**Name: Hamza Awais Roll#L154144**

**Question 1: MongoDB vs Mongoose. Why are we using Mongoose package instead of MongoDB package? How would we perform CRUD operations using the MongoDB package? Give code examples for creating, retrieving, updating and deleting MongoDB documents.**

**MongoDB vs Mongoose. Why are we using Mongoose package instead of MongoDB package?**

1. Developers often choose Mongoose because of the following reasons:
2. Mongoose brings schema’s and models to Node.js / Mongodb. It’s a **Object modeling tool**.
3. Mongoose is higher level interface to Mongodb and actually uses mongodb.js,  the MongoDB driver.
4. Better readability.
5. much more intuitive to learn if you're a beginner.
6. Mongoose automatically performs additional validation, and has many more search functions.

**Downfalls of Mongoose:**

As it’s an additional layer in the app, the performance drops as compared t o the native driver.

Access time of Moongose is greater than MongoDB native driver.

**How would we perform CRUD operations using the MongoDB package?**

**Create-**

The *insertOne* and *insertMany* methods exists on the *Collection* class and is used to insert documents into MongoDB

*insertOne:*

**var** r = **yield** db.collection(**'inserts'**).insertOne({a:1});

assert.equal(1, r.insertedCount);

*insertMany*

const collection = db.collection('documents');

// Insert some documents

collection.insertMany([

{a : 1}, {a : 2}, {a : 3}

], function(err, result) {

assert.equal(err, null);

assert.equal(3, result.result.n);

assert.equal(3, result.ops.length);

console.log("Inserted 3 documents into the collection");

callback(result);

});

The **insert** command returns an object with the following fields:

* **result** Contains the result document from MongoDB
* **ops** Contains the documents inserted with added **\_id** fields

**-Read**

Or find queries in the MongoDb can be handled in 2 ways.

**- Find All Documents**

const collection = db.collection('documents');

// Find some documents

collection.find({}).toArray(function(err, docs) {

assert.equal(err, null);

console.log("Found the following records");

console.log(docs)

callback(docs);

});

-**Find Document with a filter**

const collection = db.collection('documents');

// Find some documents

collection.find({'a': 3}).toArray(function(err, docs) {

assert.equal(err, null);

console.log("Found the following records");

console.log(docs);

callback(docs);

});

**-Update**

The following operation updates a document in the documents collection.

const collection = db.collection('documents');

// Update document where a is 2, set b equal to 1

collection.updateOne({ a : 2 }

, { $set: { b : 1 } }, function(err, result) {

assert.equal(err, null);

assert.equal(1, result.result.n);

console.log("Updated the document with the field a equal to 2");

callback(result);

});

The method updates the first document where the field a is equal to 2 by adding a new field b to the document set to 1.

**-Remove**

Remove the document where the field **a** is equal to **3**.

const collection = db.collection('documents');

// Delete document where a is 3

collection.deleteOne({ a : 3 }, function(err, result) {

assert.equal(err, null);

assert.equal(1, result.result.n);

console.log("Removed the document with the field a equal to 3");

callback(result);

});

**Question 2: POST vs PUT. Why was POST used to update employee data in the Employees sample AJAX application? What happens if you replace POST with PUT? What difference does it make? (2)**

**POST VS PUT**

|  |  |
| --- | --- |
| **PUT** | **POST** |
| PUT is update or create with a user-supplied URL. | POST is create with a server generated URL. |
| PUT should be used when the client specifies the location of the page. | POST is used when the client sends the page to the the server, and the server then tells the client where it put it. |
| PUT is a much more limited operation. | POST has not limited operation. |
| The URI in a PUT request identifies the entity enclosed with the request. | The URI in a POST request identifies the resource that handles the enclosed entity. |
| PUT is supposed to accept the body of the request, and then store that at the resource identified by the URI. | POST is supposed to initiate an action on the server. It could be a different URI, or it could be a different action. |
| obj.attribute = value # A PUT request. | obj.set\_attribute(value) # A POST request. |

**Why was POST used to update employee data in the Employees sample AJAX application**

Post was used because clients send a request to the server with the URI as the resource enclosed in the request object.

**Question 3: PUT vs PATCH. Can PUT be used for partial updates e.g. in case of updating your name in an online university application form what method should be used? PUT or PATCH or anyone? Why?**

|  |  |
| --- | --- |
| **PUT** | **PATCH** |
| 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. | 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. |
| when you want to update a resource with PUT request, you have to send the full payload as the request | you only send the parameters which you want to update. |
| Full payload. | PATCH request requires less bandwidth. |

**Can PUT be used for partial updates e.g. in case of updating your name in an online university application form what method should be used?**

Yes, infact PUT should be preferred in such cases.

**Question 4. Where does React lie in comparison to AngularJS? Compare with respect to advantages and disadvantages.**

**Why choose AngularJS**

Angular is a JS framework for front-end development. It’s the second version of the product that is supported by Google. The first version known as AngularJS was accused of redundant complexity. So nearly all of its concepts (modules, controllers, scopes, directives, etc.) were replaced in the Angular 2, aka Angular. Angular received some new features, as well, such as native app support and server-side rendering.

Reasons for Angular popularity:

* It’s Google’s product which means a lot in terms of trust;
* Developers can adapt to it without any troubles;
* The code looks really simple;
* Apps developed with Angular are highly customizable and interactive;
* Angular ensures advanced testing features and Model-View-Controller balance.
* MVVM (Model-View-ViewModel) that allows developers to work separately on the same app section using the same set of data.
* Dependency injection of the features related to the components with modules and modularity in general.

**Why not to choose AngularJS**

* Websites powered by Angular (usually dynamic single-page ones) are not as SEO-friendly as expected.
* In addition, despite the overall great performance results, websites that have to display large amounts of data are rather slow.

**Why choose React?**

React or ReactJS is a JS-based library with a JSX compiler, which is focused on reusable UI views components in user interfaces. Be careful with terms: React is not an MVC framework, it’s an open-source library for views rendering. UIs created with React are highly responsive and load smoothly.

Reasons for React popularity:

* The library is strongly supported by Facebook;
* Solutions based on React are SEO-friendly and perform well;
* React is a flexible library.
* Downward data binding which means that with this kind of data flow the child elements cannot affect parent data.
* light-weighted because the data performing on the user side can be easily represented on the server side simultaneously.

**Why not to choose React**

* The view-oriented approach can be problematic.
* unopinionated — meaning that developers sometimes have too much choice;
* The use of React can require more code and manual processing of data changes.

**Detailed Comparison**

|  |  |  |
| --- | --- | --- |
|  | **AngularJS** | **React** |
| **License** | * Open source * Angular is licensed under the MIT license | * Open source * React uses 3-clause BSD |
| **MVC pattern** | Angular is based on the MVC architecture | React has just “V” (view) of MVC. |
| **Templating** | Angular uses templating in TypeScript files | React uses templating in JSX files. |
| **Data Binding** | Angular implements two-way data binding, which allows the framework to connect DOM to Model data via the Controller. | React implements one-way data binding. So the data is directed only one way. |
| **Rendering** | server-side rendering | server-side rendering |
| **Performance** | two-way data binding influences the performance when it comes to massive apps | ReactJS developers have introduced the virtual DOM that allows creating light DOM tree saving it on the server, which reduces the load on the browser. |

**Question 5:What is Vue.js? Where does Vue lie in comparison to React and AngularJS? Give examples.**

**Angular Vs React Vue comparison**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Angular Js** | **React** | **Vue Js** |
| **Complexity** | Is not as simpler as Vue in terms of design and API as vue is created after major frameworks release. | React makes it easy to build both complex and straightforward web and mobile applications, but it comes with a price**— framework complexity** and boilerplate. | Is very simpler in terms of design and API. |
| **Data binding** | Two way binding between scopes but we want to connect with async services, we need to have third-party developed components to integrate with the application. | Both one-way and two way data bindings a re possible in React. | One-way data glow bw the components and leads to develop non-trivial application in less time. |
| **Integration** | Is also easy to integrate but with third-party component and Js is popular. Many developers developed component which can be integrated into any angular.js application. | Little difficult than Vue when integrating with other third party frameworks. | Is easy to be integrated with other frontend libraries which makes it a versatile and easy to use the framework. If any feature is missing or any enhancement need we can easily integrate with the library. |
| **Performance** | Performs fast even though due to a lot of watchers as whenever scope changes, watchers need to revaluated again. But ir performs equally with Vue.js having similar metrics on different benchmarks. | Comparing DOM manipulations, React’s overall performance is great. It is much faster than Angular but is a bit slower than Vue | Good performances as it doesn’t have any wathcers as it uses a transparent dependency tracking system with async queues. Both Angular and Vue performs fast with very similar metrics on benchmarks and addressing similar issues. |
| **Flexibility** | More flexible, with a lot of support available. | React focuses on UI, so the essential thing you get is its support for building user-facing components.  What it doesn’t offer as part of the React official library is more advanced features like state management. | Not as flexible. Less opiniated as it has some rules to follow the project structure while developing applications but it is more flexible for modular solutions. |
| **Learning Curve** | Very steeper as API documentation is very huge and user needs to familiarize with some concepts before and its complexity is large due to its design. It is difficult for fewer experience people to pick up and do development of application. | large React projects tend to get complicated, and the fragmentation in the community does not help. The fact that React introduced many new paradigms has some negative effect on its learning curve. | Lesser steep as if the user knows HTML/CSS/JS then they can develop SAP and non trivial application development within less than a day. So VUE is easier. |
| **Typescript** | Essentially required as all learning sources and documentation is typescript based which offers static type checking for large scale application | Uses Babel instead of Typescript | Doesn’t have typescript but it has some official typings decorators and they are interacting with MS typescript team to support Vue Js. |

**Question 6: How is AngularIO different from AngularJS? Give examples.**

* AngularJS and Agular are two separate frameworks.
* Angular JS was very active from 2008-2011. But after that, the developers realized the modern apps are growing very rapidly in terms of complexity and sizes and they must do something to meet the pace.
* They decided and came up with a brand new framework, which was totally unrelated to Angular JS. They called it Angular.
* Angular JS was MVC based architecture, whereas Angular is Model based architecture, also known as MVVC.
* AngularJS used Javascript. Angular uses Typescript. You can develop app on Angular using JS, but it is not recommended.

**Question 7: What is Linting? What is the use of JSLint? What is the use of ESLint? Give examples.**

**Linting**: 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..

**JS Lint and use.**

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. It was created in 2002 by Douglas Crockford.

**When to use:**

* When you don’t want to install another linter. It comes pre-configured.
* For javascript validation
* Sometimes JSLint tells you things that are critical, sometimes it just nags. You can get lost trying to make it happy, but you might just *be* lost without it.

**ES Lint**

ESLint is completely plug-able, every single rule is a plugin and we can add more rules at run-time. It is the most recent out of the three libraries. It is designed to be easily extensible, comes with a large number of custom rules, and it’s easy to install more rules in the form of plugins. It gives concise output, and also includes the rule name by default so you always know which rules are causing the error messages.

**When to use:**

* When you want to create your own rules.
* When you want to configure your own rules.
* When you are following ES6 standards.
* For validation js

**Question 8 Give an example where you would prefer to use AngularJS over AJAX and vice versa.**

**When to use Angular**

* When you are dealing with SPA.
* When a component needs to be refreshed instead of complete page.
* When building big websites.
* When you have two way binded applications.

**When to use AJAX**

* When you want to get data from a server without the need to refresh a webpage.
* When you have to send request to server instead of client- rendering is done by server
* Multi page applications