MODULE 03

BACKEND DEVELOPMENT WITH Node.js AND Database

➤ ASSIGNMENT 01

Node.js Cheat Sheet

◆ Introduction to Node.js

Node.js is a powerful, open-source runtime environment that enables the execution of JavaScript on the server side. It is built on the V8 JavaScript engine and employs an event-driven, non-blocking I/O model, making it ideal for building scalable network applications. This cheat sheet summarizes the essential concepts of Node.js, including its architecture, asynchronous programming capabilities, and the npm ecosystem.

◆ Key Concepts

1. Event-Driven Architecture

- Definition: Node.js utilizes an event-driven architecture where events trigger callbacks.
- Functionality: This design allows the server to handle multiple connections simultaneously without blocking the execution thread.
- Event Loop: The event loop processes tasks in the event queue, enabling efficient management of I/O operations.

2. Asynchronous Programming

- Core Feature: Asynchronous programming is central to Node.js, allowing non-blocking execution of tasks.
- Mechanism:
  - Callbacks: Functions passed as arguments that are executed after a task completes.

- Promises: Objects representing the eventual completion (or failure) of an asynchronous operation.
- Async/Await : Syntactic sugar over promises that allows writing asynchronous code in a more synchronous style.

3. npm Ecosystem

- Node Pakage Manager (npm): A vital tool for managing packages and dependencies in Node.js applications.
- Package Installation: Use `npm install <package-name>` to add packages to your project.
- Package.json : A configuration file that holds metadata about the project and its dependencies.

4. Modules and CommonJS

- Modular Programming: Node.js supports modular design through the CommonJS module system.
- Exporting Modules: Functions or variables can be exported from a module using `module.exports`.
- Importing Modules : Use `require()` to include modules in other files, promoting code reusability and organization.

◆ Practical Implementation Steps

• Setting Up a Simple Node.js Project

1. Install Node.js: Download from the Official website and install it on your machine.
2. Create Project Directory:
```
mkdir my-node-project
cd my-node-project
```
3. Initialize Project:
```
npm init -y
```
This command creates a 'package.json' file with default settings.

• Installing Express.js Framework

Express.js simplifies routing and request handling:
```
> npm install express
```

• Creating a Basic Server

1. Create 'server.js' File:

[javascript]
```
const express = require('express');
const app = express();
const port = 3000;
app.get('/', (req, res) => {
    res.send('<h1>Hello, Express.js Server! </h1>');
});
app.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});
```
2. Run the Server:

Execute the following command in your terminal:
```
node server.js
```
Open a web browser and navigate to 'http://localhost:3000' to view the application.

◆ Development Tools :

- Nodemon : A utility that automatically restarts the server upon code changes.

```
npm install -g nodemon
nodemon server.js
```

◆ Conclusion

This cheat sheet encapsulates the fundamental concepts of Node.js, providing a concise reference for understanding its architecture, asynchronous programming model, npm ecosystem, and modular design principles. Mastery of these concepts is essential for developing robust and scalable applications using Node.js. By implementing a simple project, practical experience is gained, reinforcing theoretical knowledge and enhancing backend development skills.