WEB DEVELOPMENT USING MEAN STACK

Unit 2 - Express Framework

1. Introduction to Express.js?

- Node.js is fast, portable, and written in JavaScript but it does not directly support common tasks such as handling requests, serving files, and handling HTTP methods such as GET and POST. So a better framework had to fill the gap! And that's how emerged Express js.

- Express.js is the fastest, unopinionated, and simplest web framework for Node.js.

- It was flexibly created to build simple pages, multi-pages, and hybrid apps with robust features for web and mobile development.

- The framework is small with undisputed performances benefits. It is the leading Node.js framework designed to help you build web apps and APIs with powerful tooling for HTTP servers.

Characteristics of Express JS?

- **Nodejs Framework** – It is free to use the web framework of Node.js language.
- **Flexibility** – It is a framework that does not follow a standard structure. This implies that a developer can integrate any compatible module in the series of his choice.
- **Simplicity** – From a developer's perspective, learning and even writing the code in Express is simple. This is so because there are no exact rules or chronology to be followed while writing the code for a web app being built in Express JS.
- **Minimalism** – This framework only requires javascript, which makes it easy, quick, and simple to code the app into. Also, it uses less overhead in the core framework.
- **Scalability** – Since it is a lightweight framework, unlike the express js alternatives, it is easy to leverage it to scale an app built in this.
- **Architecture** – It follows a single-threaded event loop architecture.
- **Unopinionated** – There is no proper or strict format to be followed for writing the code or executing the elements.

Needs & Pillars of Express js?

- **Time Efficient–** Allows fast coding and even faster deployment procedures.
- **Money Efficient–** Express JS is not a very high-maintenance web backend framework.
- **Asynchronous–** The code running does not stop, it is a continuous process.
- **Extensive Uses–** Can be used to build a single page, multi-page, or hybrid web app.

Difference between Expressjs vs Nodejs?

Node.js	Express.js
Used to develop the frontend and backend of a web application.	Used to develop the backend of a web application.
It is a development platform.	It is a web framework.
It is used for both client and server-side programming.	It is used for server-side programming.
Fewer features comparatively.	More features than Node.js
Built on Google's V8 engine.	Built on Node.js
C, C++, JavaScript	Express.js written in JavaScript
TypeScript, CoffeeScript, Ruby	JavaScript

Advantages of Express JS?

1. Shorter Development Duration

The name itself suggests Express is known for its speedy development. It uses javascript, which further allows fast coding. As a matter of fact, developers have reported that it is easier and quicker to complete the backend programming in Express js.

2. Express.js Error Handling

The Expressjs error handling process is designed in a way that it is capable of detecting and solving bugs in both synchronous and asynchronous codes.

3. Appropriate I/O Request Handling

Depending on the seasons, on-demand service app businesses receive hundreds or sometimes even thousands of notifications per day. If built in the Express backend framework, such apps are at an advantage, as this framework is robust enough to receive numerous Input and Output requests all at once.

4. Unopinionated Framework

Developers love the Express framework, for it does not have any strict rules on the placement or order of code components.

5. Fast & Easy Installation

Usually, installing frameworks is a very tedious task. But, installing Express.js is pretty easy. Additionally, it is also simpler to set up, configure, and customize.

6. Ability to Create a REST Expressjs API Server

The main role of a REST API server is to access and use data. And none other than a developer or an IT geek can better understand the importance of creating a REST API server. Express allows a developer user to create the same with utmost security levels.

7. Easy Integration of Databases

Express is an easy-to-handle backend framework. One can integrate any database — MongoDB, Redis, MySQL — you name it, and its integration is possible.

What are ExpressJS Cons?

1. Middleware is a Philosophy- Developers who are not very familiar with Express can have difficulty understanding and executing the middleware functions in the coding.

2. No Structure – No Standardization-When more developers join the team to work on different functionalities, there can be little room for confusion. That is because Express does not follow a standard structure for typing the code. So, every developer builds the programs according to their understanding.

When to Use Express js? OR What is Express.js Used For?

- Single-page apps
- Reusable apps
- Middleware Applications
- RESTful APIs
- Serve and access static files using a browser

- Enterprise web applications
- Ecommerce web application

Following big-name brands use Express js?

- IBM
- Accenture
- Uber
- eBay
- PayPal
- LinkedIn
- FoxSports

Postman?

Postman is one of the most popular software testing tools which is used for API testing. With the help of this tool, developers can easily create, test, share, and document APIs.

Postman is a standalone software testing API (Application Programming Interface) platform to build, test, design, modify, and document APIs. It is a simple Graphic User Interface for sending and viewing HTTP requests and responses.

This tool has the ability to make various types of HTTP requests like GET, POST, PUT, DELETE, and convert the API to code for languages like JavaScript and Python.

Why Postman is used?

1. **Accessibility-** One can use it anywhere after installing Postman into the device by simply logging in to the account.
2. **Use Collections**-Postman allows users to build collections for their API-calls. Every set can create multiple requests and subfolders. It will help to organize the test suites.
3. **Test development-** To test checkpoints, verification of successful HTTP response status shall be added to every API- calls.
4. **Automation Testing-**Tests can be performed in several repetitions or iterations by using the Collection Runner or Newman, which saves time for repeated tests.
5. **Creating Environments-** The design of multiple environments results in less replication of tests as one can use the same collection but for a different setting.
6. **Debugging-** To effectively debug the tests, the postman console helps to track what data is being retrieved.
7. **Collaboration-** You can import or export collections and environments to enhance the sharing of files. You may also use a direct connection to share the collections.
8. **Continuous integration-**It can support continuous integration.

Environment Variables in Node.js?

Environment variables are predetermined values that are typically used to provide the ability to configure a value in your code from outside of your application.

.env

```
MY_SECRET_KEY="shhhhh"
```

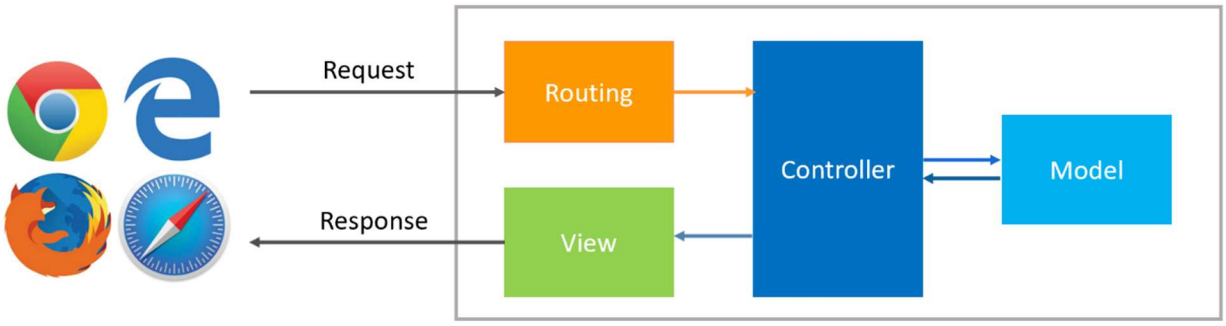
lib/authorization.js

```
function authorizeService() {
  const auth = Service.auth({
    key: process.env.MY_SECRET_KEY
  })
}
```

MY_SECRET_KEY environment variable used for authorization

- Install the dotenv library : npm install dotenv
- Create an .env file. ...
- Wherever you need to use environment variables (e.g. in GitLab, in node js, in Heroku , …) you need to add your environment variables

Routing in Express js?



- Routing refers to how an application's endpoints (URIs) respond to client requests.
- Routing refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).
- Each route can have one or more handler functions, which are executed when the route is matched.
- Route definition takes the following structure:

```
app.METHOD(PATH, HANDLER)
```

Where:

- app is an instance of express.
- METHOD is an HTTP request method, in lowercase.
- PATH is a path on the server.
- HANDLER is the function executed when the route is matched.

The following examples illustrate defining simple routes.

Respond with `Hello World!` on the homepage:

```
app.get('/', (req, res) => {
  res.send('Hello World!')
})
```

Respond to POST request on the root route (/), the application's home page:

```
app.post('/', (req, res) => {
  res.send('Got a POST request')
})
```

Pug templates?

- Pug is a high performance, robust, elegant, feature rich and easy to understand template engine. Pug is implemented using JavaScript for node.js and web browsers.

- Pug was formerly known as jade. However "jade" was a registered trademark and as a result a rename was required. After some brainstorming, the maintainers come up with the name pug

- In order to understand the concepts properly, the programmer must have some basic understanding of the following: HTML , CSS , Javascript and node.js

- It provides the ability to write dynamic and reusable HTML documents, it's an open source HTML templating language for Node.js (server-side JavaScript), totally free to use and provides fast, easy, and fun HTML.

- Just like the programming language Python, Pug works with indentation or white spaces, like this example:

```
doctype html
html(lang='en')
 head
   title Pug
 body
   h1 Pug Examples
   div.container
     p Cool Pug example!
```

- Here are no closing tags, everything is indented and you scan the file much quicker. Also by using Pug we can ensure that our HTML is well-formed and valid. This will translate to:

```
<!DOCTYPE html>
<html lang="en">
 <head>
   <title>Pug</title>
 </head>
 <body>
   <h1>Pug Examples</h1>
   <div class="container">
     <p>Cool Pug example!</p>
   </div>
 </body>
</html>
```

During the time Pug is compiling the .pug file to plain HTML, the compiler throws build errors if the indention in your file isn't correct.

HTTP method of Express?

GET-The GET method requests a representation of the specified resource. Requests using GET should only retrieve data and should have no other effect.

POST- The POST method requests that the server accept the data enclosed in the request as a new object/entity of the resource identified by the URI.

PUT- The PUT method requests that the server accept the data enclosed in the request as a modification to existing object identified by the URI. If it does not exist then the PUT method should create one.

DELETE -The DELETE method requests that the server delete the specified resource.

URL binding in Expressjs?

- In the previous chapter we define routes, but those routes are static and fixed. In this chapter, we are going to learn about URL parameters and routing in Express.js.

- To use the dynamic routes, we should provide different types of routes. Using dynamic routes allows us to pass parameters and process based on them.

- The URL module splits up a web address into readable parts.

```
var express = require('express');
var app = express();

app.get('/:id', function(req, res){
    res.send('The id you specified is ' + req.params.id);
});

app.listen(2000);
```

To run this go to http://localhost:2000/abc. The following response will be displayed.

```
The id you specified is abc
```

Lets us go with some complex example:

```
var express = require('express');
var app = express();

app.get('/things/:name/:id', function(req, res) {
    res.send('id: ' + req.params.id + ' and name: ' + req.params.name);
});
app.listen(2000);
```

Note: Route parameters are named URL segments that are used to capture the values specified at their position in the URL. The captured values are populated in the`req.params` object, with the name of the route parameter specified in the path as their respective keys.

To run this go to http://localhost:2000/things/pabbly/abcde. The following response will be displayed.

```
id: abcde and name: pabbly
```

Middleware function in Express js?

- ✓ Middleware functions are functions that have access to the request object (req), the response object (res), and the next middleware function in the application's request-response cycle. The next middleware function is commonly denoted by a variable named next.
- ✓ Middleware is executed during the window between when a server receives a request and when it sends a response.

Middleware functions can perform the following tasks:

- Execute any code.
- Make changes to the request and the response objects.
- End the request-response cycle.
- Call the next middleware function in the stack.

If the current middleware function does not end the request-response cycle, it must call next() to pass control to the next middleware function. Otherwise, the request will be left hanging.

An Express application can use the following types of middleware:

- Application-level middleware
- Router-level middleware
- Error-handling middleware
- Built-in middleware
- Third-party middleware

Serving Static files in Express?

- Express provides built-in middleware that allows us to serve static files directly from the file system.

- Static files such as images, CSS files, and JavaScript files, use the express.static built-in middleware function in Express.

Here is the simple example code:

```
const express = require('express);
const router = express.Router();
const app = express();

app.use(express.static('public'));

app.use('/', router);
app.listen(3000);
```

Assuming you have files stored in the **public** folder, you can now make the request to the files from the browser.

```
app.use(express.static('public'))
```

Now, you can load the files that are in the `public` directory:

```
http://localhost:3000/images/kitten.jpg
http://localhost:3000/css/style.css
http://localhost:3000/js/app.js
http://localhost:3000/images/bg.png
http://localhost:3000/hello.html
```

Express sessions?

- A website is based on the HTTP protocol. HTTP is a stateless protocol which means at the end of every request and response cycle, the client and the server forget about each other.

- This is where the session comes in. A session will contain some unique data about that client to allow the server to keep track of the user's state. In session-based authentication, the user's state is stored in the server's memory or a database.

- Express-session - an HTTP server-side framework used to create and manage a session middleware.

```
const oneDay = 1000 * 60 * 60 * 24;
app.use(sessions({
    secret: "thisismysecrctekeyfhrgfgrfrty84fwir767",
    saveUninitialized:true,
    cookie: { maxAge: oneDay },
    resave: false
}));
```

Secret - a random unique string key used to authenticate a session. It is stored in an environment variable and can't be exposed to the public.

Resave - takes a Boolean value. It enables the session to be stored back to the session store, even if the session was never modified during the request. This can result in a race situation in case a client makes two parallel requests to the server. Thus modification made on the session of the first request may be overwritten when the second request ends. The default value is true. However, this may change at some point, false is a better alternative.

SaveUninitialized - this allows any uninitialized session to be sent to the store. When a session is created but not modified, it is referred to as uninitialized.

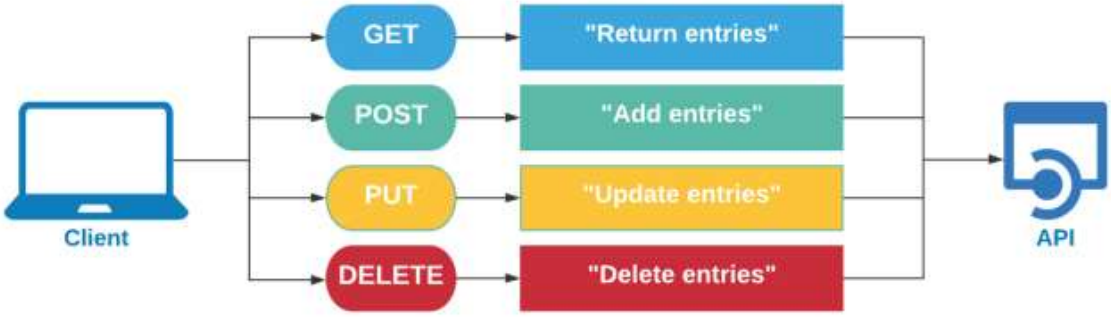
Cookie: { maxAge: oneDay } - this sets the cookie expiry time. The browser will delete the cookie after the set duration elapses.

REST full API's?

An API, or Application Programming Interface, can be simply defined as an interaction between various software components. When a user clicks a button to see a list of their Facebook friends, likes on Instagram, or emails within an inbox, data is generally being exchanged in one way or another through a web service API.

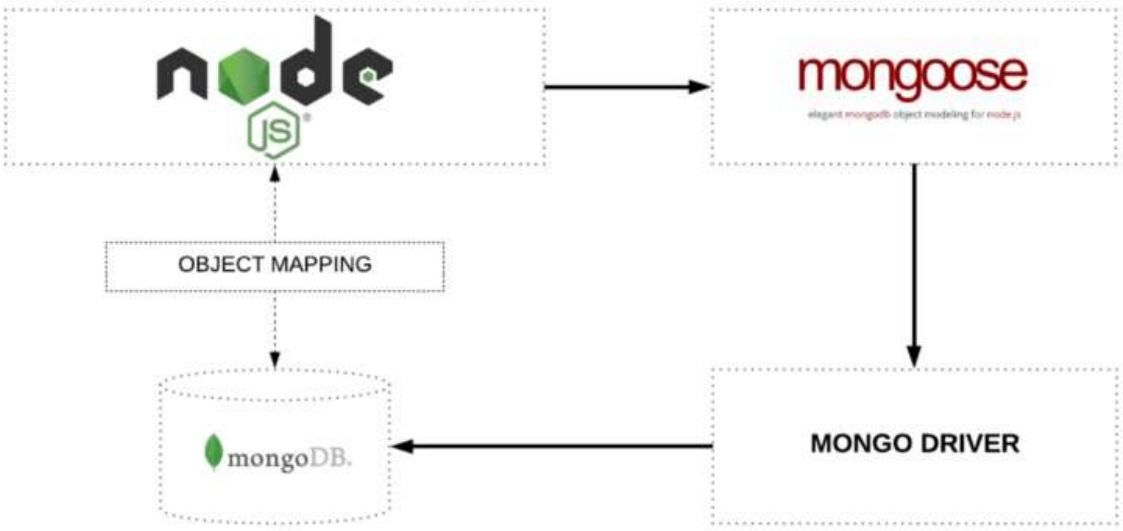
There are a number of different types of web service APIs such as:

1. **SOAP** — Simple Object Access Protocol
2. **XML-RPC** — XML Format for Data Transfer
3. **JSON-RPC** — JSON Format for Data Transfer
4. **REST** — Representational State Transfer



Introduction to Mongoose for MongoDB?

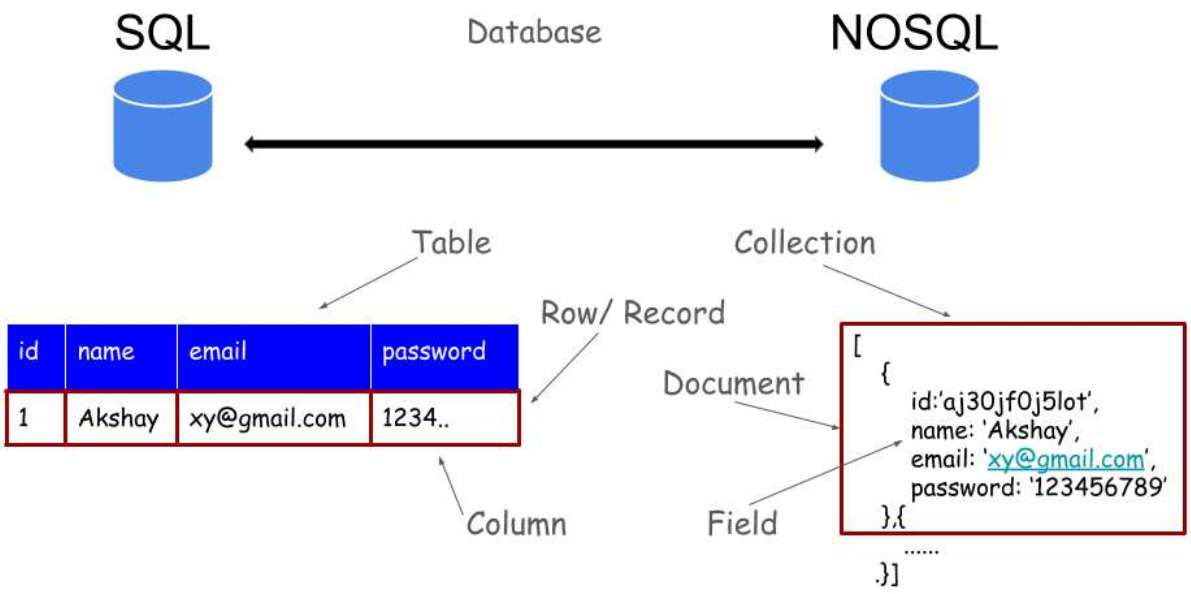
Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.



Object Mapping between Node and MongoDB managed via Mongoose

MongoDB is a schema-less NoSQL document database. It means you can store JSON documents in it, and the structure of these documents can vary as it is not enforced like SQL databases. This is one of the advantages of using NoSQL as it speeds up application development and reduces the complexity of deployments.

Below is an example of how data is stored in Mongo vs. SQL Database



1. **Database -** To get started, the term database is used in both. The term **‘ database ’** is defined as any collection of electronic records that can be processed to produce useful information.
2. **Collections -** ‘Collections’ in Mongo are equivalent to tables in relational databases. They can hold multiple JSON documents.
3. **Documents -** ‘Documents’ are equivalent to records or rows of data in SQL. While a SQL row can have reference to data in other tables, Mongo documents usually combine that in a document.
4. **Fields -** ‘Fields’ or attributes are similar to columns in a SQL table.

```
npm install mongoose --save
```

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/SocialMediaApp_db');
```

With these two lines in place, the mongoose is now configured. V

Assignment - 2