*Node Cookbook 3rd Edition*

[Book mission] by David Mark Clements

# Audience

# *If you have some knowledge of JavaScript and want to build fast, efficient, scalable client-server solutions, then Node Cookbook Third Edition is for you.*

# Mission

# *Experienced users of Node will improve their skills, and even if you have not worked with Node before, these practical recipes will make it easy to get started.*

# Objectives and achievements

*List the book’s key objectives here. What will the reader achieve with this book?*

* Gain a holistic yet intricate overview of Node.js
* Improve technical adeptness with practical examples
* Employ contemporary techniques to create human-symbiotic, readily scalable production systems
* Achieve deep understanding of Node internals for debugging and performance purposes
* Create secure JavaScript applications and systems
* Use best-of-breed libraries to rapidly create web services, interact with databases and transparently develop distributed systems
* Learn options, techniques and best practices for deploying Node software
* Advance from frontend JavaScript developer to Full Stack developer
* Transfer server experience with JavaScript knowledge to engineering network applications with Node.js

# Detailed outline

# **1: WRITING MODULES (45 pages)**

**Description:** In idiomatic Node, modules are a fundamental unit of logic and applications or systems should only consist of many generic modules composed together with the small amount of domain specific logic sitting atop these modules. In this chapter we’ll learn how Node’s module system works and how to create modules for various scenarios according to industry best practices. As a bonus we’ll also touch upon how to create native C modules with an exploration of the costs and benefits of doing so.

**Level:**

Relative to the overall scope of the book this chapter is dealing with mostly BASIC topics with some ADVANCED toward the end.

**Topics covered:**

* Node’s module system
* NPM’s package tooling and infrastructure
* Initializing a module
* Setting up developer tooling for module writing
* Best practices
* Decomposition
* Publishing to NPM privately

**Skills learned:**

* Implement exemplary vertical architecture within their system and/or application
* Create industry preferred module API’s according to a functional programming style
* Think in terms of modularity that forms an approach preferring composition over inheritance, and decomposition over distention.
* Understand and leverage npm tooling and infrastructure
* Set up a comprehensive module development environment
* Write a module

# **2: COORDINATING I/O (35 pages)**

**Description:** While browser-side JavaScript abstracts most input/output operations away to higher level API’s, I/O control is a crucial concept when it comes to network-level processes and command line tools. This is what Node brings to JavaScript – the ability to interact with the file system and network sockets. In this chapter we’ll explore how to read and write from the file system and the user terminal, set up basic network servers and client.

**Level:**

Relative to the overall scope of the book this chapter is dealing with MEDIUM topics.

**Topics covered:**

* File reading, writing, appending
* File system meta data
* STDOUT, STDIN, STDERR
* Creating TCP and UDP servers and clients

**Skills learned:**

* Communicate with the terminal interface
* Understand various ways to take data in and out of a process
* Set up different kinds of servers, clients and peers at a low level
* Familiarization with the file system API

# **3: USING STREAMS (50 pages)**

**Description:** JavaScript was not originally built for high intensity in-memory data processing. This is why streams are a fundamental essential in the tool kit of a Node developer. Streams allow us to process data in pieces, instead of loading an entire data set into memory first. In this chapter we’ll explain why streams are so important, how they bring functional programming to an asynchronous world and how to avoid problems with core stream API’s.

**Level:**

Relative to the overall scope of the book this chapter is dealing with MEDIUM and ADVANCED topics.

**Topics covered:**

* Idiomatic approach to stream creation
* Best practices for connecting stream inputs and outputs
* Avoiding memory leaks
* Error handling in pipelines
* Reliably detecting when a stream has ended

**Skills learned:**

* Create streams
* Create pipelines
* Manage memory efficiently
* Use streams in the most optimal fashion
* Be proficient at error handling in streams

# **4: MAKING CLIENTS AND SERVERS (50 pages)**

**Description:** Most traditional network topologies tend to have clients and servers. In this chapter we’ll show how Node can be used to create different types of severs, and how it can become a client to another server.

**Level:**

Relative to the overall scope of the book this chapter is dealing with BASIC and MEDIUM topics.

**Topics covered:**

* Making an HTTP request
* Creating an HTTP server
* Processing an HTTP POST request
* Handling a file upload over HTTP
* Creating a RESTful API server
* Creating an SMTP server
* Creating a WebSocket server-client application

**Skills learned:**

* Use lower level core API’s to interact with common protocols like HTTP
* Use third party packages to implement a real-time WebSocket solution
* Rapidly scaffold a RESTful service

# **5: WORKING WITH DATABASES (45 pages)**

**Description:** In any system, databases are essential for persisting data. The type of data inevitably influences the choices of databases in use. In this chapter we work through a cross section of database systems and find out how to talk to them from a Node a process.

**Level:**

Relative to the overall scope of the book this chapter is dealing with BASIC and MEDIUM topics.

**Topics covered:**

* Connecting and sending SQL to a MySQL server
* Populating and querying a Postgres database
* Storing and retrieving data with MongoDB
* Storing data and retrieving data with CouchDB
* Storing and retrieving data with Redis
* Implementing PubSub with Redis
* Persisting with LevelDB
* Recording to the InfluxDB time series database

**Skills learned:**

* Use various types of databases from Node
* Gain an overview of the different database options and how they suit particular scenarios
* Discover the best modules to use with different databases

# **6: WEILDING EXPRESS (60 pages)**

**Description:** Express is an often-used web framework in the Node world. It builds on HTTP server primitives supplied by core to provide an API that leads to easily composable behaviors and routes. In this chapter we will discover how to use Express for several common use cases.

**Level:**

Relative to the overall scope of the book this chapter is dealing with MEDIUM topics.

**Topics covered:**

* Generating Express scaffolding
* Managing server tier environments
* Implementing dynamic routing
* Templating in Express
* CSS preprocessors with Express
* Initializing and using a session

**Skills learned:**

* Use the popular express web framework to rapidly build web servers
* Build Express middleware
* Create sessions in Express
* Understanding how to perform CSS preprocessing with Express

# **7: GETTING HAPI (30 pages)**

**Description:** Hapi is an enterprise grade web framework for Node.js. Its ability to compose servers from plugins as well as the established request lifecycle is one of the reasons why Hapi is growing in popularity.  In this chapter we will explore Hapi and it’s ecosystem

**Level:**

Relative to the overall scope of the book this chapter is dealing with INTERMEDIATE topics.

**Topics covered:**

* Simple server creation with environment variables
* Plugin creation
* Routing in plugins
* Serving static assets
* Serving templates
* Setting up an OAuth consumer
* Adding route validation
* Testing routes

**Skills learned:**

* Getting started with Hapi.
* Using the plugin architecture.
* Authentication and validation.

# **8: UNDERSTANDING MICROSERVICES (40 pages)**

**Description:** With its event driven paradigm, focus on network I/O and tendency to fail quickly Node is well suited to the microservice architecture. In this chapter we learn what microservices are, how they inherently facilitate the scaling up of robust production systems, and what’s available in the ecosystem to assist in microservice development.

**Level:**

Relative to the overall scope of the book this chapter is dealing with MEDIUM and ADVANCED topics.

**Topics covered:**

* Breaking out event-loop blocking code into a separate process
* Separating domain logic into several processes
* Using different transports mechanisms with a microservices deployment
* Running multiple service instances
* Setting up monitoring

**Skills learned:**

* Know when and when not to use microservices
* Understand how to split a system into independent pieces
* Use an approach that’s decoupled from transport implementation
* Manage a microservices system

# **9: DEBUGGING SYSTEMS (55 pages)**

**Description:** While languages such as Java have tremendous (though often bloated) debugging tools and IDE’s the JavaScript world has always been less equipped when it comes to debug tools. This is down to a variety of factors: technical challenges with a highly dynamic language, developer preference for small composable pieces even in tooling environments, historically lower enterprise interest. In this chapter we’ll explore some excellent debugging tools for Node, along with techniques and practices to increase visibility and process information as we encounter debugging scenarios.

**Level:**

Relative to the overall scope of the book this chapter is dealing with MEDIUM and ADVANCED topics.

**Topics covered:**

* Turning on internal debug flags
* Instrumenting code with debug logs
* Enhancing stack traces
* Improve stack trace presentation
* Tracing all asynchronous operations
* Using Node debugger
* Using Node Inspector
* Tips and Tricks
* Postmortems

**Skills learned:**

* Advanced debugging techniques
* Knowledge of best-of-breed debugging libraries and tools
* Get longer/asynchronous/higher detail stack traces
* Manipulate and decorate stack trace output
* Access (very) low-level API’s to trace, record and understand asynchronous activity
* Proficiency in Node debugger
* Proficiency in Node Inspector

# **10: DEALING WITH SECURITY (35 pages)**

**Description:** It goes without saying that security is of paramount importance. In this chapter we’ll cover various attacks that can be made against a system and show common programmer errors that lead to vulnerable systems along with some best practices and approaches that help to create secure systems. We’ll also discuss authentication, HTTPS and cryptography.

**Level:**

Relative to the overall scope of the book this chapter is dealing with MEDIUM and ADVANCED topics.

**Topics covered:**

* Injection
* Stress attacks
* Passive attacks
* Vulnerable configurations
* Path traversal
* Unicode exploits
* Escape sequences
* XSS, CSRF, DoS/DDoS
* DOR programmer error
* Server hardening
* Dependency auditing

**Skills learned:**

* Avoid common pitfalls
* Understand how the limitations of the language and/or certain databases can lead to vulnerabilities
* Know how certain powerful language features can also be extremely dangerous, particular in a server-side environment
* Build more secure systems

# **11: OPTIMIZING PERFORMANCE (60 pages)**

**Description:** We should write code first for humans then computers. Nevertheless, when we have a bottleneck the reality soon sinks in that we will need to optimize for performance over clarity. In this chapter we demonstrate how to identify bottlenecks, refactor for performance and develop habits for writing efficient, optimizable JavaScript as an every day practice.

**Level:**

Relative to the overall scope of the book is this chapter dealing with ADVANCED topics.

**Topics covered:**

* Benchmarking
* Generating load
* Leak detection
* Garbage collection
* Heap analysis
* CPU profiling
* Understanding and responding to v8 inlining, deoptimization and optimization output

**Skills learned:**

* Benchmark a server
* Profile a Node process
* Detect leaks
* Effectively control garbage collection
* Determine memory consumption and allocation
* Determine CPU usage of individual function calls
* Profile the v8 JavaScript engine
* Identify bottlenecks
* Develop optimal practices

# **12: DEPLOYING SYSTEMS (50 pages)**

**Description:** This is what it’s all about, taking our efforts to the world by going live. In this chapter we explore the benefits of containerization, look at creating production builds, crash recovery, alert systems, continuous deployment, load balancing, reverse proxying, deploying single services to a PaaS and deploying microservice systems to a cloud platform.

**Level:**

Relative to the overall scope of the book this chapter is dealing with ADVANCED topics.

**Topics covered:**

* Containerization with Docker
* Continuous deployment
* AWS
* Heroku

**Skills learned:**

* Small scale deployment
* Large scale deployment
* CI work flow set up
* AWS administration basics
* Heroku deployment basics