13.8.1 Key Terms

CommonJS
CRUD
Database-as-a-Service
Environment variables
Express
Functions-as-a-Service
JAM Stack
middleware
module
Node
nonblocking
npm
Platform-as-a-Service
push-based
route
serverless computing
templates
V8
views
view engine
WebSockets

13.8.2 Review Questions

1. What are the key advantages and disadvantages of using Node?
2. What is npm? What is its role in contemporary web development?
3. A nonblocking architecture can typically handle more simultaneous requests. Why is that?
4. What are modules in JavaScript? How does the Node CommonJS module system differ from the one introduced in ES6?
5. In the context of Node, what is Express?
6. What are Express routes?
7. In Express, what is middleware?
8. What is a CRUD API?
9. What are WebSockets? How do they differ from HTPP?
10. What role do view engines play in Node?
11. What are the benefits of serverless computing?
12. How does functions-as-a-service differ from platform-as-a-service?

13.8.3 Hands-On Practice

PROJECT 1:

DIFFICULTY LEVEL: Beginner

Overview

In this project, you will be creating a data retrieval API.

Instructions

1. You have been provided a folder named **project1,** that contains the data and other files needed for this project. Use `npm init` to setup the folder, and `npm install` to add express.
2. Name your server file **art.js**. Add a static file handler for resources in the `static` folder.
3. The data for the APIs is contained in a supplied json file. Create a provider module for this file.

4. Add the following `GET` route handlers:

Route	Description
`/`	Returns JSON for all paintings
`/:id`	Returns for just a single painting
`/gallery/:id`	Returns all paintings for a specific gallery id
`/artist/:id`	Returns all paintings for a specific artist id
`/year/min/max`	Returns all paintings whose `yearOfWork` field is between the two supplied values.

Guidance and Testing

1. Break this down into small steps and test after each step.

PROJECT 2:

DIFFICULTY LEVEL: Intermediate

Overview

In this project, you will be creating a full CRUD API.

Instructions

1. You have been provided a folder named **project2,** that contains the data and other files needed for this project. Use `npm init` to set up the folder, and `npm install` to add express.
2. Add a static file handler for resources in the **static** folder.
3. The data for the APIs is contained in a supplied json file. Create a provider module for this file.
4. Add the following `GET` route handlers:

Route	Description
`/`	Returns JSON for all companies
`/:id`	Returns for just a single company

5. Add `PUT`, `POST`, and `DELETE` route handlers, to handle updating an existing company, inserting a new company, and deleting an existing company.
6. Use the supplied form **tester.html** to verify your APIs work as expected.

Guidance and Testing

1. Break this down into small steps and test after each step.
2. While there is a provided form that you can use to test your APIs, it is often easier to test your APIs using a tool such as Postman and Insomnia. We recommend that you install one of these tools and try testing your API with it.

PROJECT 3:

DIFFICULTY LEVEL: Intermediate

Overview

In this project, you will create a more sophisticated chat application.