National University
of computer and emerging sciences

Research Assignment web programming

Submitted from: Omer Ali(16L-6309)

Submitted to : Shamsa Abid

Q#1 : MongoDB package vs Mongoose

MongoDB package	Mongoose
• In terms of Node.js, mongodb is the **native driver** for interacting with a mongodb instance. MongoDB driver is very powerful it is not particularly easy to work with. It also does not offer a built-in way of defining and maintaining data structures. • If your collection schema is unpredictable, or you want a Mongo-shell like experience inside Node.js, then go ahead and use the MongoDB driver.	• Mongoose is an **Object modeling tool** for MongoDB. Mongoose is built upon the MongoDB and expose most of the functionality of the native driver, but in a more convenient way, designed to fit into the flows of application development. Mongoose really enables us to define data structures and models, maintain them, and use them to interact with our database. • In 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.

Code examples for CRUD in mongoDB

Create:

```
const insertDocuments = function(db, callback) {
  // Get the documents collection
  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);
  });
}
```

Read:

```
const findDocuments = function(db, callback) {
  // Get the documents collection
  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);
  });
}
```

Update:

```
const updateDocument = function(db, callback) {
  // Get 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);
  });
}
```

Delete:

```
const removeDocument = function(db, callback) {
  // Get the documents collection
  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);
  });
}
```

Q#2 : Put vs Post

PUT	POST
PUT is used to update an existing document in database. If the Request-URI refers to an already existing resource – an update operation will	POST is used to create a new entity in database. 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

happen, otherwise create operation should happen if Request-URI is a valid resource URI.	identified by the Request-URI in the Request-Line.

Q#3 : Put vs Patch

PUT	Patch
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.	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 put method is used to update data when you are sending an entire entity and expecting to change some entities.	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.
Put is idempotent which means that if you update an entity with same update multiple times, it will send same response everytime.	Patch is not idempotent.

- In **case study**, updating name in an online university application form **PUT** should be used because in case of form the whole entity is returned to the server.

Q#4 : React vs Angular

React and Angular both used to create single page applications (SPA).

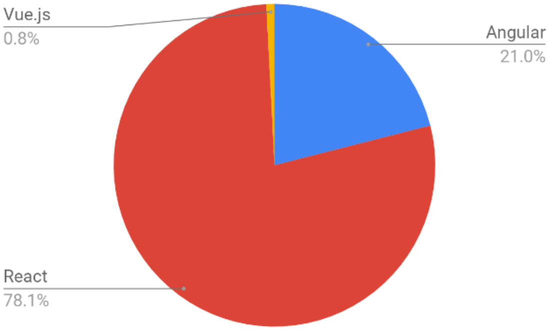
React	Angular
React is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies	Angular is a TypeScript-based open-source front-end web application platform led by the Angular Team at Google and by a community of individuals and corporations
One way Data binding React uses *one-way* data binding, meaning we are able to direct the flow of data only in one direction. Because of this, it’s always clear where the data was changed.	Two way data binding AngularJS connects Document Object Model (DOM) values to Model data through the Controller using two-way data binding.
Requires additional tools to manage dependencies	Manages dependencies automatically
React work with virtual DOM which increase its performance as compare to angular.	Angular uses original DOM.

Q#5 Vue.js vs (Angular and React)

Vue.js is an open-source JavaScript framework for building user interfaces and single-page applications. It was released in February 2014 by ex-Google-employee EvanYou.

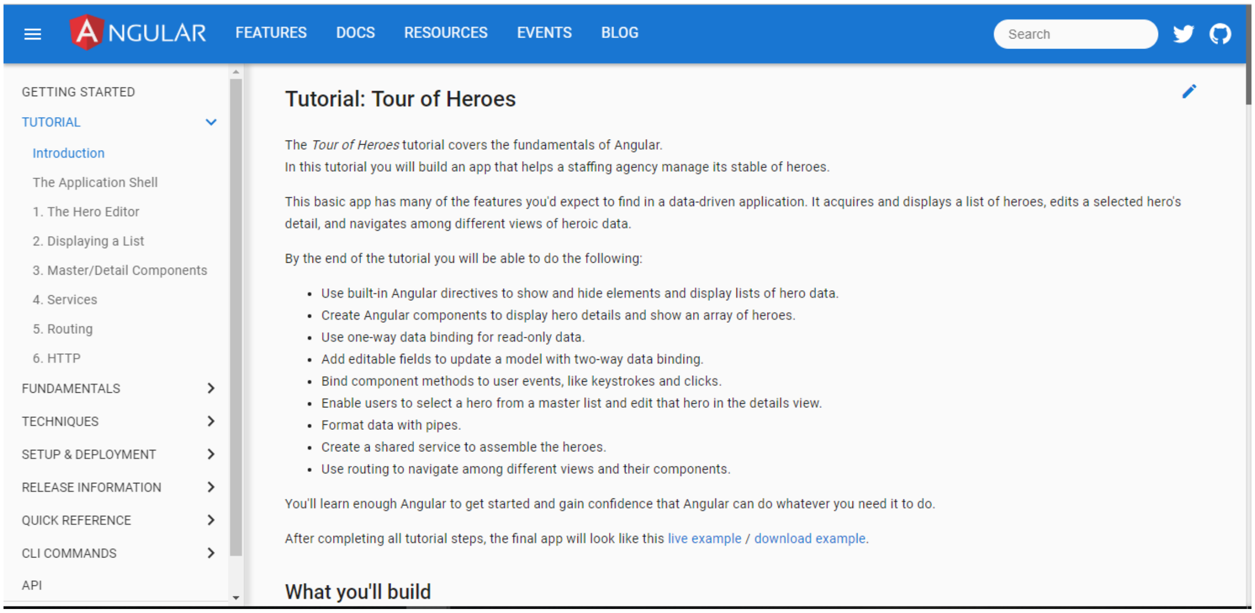
Comparison:

- Virtual DOM model is very helpful in terms of performance. Both, React and Vue has a Virtual DOM. Due to a well-built structure, Vue delivers great performance and memory allocation.
- Detailed documentation. Vue.js has very circumstantial documentation which can fasten learning curve for developers and save a lot of time to develop an app using only the basic knowledge of HTML and JavaScript.
- Vue has one way data binding.
- Detailed documentation. Vue.js has very circumstantial documentation which can fasten learning curve for developers and save a lot of time to develop an app using only the basic knowledge of HTML and JavaScript.
- If you want separation of concerns in your application, use Vue.



Q#6 Angular.io

Angular.io is a website to learn angular. It contain complete tutorials for angular.



Tutorial: Tour of Heroes

The *Tour of Heroes* tutorial covers the fundamentals of Angular.
In this tutorial you will build an app that helps a staffing agency manage its stable of heroes.

This basic app has many of the features you'd expect to find in a data-driven application. It acquires and displays a list of heroes, edits a selected hero's detail, and navigates among different views of heroic data.

By the end of the tutorial you will be able to do the following:

- Use built-in Angular directives to show and hide elements and display lists of hero data.
- Create Angular components to display hero details and show an array of heroes.
- Use one-way data binding for read-only data.
- Add editable fields to update a model with two-way data binding.
- Bind component methods to user events, like keystrokes and clicks.
- Enable users to select a hero from a master list and edit that hero in the details view.
- Format data with pipes.
- Create a shared service to assemble the heroes.
- Use routing to navigate among different views and their components.

You'll learn enough Angular to get started and gain confidence that Angular can do whatever you need it to do.

After completing all tutorial steps, the final app will look like this live example / download example.

What you'll build

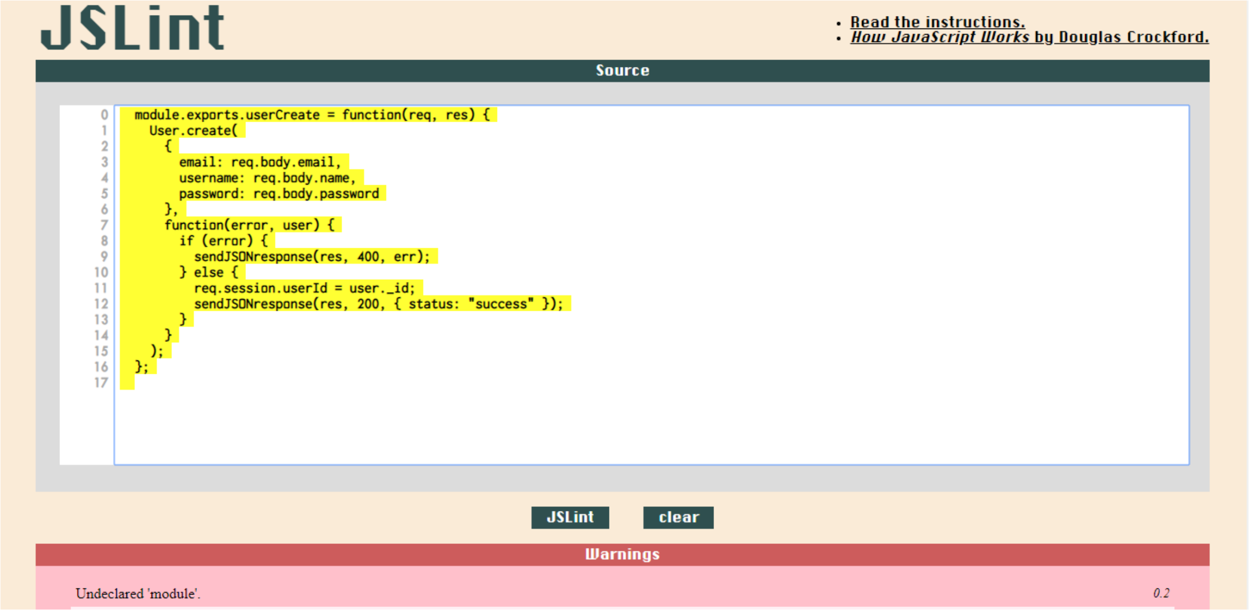
Q#7 Linting

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

A **linter or lint** refers to tools that analyze source code to flag programming errors, bugs, stylistic errors, and suspicious constructs.

JSLint:

It is a website, which analyze the JavaScript code. You copy and paste your code into their editor and the application gives a report about warnings and errors in your code.



```
module.exports.userCreate = function(req, res) {
  User.create(
    {
      email: req.body.email,
      username: req.body.name,
      password: req.body.password
    },
    function(error, user) {
      if (error) {
        sendJSONresponse(res, 400, err);
      } else {
        req.session.userId = user._id;
        sendJSONresponse(res, 200, { status: "success" });
      }
    }
  );
};
```

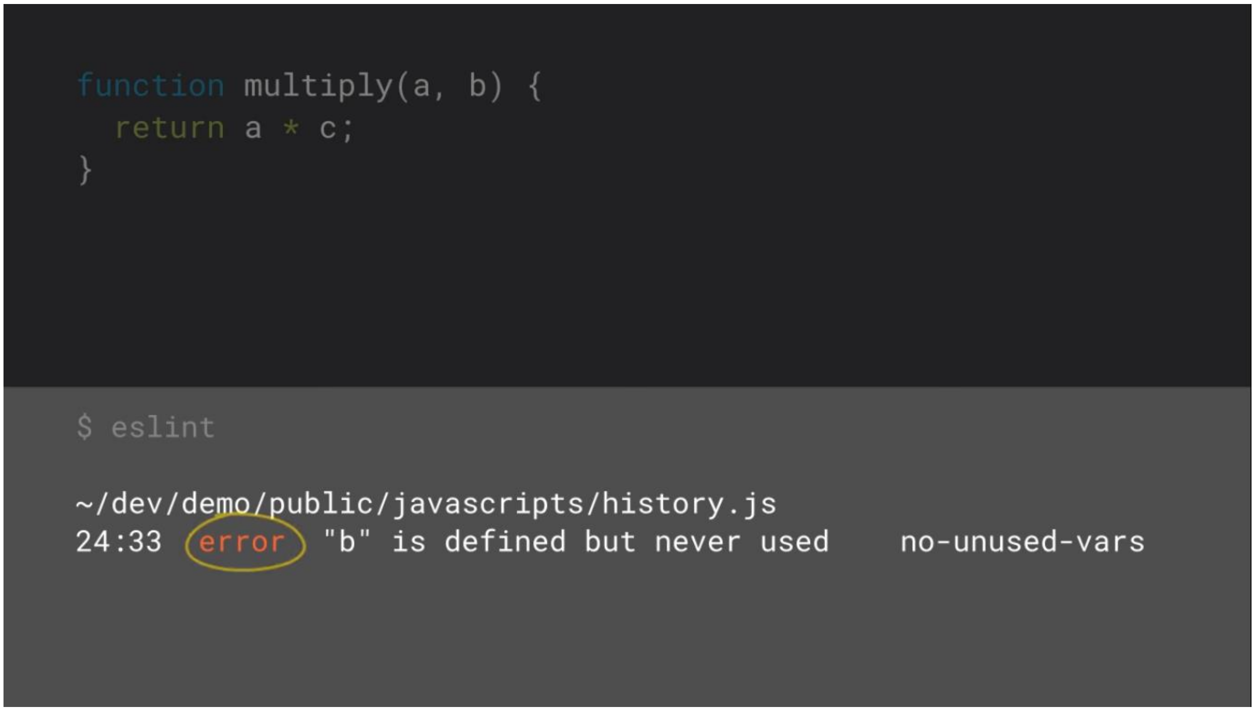
```
Undeclared 'module'.                                            0.2
```

ESLint:

A pluggable and configurable linter tool for identifying and reporting on patterns in JavaScript. It is an npm module to be installed within the project or globally.

```
$ npm install eslint --save-dev
```

You run eslint command from command line and it gives the report against your code.



```
function multiply(a, b) {
  return a * c;
}
```

```
$ eslint

~/dev/demo/public/javascripts/history.js
24:33  error  "b" is defined but never used    no-unused-vars
```

Q#8 : Angular vs Ajax

Ajax is the feature which allows you to update the part of the page without update or refresh the page while AngularJS is one of the JavaScript framework (to be specific) client side MVC framework.

Example for Ajax: If you only want to update some view without refreshing the page in multipage web application. For example, To check whether username exist in database or not then use Ajax.

Example for Angular: To make a single Page Application use Angular because it is complete framework for client side.

References

- https://restfulapi.net/rest-put-vs-post/
- https://stackoverflow.com/questions/630453/put-vs-post-in-rest
- https://stackoverflow.com/questions/28459418/rest-api-put-vs-patch-with-real-life-examples
- Getting MEAN With Mongo, Express, Angular, and Node. Book by Simon Holmes
- **https://rubygarage.org/blog/react-vs-angularjs**