Developing Back-End Apps with Node.js and Express

Table of Contents

[Back-end and Front-end Development Overview 1](#_Toc180778010)

[Getting Started with Node.js 2](#_Toc180778011)

[Advanced Node.js Modules 2](#_Toc180778012)

[Module 1 Summary 2](#_Toc180778013)

[Glossary: Introduction to Server-Side JavaScript 2](#_Toc180778014)

[Cheatsheet: Introduction to Server-Side JavaScript 2](#_Toc180778015)

[Promises, Async/Await, and Axios Requests in Node.js and Express 2](#_Toc180778016)

[Module Summary 3](#_Toc180778017)

[Glossary - Asynchronous I/O with Callback Programming 3](#_Toc180778018)

[Cheatsheet - Asynchronous I/O with Callback Programming 3](#_Toc180778019)

[Introduction to Middleware & Routers 3](#_Toc180778020)

[Authentication and Authorization in Node.js 4](#_Toc180778021)

[HTTP Methods and Rest APIs 4](#_Toc180778022)

[Module Summary 4](#_Toc180778023)

[Glossary - Express Web Application Framework 4](#_Toc180778024)

[Cheatsheet - Express Web Application Framework 4](#_Toc180778025)

## Back-end and Front-end Development Overview

<https://cf-courses-data.static.labs.skills.network/mH1iN5P2PJlTeg675ZBoGQ/backend-vs-frontend-v1.md.html?origin=www.coursera.org>

## Getting Started with Node.js

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-CD0220EN-SkillsNetwork/Readings/ServerSideJavascript.md.html?origin=www.coursera.org>

## Advanced Node.js Modules

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-CD0220EN-SkillsNetwork/Readings/Advanced_Node.js_modules.md.html?origin=www.coursera.org>

## Module 1 Summary

* Backend technologies include various types of servers and supporting infrastructures such as programming languages, frameworks, and other hardware.
* Node.js is the server-side component of JavaScript.
* The require statement can be called from anywhere in the app code, is bound dynamically, and is synchronous. The import statement must be called at the beginning of a file, is bound statically, and is not asynchronous.
* Client-side JavaScript is used to process front-end user interface elements, and server-side JavaScript is used to enable access to different kinds of servers and web applications.
* With server-side JavaScript, Node.js applications process and route web service requests from the client.
* To make a function or a value available to Node.js applications that import your module, add a property to the implicit exports object.
* Core modules include bare minimum functionality, local modules are those that you create for your application, and the Node.js community creates third-party modules.
* A local install means only the application within the directory of the installed can access the package, whereas a global install means that any application on the machine can access the package.

## Glossary: Introduction to Server-Side JavaScript

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-CD0220EN-SkillsNetwork/Glossary/200389.11_M1_Glossary.md.html?origin=www.coursera.org>

## Cheatsheet: Introduction to Server-Side JavaScript

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-CD0220EN-SkillsNetwork/Cheatsheets/200389.25_M1_CheatSheet.md.html?origin=www.coursera.org>

## Promises, Async/Await, and Axios Requests in Node.js and Express

<https://cf-courses-data.static.labs.skills.network/HL5EDg9nG5bPWjsaH-svbg/Await-and-Axios-Requests-v1.md.html?origin=www.coursera.org>

## Module Summary

* Asynchronous network operations can be handled using callback functions in order to prevent blocking JavaScript code
* A callback function must invoke another callback function to pass a message from the Node.js module back to the main application after the Node.js module receives a response message
* Nested callbacks can be difficult to read and debug. Inversion of control causes trust issues when dealing with third-party code
* Promise objects are most useful for operations that are time-consuming and can block resources
* JSON.parse() and JSON.stringify() are two methods used to parse JSON objects

## Glossary - Asynchronous I/O with Callback Programming

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-CD0220EN-SkillsNetwork/Glossary/200389.24_M2_Glossary.md.html?origin=www.coursera.org>

## Cheatsheet - Asynchronous I/O with Callback Programming

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-CD0220EN-SkillsNetwork/Cheatsheets/200389.25_M2_CheatSheet.md.html?origin=www.coursera.org>

## Introduction to Middleware & Routers

In this article, we will discuss the terms *middleware* and *routes*.

Middleware is software that sits between applications, databases, or services and allows those different technologies to communicate. It creates seamless interactions for the end user in a distributed system.

Express is a messaging framework used to handle routes and write middleware. The front end of an application uses Express to facilitate communication between components on the back end without the front-end and back-end services needing to use the same language. The front end communicates with the middleware, not directly with the back end.

Messaging frameworks like Express commonly contain JSON, REST APIs, and web services. Older messaging frameworks may contain extensible markup language (XML) and simple object access protocols (SOAP) instead of JSON and REST APIs, respectively. The messaging framework provides a standardized way to handle data transfer among different applications.

A web server is an example of middleware that connects a website to a database. The web server handles the business logic and routes the data from the database based on the request. A *route* is the part of the code that associates an HTTP request, such as GET, POST, or DELETE, with a URL and the function that gets called that handles that URL. Routing is used in web development to split an application’s user interface based on rules identified by the browser’s URL.

Router functions are called “middleware” collectively. Middleware is responsible for responding to an HTTP request or calling another function in the middleware chain. Express handles router functions through the Router class, such as Router.get(). As the name suggests, Router.get() handles HTTP GET requests. Other Router functions include Router.post(), Router.put(), Router.delete() in mostly the same way. These methods take two arguments, a URL path and a callback function.

In addition to routing, middleware is also responsible for providing secure connections among services by encrypting and decrypting data, managing application loads by distributing traffic to different servers, and sorting or filtering data before the data is returned to the client.

## **Authentication and Authorization in Node.js**

<https://cf-courses-data.static.labs.skills.network/huRoyukqvs1OMrWLcr8RDQ/Typesofauthenication-v1.md.html?origin=www.coursera.org>

## HTTP Methods and Rest APIs

<https://cf-courses-data.static.labs.skills.network/HgSYGZqoKUWhw4z4Yq_MwA/HTTP-Methods-and-Rest-APIs-v1.md.html?origin=www.coursera.org>

## Module Summary

* Developers rely on third-party packages to extend Node.js.
* You can use the npm application to manage Node.js packages in your Node.js framework installation.
* The MVC architecture style divides a back-end application into three parts: the model, the view, and the controller.
* REST API frameworks use HTTP methods to communicate with each other.
* Express abstracts low-level details.
* Routing can be handled at the application level or at the router level.
* Five types of middleware are application level, router level, error handling, built-in middleware, and third party.
* Template rendering is the ability of the server to fill in dynamic content.
* The npm jsonwebtoken package should be required in an Express application to authenticate a user.

## Glossary - Express Web Application Framework

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-CD0220EN-SkillsNetwork/Glossary/200389.38_M3_Glossary.md.html?origin=www.coursera.org>

## Cheatsheet - Express Web Application Framework

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-CD0220EN-SkillsNetwork/Cheatsheets/200389.39_M3_CheatSheet.md.html?origin=www.coursera.org>