**ASSIGNMENT#2**

**ROLL#:16l-4084**

**MUNEEB AKHTAR**

**Question # 1**

**Mongodb vs Mongoose**

**1:** Mongodb is the **database itself**, while Mongoose is **an object modeling tool** for Mongodb.

**2: Mongoose** Object data modeling (ODM) library that provides a rigorous modeling environment for your data. Used to interact with MongoDB, it makes life easier by providing convenience in managing data while **Mongodb is**  native driver in Node.js to interact with MongoDB.

**3:** **Mongoose** is open-sourced with the MIT license and is also maintained by **MongoDB**, Inc.

**4:** 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:**

**MongoDB** provides rich semantics for reading and manipulating data. **CRUD** stands for create, read,update, and delete. These terms are the foundation for all interactions with the database.

**1: 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.

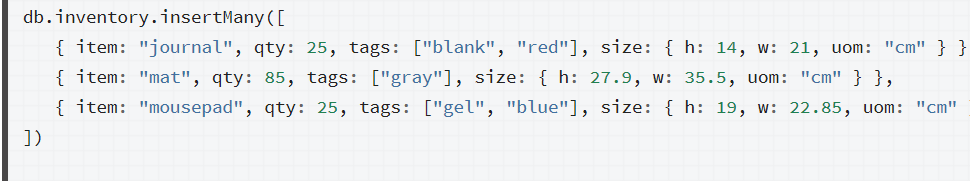
MongoDB provides the following methods to insert documents into a collection:

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

EXAMPLE:



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



**2: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:

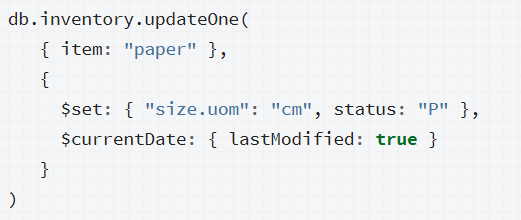
[db.collection.find()](https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find)



**3: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:

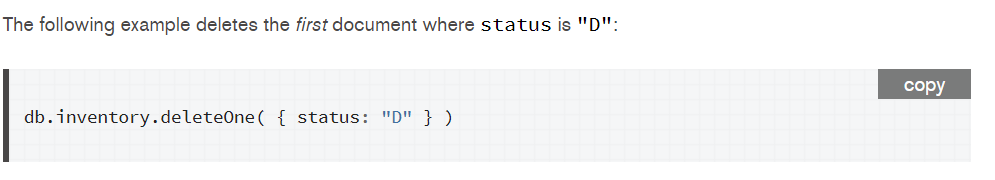
[db.collection.updateOne()](https://docs.mongodb.com/manual/reference/method/db.collection.updateOne/#db.collection.updateOne)



**4: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)



**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 **POST and 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 postbacks 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**

**PUT vs PATCH**

**PATCH** can be used when the client is sending one or more changes to be applied by the server. The **PATCH** method requests that a set of changes described **in the** request entity be applied to the resource identified by the Request-URI. The set of changes is represented **in a** format called a **patch** document.

The difference between the **PUT** and **PATCH** requests is reflected in the way the server processes the enclosed entity to modify the resource identified by the Request-URI.

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.

**Conclusion:**

Since PUT requests include the entire entity, if you issue the same request repeatedly, it should always have the same outcome (the data you sent is now the entire data of the entity). Therefore PUT is idempotent.

**QUESTION#4**

**React** (also known as **React**.**js** or **ReactJS**) is an open-source **JavaScript** library for rendering views. While full frameworks like **AngularJS** provide the model, view, and controller, **React** is a library that caters only to the view.

Throughout the years, AngularJs and ReactJs-the two prominent structure of JavaScript have been in discussion, there's no obvious answer at whatever point it comes to picking the best between two extremely well known and predictable structures. However, there are few points of interest of each and have some key contrasts too.

1. **Data Binding**

Angular allows two-way data binding while React allows one-way data binding. Two-way data binding means that any changes you make to the model affect the view, and vice versa. One-way data binding means any changes you make to the model affect the view, but not the other way around. Two way data binding is the primary key difference, it disperses after every set of data changes.

### **DOM Usage**

DOM is the Data Object Model of a web app. **Angular** uses the browser's DOM, while **React** uses a virtual DOM which help developers manage an extensive database. By using a virtual DOM, you can change any element very quickly and without needing to render the whole DOM. It drastically changes the performance from good to excellent.

### **Learning Curve**

This will differ from individual to individual based on skill and experience. On average, TypeScript is considered harder to learn than JSX, in turn increasing the learning curve with Angular as compared to React.

1. **Overall performance**

Respond has an incite rendering highlight that gives it a slight edge over the Rakish JavaScript. It comprises of different ways to deal with reduce the measure of DOM activity and in this manner accelerates the refreshing procedure, making it progressively proficient.

As indicated by previously mentioned elements both Rakish and Respond works totally on different methodologies. Both the innovations are predominant; it altogether relies upon framework prerequisites that will help settle on an official choice. On the off chance that in the event that you are building dynamic applications that utilizes virtual DOM, single page app(display all progressions made to the substance without reloading the present page.), and local application, Respond is the perfect decision and in the event that you are building cross-stage portable apps(addresses constraining components, for example, route by means of touch, diverse screen sizes, and versatile equipment.), venture software(Angular utilizes a MVC design, it is incredible for making undertaking level apps.)and dynamic web applications or half breed versatile applications, Rakish is the best of all. More or less, both the systems give a fiery arrangement of apparatuses for adaptable, quality and receptive electronic applications. 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 assets and experienced individuals utilizing Respond (essentially in light of the fact that it's more up to date).

• Doing diffs and recomputing the DOM can be less productive at that point messy checking.

• Doing diffs and recomputing the DOM can be less proficient at that point grimy 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

Vue is also perfectly capable of powering sophisticated Single-Page Applications when used in combination with [**modern tooling**](https://vuejs.org/v2/guide/single-file-components.html) and [**supporting libraries**](https://github.com/vuejs/awesome-vue#components--libraries).

1. **Flexibility**

A lot of adaptability is another preferred standpoint of Vue.js. It enables the client to compose his format in HTML record, JavaScript document, and unadulterated JavaScript record utilizing virtual hubs. This adaptability likewise makes it straightforward for the designers of React.js, Angular.js, and some other new JavaScript structure. Vue.js has demonstrated a ton helpful in the improvement of those straightforward applications that run specifically from programs.

1. **Size**

The achievement of JavaScript structure relies upon its size. The littler the size is, the more it will be utilized. One of the best points of interest of Vue.js is its little size. The extent of this system is 18– 21KB and it requires no investment for the client to download and utilize it. This does not imply that it has low speed in light of little size. Rather, it beats all the massive systems like React.js, Angular.js, and Ember.js.

1. **Learning bend**

One reason for the fame of this system is that it is very straightforward. The client can without much of a stretch add Vue.js to his web venture as a result of its basic structure. Both the little and in addition vast scales formats can be created through this structure which spares a considerable measure of time. If there should be an occurrence of any issue, the client can without much of a stretch follow the squares with blunders. This is a direct result of its basic structure.

**4- Directives versus Segments**

Vue has a clearer partition among mandates and segments. Mandates are intended to embody DOM controls just, while parts are independent units that have their very own view and information rationale. In AngularJS, mandates do everything and segments are only an explicit sort of order.

**QUESTION# 6**

First of all, **Angular** is based on TypeScript while **AngularJS** is based on JavaScript. TypeScript is a superset of ES6 and it's backward compatible with ES5.  **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.

**Angular JS** is an open source JavaScript framework that is used to build web applications. It can be freely used, changed and shared by anyone. **Angular**  is a component based model completely, where as **Angular JS** is not.

| 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. Instead of them it uses a hierarchy of components as its main architectural concept. Component is a directive with a template. That is a similar approach as in ReactJS &ndash; another library used for building user interfaces.

Let me point out some advantages here:

* **TypeScript:**Angular recommends TypeScript as a primary language for writing angular app code which is a superset of JavaScript and provides amazing features like auto-completion, static types, etc.
* **Native/Hybrid Mobile Application:**With the smooth integration of Ionic and NativeScript, you can build mobile or hybrid apps without dwelling into any of the native programming languages (iOS, Android, etc.).
* **Components:**The concepts of controllers has been merged inside the component functionality and hence, the new angular follows a tight component based architecture where components are everything (figuratively speaking).
* **Angular Routers:**Angular routers have been optimized and gives a good performance for navigation in the modern web browser.

**QUESTION# 7**

**Linting** is the process of checking the source code for Programmatic as well as Stylistic errors. This is most helpful in identifying some common and uncommon mistakes that are made during coding.

A Lint or a Linter is a program that supports linting (verifying code quality). They are available for most languages like JavaScript, CSS, HTML, Python, etc..

**JSLint** is a [static code analysis](https://en.wikipedia.org/wiki/Static_code_analysis) tool used in [software development](https://en.wikipedia.org/wiki/Software_development) for checking if [JavaScript](https://en.wikipedia.org/wiki/JavaScript) [source code](https://en.wikipedia.org/wiki/Source_code) complies with [coding rules](https://en.wikipedia.org/wiki/Programming_style). It is provided primarily as a [web application](https://en.wikipedia.org/wiki/Web_application) through jslint.com, but there are also command-line adaptations. It was created in 2002 by [Douglas Crockford](https://en.wikipedia.org/wiki/Douglas_Crockford).

**Advantages**

* Comes configured and ready to go (if you agree with the rules it enforces)

**Disadvantages**

* JSLint doesn’t have a configuration file, which can be problematic if you need to change the settings
* Limited number of configuration options, many rules cannot be disabled
* Can’t add custom rules
* Undocumented features
* Difficult to know which rule is causing which error

ESLint is the most recent out of the four. It was designed to be easily extensible, comes with a large number of custom rules, and it’s easy to install more in the form of plugins. It gives concise output, but includes the rule name by default so you always know which rules are causing the error messages.

Uses

ESLint is a tool for identifying and reporting on patterns found in JavaScript code, with the goal of making code more consistent and avoiding bugs. ESLint uses [Espree](https://github.com/eslint/espree) for JavaScript parsing, ESLint uses an AST to evaluate patterns in code and ESLint is completely pluggable, every single rule is a plugin and you can add more at runtime.

### Advantages

* Any rule can be toggled, and many rules have extra settings that can be tweaked
* Very extensible and has many plugins available
* Easy to understand output

• Includes numerous principles not accessible in different linters, making ESLint increasingly valuable for recognizing issues

• Best ES6 bolster, and furthermore the main apparatus to help JSX

• Supports custom correspondents

### Disadvantages

* Some configuration required
* Slow, but not a hindrance

## **My Recommendations**

My choice of these two is ESLint as it is extensible, flexible and easy to interpret and understand

**QUESTION#8**

**Increased Speed:**

The main purpose of Ajax is to improve the speed, performance and usability of a web application. A great example of Ajax is the movie rating feature on Netflix. The client rates a movies and their own rating for that motion picture will be spared to their database without trusting that the page will invigorate or reload. These motion picture appraisals are being spared to their database without posting the whole page back to the server. Facebook, Gmail, and Pinterest are instances of destinations that utilization a ton of AJAX.

The "Offbeat" part alludes to the way that when the JavaScript makes the AJAX call to the webserver, it keeps on working until the reaction – it doesn't "square" and stop while the information is being handled server-side.

**1:Creating a Menu**

Navigation menus are a staple of all websites, whether the site is a traditional, multi-page experience or a single-page site. Menus that respond to user input (like a touch or click) and include attractive animation effects are one of the ways that framework like AngularJS can be utilized -simply by combining the framework with a little HTML and CSS.

**2:Creating a SPA**

There are a number of advantages to creating a single page website. As opposed to isolate pages waiting be brought and stacked amid a guest's time on the webpage, a solitary page site can give a substantially more liquid experience. This is on the grounds that all the code for the webpage is recovered in advance or powerfully stacked as important to make an ordeal that feels more like a work area application than a conventional, multi-page site.