Nodejs

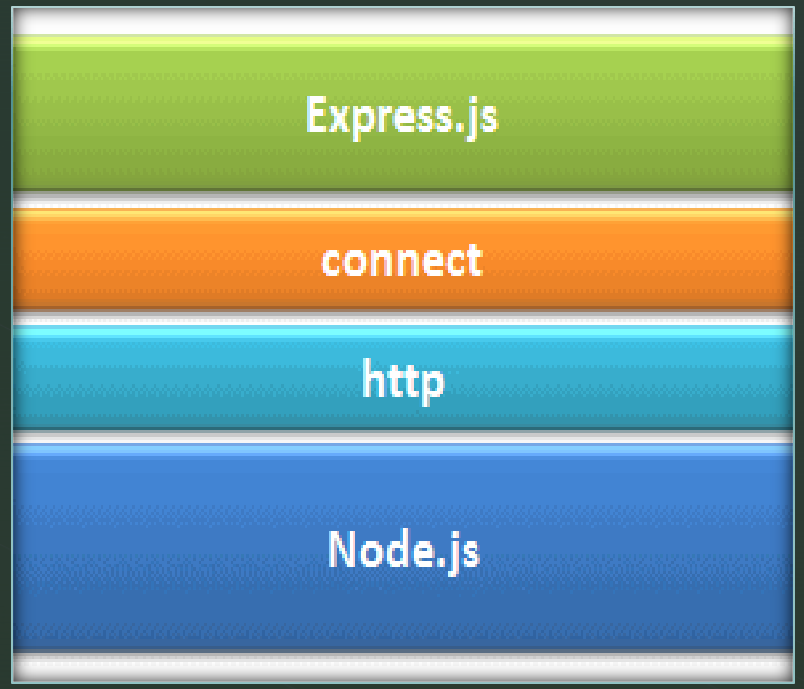
Day 2

By Mostafa Elewa

Email: mostafatechhub@gmail.com

Express.js

- *Express.js is a web application framework for Node.js. It provides various features that make web application development fast and easy which otherwise takes more time using only Node.js.*
- *Express.js is based on the Node.js middleware module called connect which in turn uses http module. So, any middleware which is based on connect will also work with Express.js.*



Express.js

- ***Advantages of Express.js***

1. ***Makes Node.js web application development fast and easy.***
2. ***Easy to configure and customize.***
3. ***Allows you to define routes of your application based on HTTP methods and URLs.***
4. ***Includes various middleware modules which you can use to perform additional tasks on request and response.***
5. ***Easy to integrate with different template engines like Jade, Vash, EJS etc.***
6. ***Allows you to define an error handling middleware.***
7. ***Easy to serve static files and resources of your application.***
8. ***Allows you to create REST API server.***
9. ***Easy to connect with databases such as MongoDB, Redis, MySQL***

Express.js Web Application

- ***Web Server***
- ***First of all, import the Express.js module and create the web server.***

Configure Routes

- *Use app object to define different routes of your application. The app object includes get(), post(), put() and delete() methods to define routes for HTTP GET, POST, PUT and DELETE requests, respectively.*

Serving Static Resources in Node.js

- *it is easy to serve static files using built-in middleware in Express.js called express.static. Using express.static() method, you can server static resources directly by specifying the folder name where you have stored your static resources.*
  - *app.use(express.static(__dirname + 'public'))*

MongoDB

- *In order to access MongoDB database, we need to install MongoDB drivers. To install native mongodb drivers using NPM, open command prompt and write the following command to install MongoDB driver in your application.*
  - *npm install mongodb*

MySQL

- *To connect to MySQL DB*
- *Install MySQL node pakage*
  - *var mysql = require('mysql');*
  - *var connection = mysql.createConnection({*
  - *host : 'localhost',*
  - *user : 'me',*
  - *password : 'secret',*
  - *database : 'my_db'*
  - *});*
  - *connection.connect();*
  - *connection.query('SELECT 1 + 1 AS solution', function (error, results, fields) {*
  - *if (error) throw error;*
  - *console.log('The solution is: ', results[0].solution);*
  - *});*

Middleware

- ***Every middleware can access each HTTP request and respond to every path that it is attached to. Moreover, the middleware can end the HTTP request independently or transfer it to another middleware using the next function. It will allow you to categorize the code and generate reusable middleware, just like chaining.***

Application-level middleware

- *In the application-level middleware, we consider an authentication middleware and how it can be created. When the user is not authenticated, it will not be possible to call the mentioned routes. When it is necessary to build an authentication for every GET, POST call, the development of an authentication middleware will follow.*

- *When you receive the authentication request, the authentication middleware makes progress towards the authentication code logic that is available inside it. Once the authentication is successful, the rest of the route can be called using the next function. However, when it fails, you may not be able to perform the next route as the middleware will show errors.*

Router-level middleware

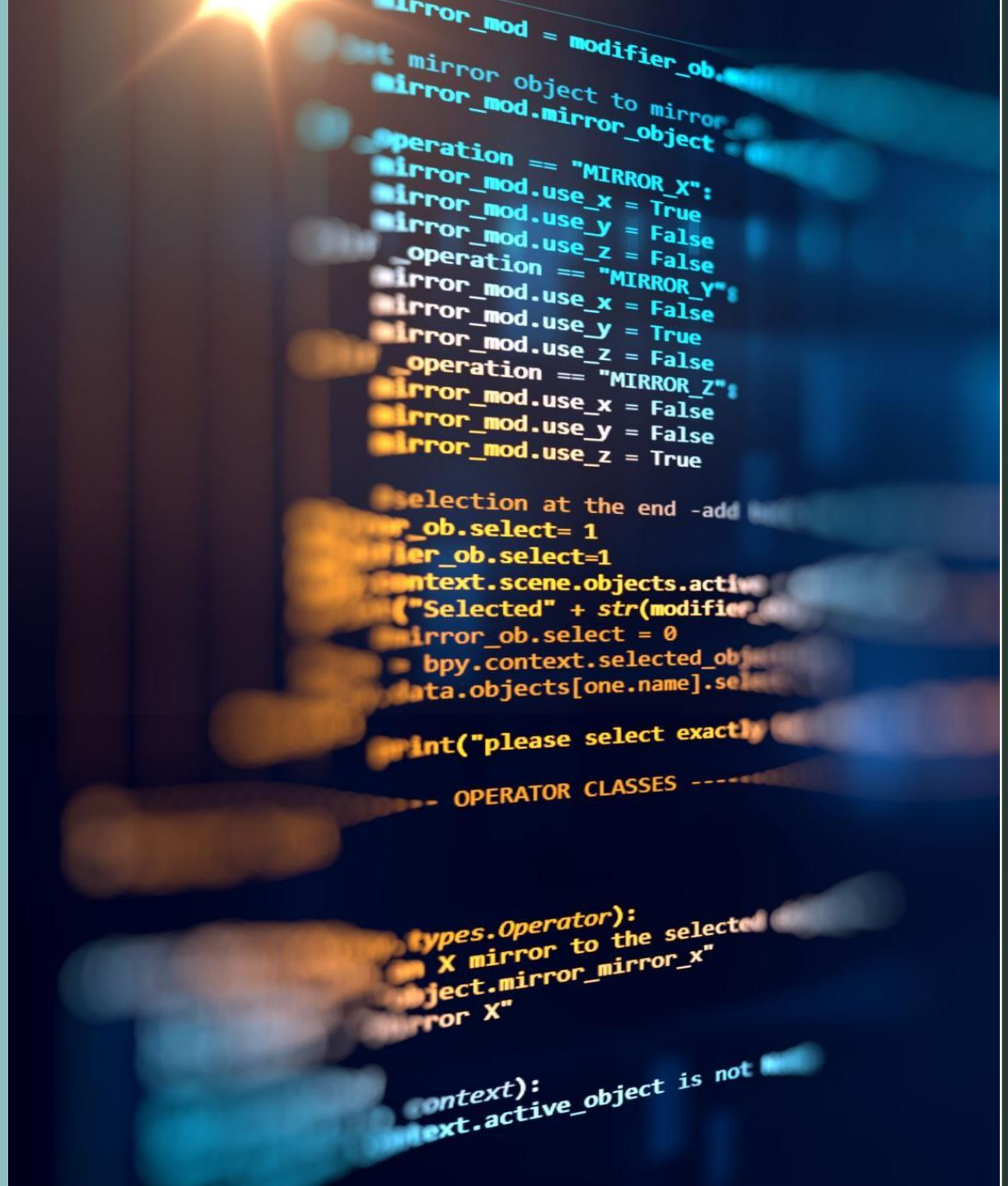
- *Router-level middleware is almost like the application-level middleware and works in the same way. The difference is that it can generate and limit an instance using the Express.Router() function. You can make use of the router.use() and router.METHOD() functions to load router-level middleware.*

Error-handling middleware

- *Express.js can handle any default errors and can also define error-handling middleware functions, which are like the other middleware functions. The major difference is the error-handling functions.*

Third-party middleware

- *Sometimes, you will need to have some additional features in the backend operations. For that, you can install the Node.js module for the specific function and then apply the same to your application (either on the application or router level).*



To be continued...

Thank you