WEB PROGRAMMING

RESEARCH ASSIGNMENT



NAME: Muhammad Ahmad Saif

SUBMITTED TO: Miss Shamsa Abid

ROLL NO: 16L-4054

**FAST-NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES**

**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.**

**Answer:**

[MongoDB](https://www.npmjs.com/package/mongodb) is the **native driver** for interacting with a MongoDB instance and [Mongoose](https://www.npmjs.com/package/mongoose) is an **Object modeling tool** for MongoDB. Using Mongoose, a user can define the schema for the documents in a particular collection. It provides a lot of convenience in the creation and management of data in MongoDB. However, if your collection schema is unpredictable, or you want a Mongo-shell like experience inside Node.js, then use the MongoDB driver.

Example of CRUD operations in MongoDB:

* **Create a new Product**

// routes/products.routes.js

...  
router.post('/create', product\_controller.product\_create);

// controllers/products.js

exports.product\_create = **function** (req, res) {  
 **let** product = **new** Product(  
 {  
 name: req.body.name,  
 price: req.body.price  
 }  
 );  
  
 product.save(**function** (err) {  
 **if** (err) {  
 **return** next(err);  
 }  
 res.send('Product Created successfully')  
 })  
};

* **Read an existing Product**

// routes/products.routes.js

...  
router.get('/:id', product\_controller.product\_details);

// controllers/products.controller.js

exports.product\_details = **function** (req, res) {  
 Product.findById(req.params.id, **function** (err, product) {  
 **if** (err) **return** next(err);

res.send(product);  
 })  
};

* **Update an existing Product**

// routes/products.routes.js

...  
router.put('/:id/update', product\_controller.product\_update);

// controllers/products.controller.js

...  
exports.product\_update = **function** (req, res) {  
 Product.findByIdAndUpdate(req.params.id, {$set: req.body}, **function** (err, product) {  
 **if** (err) **return** next(err);  
 res.send('Product udpated.');  
 });  
};

* **Delete an existing Product**

// routes/products.routes.js

...  
router.delete('/:id/delete', product\_controller.product\_delete);

// controllers/products.controller.js

exports.product\_delete = **function** (req, res) {  
 Product.findByIdAndRemove(req.params.id, **function** (err) {  
 **if** (err) **return** next(err);  
 res.send('Deleted successfully!');  
 })  
};

**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?**

**Answer:**

POST was used to update employee data in the Employees sample AJAX application because the Employee ID was a part of the URI and was automatically created by the server i.e. POST is used if the server is responsible for creating new resources .

If we use PUT instead of POST it will always produce the same result because it is idempotent.

**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?**

**Answer:**

PUT cannot be used for partial updates because it is idempotent and will not update the name in an online university application form. The HTTP method PATCH can be used to update partial resources, for instance when we only need to update one field of the resource.

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

**Answer:**

* **AngularJS** has a built-in architecture for an application's front end. When your application can fit within those confines, then Angular can be a joy to work with. However, if you find yourself deviating from Angular’s desired architecture, you can encounter a lot of problem.

**React**, in comparison, does not attempt to impose an architectural ideal on your code base. It allows you to do lazy loading of components, decreasing load times and allowing you to more closely manage how your data is represented.

* One benefit of **Angular’s** use of directives for data-driven display is that display templates are incredibly simple to write.

**React**, on the other hand, tends to require custom functions to drive data display.

* While **Angular’s** data representations can be highly compact, rendering the data can prove to be a problem for large data sets. Since two-way data binding requires a listener on every changeable element, a large amount of data can pose a significant performance problem.

**React**, on the other hand, makes use of a virtual DOM to track changes in elements. When a change is detected, React constructs a patch representing the actual change to the DOM and applies the patch.

* **AngularJS** began life with a close representation of a MVC (Model-View-Controller) pattern, but has since evolved into a MVVM (Model View View Model) -MVC hybrid architecture.

**React**, on the other hand, is focused on the “V” in MVC – it is designed to work with data representation, and leaves other elements of the application architecture to other components as selected by the programmer.

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

**Answer:**

Vue.js is an [open-source](https://en.wikipedia.org/wiki/Open-source_software) [JavaScript framework](https://en.wikipedia.org/wiki/JavaScript_framework) for building interfaces and [single-page applications](https://en.wikipedia.org/wiki/Single-page_application). Being a younger framework than both React and Angular, Vue took the better things from each, a mix of functional and OO programming.

By default, Vue’s coding style is a little similar in some areas to Angular but also removes most of Angular’s pain points. Vue separates HTML, JS and CSS, like web developers have been used to for years, but it also allows to use JSX if you prefer that style. So it doesn’t force changing your code style.

Vue’s take of the component lifecycle is more straightforward and more intuitive than React’s. In general, Vue’s API is broader but simpler than React.

Example:

By adding a lang=”scss” attribute to the <style> tag, one can write SCSS instead of plain CSS. Similarly, by adding the *scoped* attribute to the <style> tag, Vue components will implement CSS scoping (a.k.a CSS Modules) out of the box.

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

**Answer:**

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

* **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: “[( )]”.

In Angular, some directives have changed their names like **ng-repeat** to **ngFor**.

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

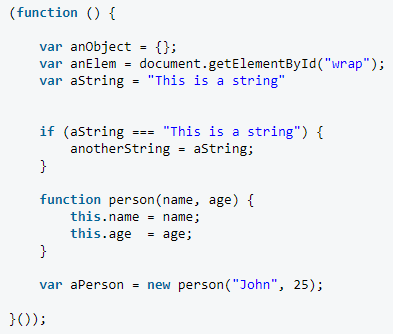
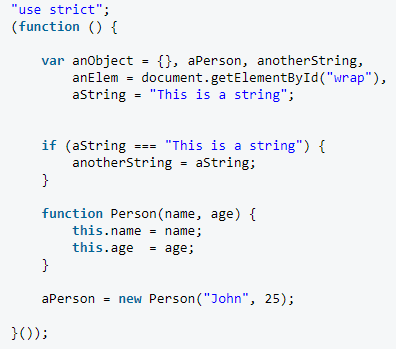
**Answer:**

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

**JSLint** is a static code analysis tool used in software development for checking if JavaScript source code complies with coding rules.

Example:

Before JSLint After JSLint

**ESLint** is a tool for identifying and reporting on patterns found in ECMAScript/JavaScript code, with the goal of making code more consistent and avoiding bugs.

Example:

var module = angular.module('uploader', []);

module.service('uploader', ['$q', function($q) {

function readyStateChange(deferred, xhr) {

if(xhr.readyState == 4) {

if(xhr.status == 200) {

deferred.resolve(JSON.parse(xhr.responseText));

}

else {

deferred.reject('HTTP status ' + xhr.status);

}

}

}

function onProgress(deferred, xhr, ev) {

if(ev.lengthComputable) {

deferred.notify({ loaded: ev.loaded, total: ev.total });

}

}

return {

send: function(url, data) {

var fd = new FormData();

for(var k in data) {

fd.append(k, data[k]);

}

var d = $q.defer();

var xhr = new XMLHttpRequest();

xhr.open('POST', url, true);

xhr.onreadystatechange = readyStateChange.bind({}, d, xhr);

xhr.upload.onprogress = onProgress.bind({}, d, xhr);

xhr.send(fd);

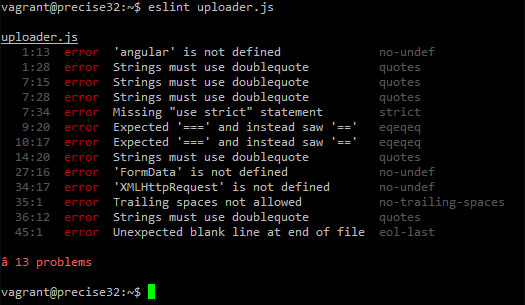
return d.promise;

}

};

}]);

Below is the ESLint output for the example module.



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

**Answer:**

AngularJS is preferred to be used instead of AJAX when making Single Page applications like Gmail,Twitter,Google etc.

AJAX is used for Form Validations like for Login Forms validation.