The URI in a POST request identifies the resource that will handle the enclosed entity… In contrast, the URI in a PUT request identifies the entity enclosed with the request.

**USING POST**

There are cases where you need to use AJAX to accomplish tedious tasks such as database lookups and updates. AJAX may not come to mind when you think about those tasks, but it should be used especially when post backs are very expensive functions in your application.

**Replacing POST with PUT**

POST is NOT idempotent. So if you retry the request N times, you will end up having N resources with N different URIs created on server.

PUT method is [idempotent](https://restfulapi.net/idempotent-rest-apis/) . So if you send retry a request multiple times, that should be equivalent to single request modification.

Question #3

In a PUT request, the enclosed entity is considered to be a modified version of the resource stored on the origin server, and the client is requesting that the stored version be replaced.

With PATCH, however, the enclosed entity contains a set of instructions describing how a resource currently residing on the origin server should be modified to produce a new version.

Also, another difference is that when you want to update a resource with PUT request, you have to send the full payload as the request whereas with PATCH, you only send the parameters which you want to update.

Let's look at one of your examples.

{ "username": "skwee357", "email": "skwee357@domain.com" }

If you POST this document to /users, as you suggest, then you might get back an entity such as

## /users/1

{

"username": "skwee357",

"email": "skwee357@domain.com"

}

If you want to modify this entity later, you choose between PUT and PATCH. A PUT might look like this:

PUT /users/1

{

"username": "skwee357",

"email": "skwee357@gmail.com" // new email address

}

You can accomplish the same using PATCH. That might look like this:

PATCH /users/1

{

"email": "skwee357@gmail.com" // new email address

}

You'll notice a difference right away between these two. The PUT included all of the parameters on this user, but PATCH only included the one that was being modified (email).

When using PUT, it is assumed that you are sending the complete entity, and that complete entity replaces any existing entity at that URI. In the above example, the PUT and PATCH accomplish the same goal: they both change this user's email address. But PUT handles it by replacing the entire entity, while PATCH only updates the fields that were supplied, leaving the others alone.

Submitted By

Roshan Ul Haq

16L-4143

Web Programming (A)