# Node.js Cheat Sheet

## 1. Event-driven Architecture

→ Node.js utilizes an event-driven, non-blocking I/O model, making it lightweight and efficient for handling concurrent connections.

→ This architecture allows Node.js to handle multiple client requests without getting blocked, improving performance and scalability for real-time applications.

## 2. Asynchronous Programming

→ Node.js is designed to handle asynchronous programming, enabling multiple operations to be executed without blocking the execution flow.

→ It is achieved using callbacks, promises and async/await, allowing developers to write non-blocking code efficiently.

## 3. npm (Node Package Manager)

→ npm is the default package manager for Node.js

→ It simplifies the process of installing, managing, and sharing reusable javascript code packages.

→ With npm, developers can easily add functionality to their applications by installing packages from the npm registry.

## 4. Modules

→ Node.js uses a modular architecture, allowing developers to break down application into smaller, reusable modules.

→ Node.js implements the CommonJS module system, allowing modules to be imported and exported using require() and module.exports.

## 5. CommonJS Module System

→ Node.js follows the CommonJS module system, enabling modular development.

→ With CommonJS, modules can be created and reused easily.

## 6. Callbacks

→ Callbacks are functions passed as arguments to other functions in Node.js, faciliating asynchronous programming.

→ They allows asynchronous operations to be executed sequentially or in parallel.

## 7. Promises

→ Promises provide an alternative to callbacks for handling asynchronous operations in Node.js.

→ They simplify asynchronous code and make it more readable and maintainable.

→ Used to handle errors more effectively and improve code readability.

## 8. Async/Await

→ It is a modern approach to asynchronous programming in Node.js.

→ Async/Await builds on top of promises, providing a more intuitive way to handle asynchronous operations.