**NODE.JS & EXPRESS.JS**

**Node.js: The Big Picture**

**Considering Node.js**

Where Is Node.js Commonly Found?

What Makes up Node.js?

A Brief History / When Node May Not Be the Best Fit

**Thinking Asynchronously**

Node's Event Loop

Asynchronous Development

Node APIs and EventEmitters

Streams

**Defining an Application and Managing Dependencies**

Modularizing Your Application

npm and Application Dependencies

**Assembling a Development Toolset**

Installing Node / Testing Applications

Debugging / Conclusion

**Node.js: Getting Started**

**Introduction**

What Is Node?

Why Node?

Some Analogies About Node

What You Get When You Install Node

Example Files

**Getting Started with Node**

Node’s REPL Mode

TAB and Underscore

Executing Scripts

Working with Timers

Node’s Command Line Interface

The “process” Object

**Modern JavaScript**

EcmaScript and TC39

Variables and Block Scopes

Arrow Functions

Object Literals

Destructuring and Rest/Spread

Template Strings

Classes

Promises and Async/Await

**NPM: Node Package Manager**

What Exactly Is NPM?

The NPM Command

The package.json and package-lock.json Files

Semantic Versioning (SemVer)

Installing and Using NPM Packages

Creating and Publishing an NPM Package

NPX and the NPM Run Scripts

Updating NPM Packages

**Modules and Concurrency**

Defining and Using Node Modules

Examples of Module APIs

Node's Global Object

The Event Loop

Errors vs. Exceptions

Node Clusters

Node’s Asynchronous Patterns

Event Emitters

**Working with Web Servers**

Hello World... The Node’s Version

Monitoring Files for Changes

The “req” and “res” Objects

Node Web Frameworks

Using Template Languages

**Working with the Operating System**

The os Module

The fs Module

The child\_process Module

Debugging Node Applications

Course

**Learning To Program - Part 1:** **Getting Started**

**Learning To Program - An Introduction**

JavaScript

Installing NodeJS (Windows)

Installing NodeJS (Mac)

The REPL

The Lunch Variable

Getting The Most

**Learning To Program - Editors and Code Files**

Text Editors

Smart Editors

Syntax Rules

When To Panic

Check Please

**Learning To Program - Variables and Types**

I Declare

Types

Functions

**Learning To Program - Functions**

Roll The Dice

Definitions

Parameters

Scope

**Learning To Program - Branch And Loop**

If The Shoe Fits

Operators

If Variations

While Do While

For Loops

Arrays Revisited

**Learning To Program - Objects**

Why Objects?

Objects In the REPL

Build Your Own

This Responsibility

Splitting Up

**Learning To Program - Tests**

Setting Up

First Test

Second Test

An Average Program

**Learning To Program - Final**

Letter Grade

ExpressJS

Web Grades

**NPM Playbook**

**NPM Basics**

What Is a Package?

Typical NPM Usage

NPM Help

NPM Shortcuts

Creating a Package.json

Setting Defaults

Installing Packages

Listing Installed Packages

Installing Global Packages

Removing a Package

Installing Specific Versions

Installing Existing Dependencies

Updating Packages

**Advanced NPM**

Installing from a Git Repo

Installing from a Gist

Installing from a Folder

The NPM Registry

Searching for Packages

Pruning

The NPM Repo Command

Upgrading NPM

Using Simple Scripts

**Publishing Your Own Package**

Setting up Your NPM User for Publishing

Preparing Your Project for Publishing

Publishing Your Package

Publishing an Update

Releasing a Beta Version

**Building a JavaScript**

**Development Environment**

**You Need a Starter Kit**

A Starter Kit Is an Automated Checklist

Who Is This Course For?

What Belongs in Your Starter Kit?

Set Up Github

Agenda

**Editors and Configuration**

What To Look For in a JavaScript Editor

JavaScript Editor Recommendations

Editorconfig

Demo: Editorconfig

**Package Management**

Package Managers

Demo: Install Node and npm Packages

Package Security

Demo: Node Security Platform

**Development Web Server**

Development Web Servers

Demo: Set up Express

Sharing Work-in-progress

Demo: Sharing Work-in-progress

**Automation**

Automation Options

Demo: npm Scripts

Demo: Pre/Post Hooks

Demo: Create Security Check and Share Scripts

Demo: Concurrent Tasks

**Transpiling**

JavaScript Versions

Transpilers

Babel Configuration

Transpiling Build Scripts

Demo: Set Up Babel

**Bundling**

Module Formats

Why ES6 Modules?

Choosing a Bundler

Demo: Configuring Webpack

Demo: Configure Webpack with Express

Demo: Create App Entry Point

Demo: Handling CSS with Webpack

Sourcemaps

Demo: Debugging via Sourcemaps

**Linting**

Why Lint?

Linters

ESLint Configuration Decisions Overview

Decision 1: Configuration File Format

Decision 2: Which Rules?

Decision 3: Warnings or Errors?

Decision 4: Plugins?

Decision 5: Preset

Watching Files with ESLint

Linting Experimental Features

Why Lint Via an Automated Build?

Demo: ESLint Set Up

Demo: Watching Files

**Testing and Continuous Integration**

Test Decisions Overview

Decision 1: Testing Framework

Decision 2: Assertion Libraries

Decision 3: Helper Libraries

Decision 4: Where To Run Tests

Decision 5: Where Do Test Files Belong?

Decision 6: When Should Tests Run?

Demo: Testing Setup

Demo: DOM Testing

Demo: Watching Tests

Why Continuous Integration?

What Does Continuous Integration Do?

Choosing a CI Server

Demo: Travis CI

Demo: Appveyor

**HTTP Calls**

HTTP Call Approaches

Centralizing HTTP Requests

Demo: Fetch

Selective Polyfilling

Why Mock HTTP?

How to Mock HTTP

Our Plan for Mocking

Mocking Libraries

Demo: Creating a Mock API Data Schema

Demo: Generating Mock Data

Demo: Serving Mock Data via JSON Server

Demo: Manipulating Data via JSON Server

**Project Structure**

Why a Demo App?

Tip 1: JS Belongs in a .js File

Tip 2: Consider Organizing by Feature

Tip 3: Extract Logic to POJOs

**Production Build**

Minification and Sourcemaps

Demo: Production Webpack Configuration with Minification

Demo: Configure Local /dist Server

Demo: Toggle Mock API

Demo: Production Build npm Scripts

Dynamic HTML Generation

Demo: Dynamic HTML Generation

Bundle Splitting

Demo: Bundle Splitting

Cache Busting

Demo: Cache Busting

Demo: Extract and Minify CSS

Error Logging

Demo: Error Logging

Demo: HTML Templates via EmbeddedJS

**Production Deploy**

Separating the UI and API

Cloud Hosting

Demo: Automated API Deploy via Heroku

Demo: Automated UI Deploy via Surge

Starter Kit Update Approaches

Option 1: Yeoman

Option 2: Github

Option 3: npm

Inspiration

Challenge

**Node.js: Introduction to** **Node.js, Events, and Streams**

**Intro to Node.js**

Hello You

Convert Blocking

Read File in Server

Writing Response Headers

Response End

**Events**

Events

Chat Emitter

Emitting Events

Request Event

Listening Twice

Listening for Close

**Streams**

Streams

File Piping

Fixing Pipe

Download Server

Additional Learning

External Links

NodeJS API Docs

npm Registry

**Node.js: Using Modules and npm**

**Modules**

Modules

Missing Exports

Export A Function

Exporting An Object

Dependency

Semantic Versioning

**Building Web Applications with Node.js and Express 4.0 (UPDATE)**

**Getting Started**

Introduction and Installation

What Is Node?

Installing Node

Dealing with Node Versions

Command Line Node

Picking an IDE

Node Package Manager

Versioning Packages

**First Page**

Setting up Express

Running Express

Debugging Options

Serving Index

Static Files - CDN

Static Files - Public Directory

Static Files - Node Modules

**Setting up Tooling**

NPM Scripts

ESLint

AirBNB Style Guide

Running ESLint

Global vs. Local

ES6

Refactoring Tips in VS Code

Setting up Nodemon

**Templating Engines**

Pug

Pug and Javascript

EJS

Templates

Routing

Navigation

**Routing**

Rendering

Separate Files

Single Book Route

Rendering a Single Book

Router Functions

**Databases**

Creating a Database

Connecting to SQL

Using Async

Input Parameters

Middleware

MongoDB

Admin Routes

Inserting Books

Testing Insert

Select Many

Select One

**Authentication**

Sign Up

Auth Routes

Passport

Local Strategy

Creating a User

Sign In

Validating User

Authorizing Users

**Structure and Third-party APIs**

Controllers

Goodreads API

Services

Making API Calls

Passing the ID

**Node.js: Introduction to Using Express.js**

**Express**

Express Routes

Route Params

Rendering

URL Building

Doing the Request

Express Server

**RESTful Web Services with Node.js and Express**

**What Is REST?**

Uniform Interface

Setting up Your Environment

Setting up Some Tooling

**Getting Data**

Implementing HTTP GET

Wiring up to MongoDB

Filtering with a Query String

Getting a Single Item

**Posting Data**

Parsing POST Data with Body Parser

Testing with Postman

Saving Data

Code Cleanup

Injecting the Book Model

**Updating Data**

Implementing PUT

Testing PUT

Middleware

Implementing PATCH

Testing PATCH

Implementing DELETE

**Testing**

Controllers

Postman and Bugs

Testing with Mocha

Running Tests

Integration Tests

**HATEOAS**

Navigating Your API

Adding Hypermedia

**Build a banking portal with Node & Express**

Setup

Use Express to create server

File handling & routing

Handling form data

Creating a data access library

Using the express router

**Node.js Testing Strategies**

**Installations**

Optional Installations

Setting up Webstorm

**Simple Tests with Node.js and Mocha**

The Membership Application

The Membership Application, Part 2

The Sad Path

Testing Dates with Moment

Testing Process Behaviors

SinonJS Spy

Using Async

**Data Access Considerations**

A Look at Node.js ORMs

Stubbing with SinonJS

Setting up a Repository

Making Things Easier with Helpers

When Helpers Hurt

Listening to Your Tools

Finishing up Review Process

**External APIs**

A Look at Stripe

Making a Test Call

The Billing Process

Stubbing Stripe

Handling Stripe Failure

Mocks with SinonJS

Simplifying Things with Nock

Involving the Database

Summary and Parting Thoughts

**Practical Cryptography in Node.js**

**Introduction to Cryptography in Node.js**

Mapping out the Course

Taking a Tour of Node.js Crypto Library

Reviewing the Learning Environment

**Protecting Passwords**

The Problem with Passwords

What Is Hashing?

Protecting Passwords the Naive Way

Comparing Password Protection Techniques

How Does a Salt Work?

Protecting Passwords the Right Way

**Protecting Data at Rest**

Threats to Data at Rest

Using Symmetric Encryption to Protect Data

Encrypting Data with Node.js

Storing Keys Safely

Using Vault to Store Keys and Encrypt

**Protecting Data in Transit**

Threats to Data in Transit

Combating Threats to Data in Transit

Asymmetric Encryption with Node.js

Exploring the HTTPS Handshake

**Authenticating Using Two-factor Authentication**

What Is a Factor?

Why a Second Factor Is Necessary

Adding 2FA to a Node.js Application

**Implementing Cryptography**

Implementing Cryptography in a Real Application

Fixing Authentication in NodeGoat

Fixing Encryption in NodeGoat

**Node Application Patterns**

**Introduction**

WebStorm

Grunt

Command Line

Web Frameworks

**Setting Up Your Project**

Setting Up Grunt

Understanding Modules

Organization

Choosing a Test Framework

Flexing Git and Github

**Building a Registration Module**

The User

A Pattern Discussion

Defining a Service

Dealing With Data

Saving the User Record

Pyramid of Doom

The EventEmitter

**Building an Authentication Module at Full Speed**

Again, in Real Time

Debugging Evented Code

**Integrating Our Module into a Web App**

Creating an API With Index.js

Plugging into a Web App

Summary and Goodbye

**Advanced Node.js**

**Node != JavaScript**

Is This Course for You?

Node's Architecture: V8 and libuv

Node's CLI and REPL

Global Object, Process, and Buffer

How require() Actually Works

JSON and C++ Addons

Wrapping and Caching Modules

Know Your NPM

**Concurrency Model and Event Loop**

What Is I/O Anyway

The Event Loop

The Call Stack

Handling Slow Operations

How Callbacks Actually Work

setImmediate and process.nextTick

**Node's Event-driven Architecture**

Callbacks, Promises, and Async/Await

Event Emitter

Arguments, Errors, and Order of Listeners

Practical Example: Task List Manager

**Node for Networking**

TCP Networking with the Net Module

Working with Multiple Sockets

Improving the Chat Server

The DNS Module

UDP Datagram Sockets

UDP Summary

**Node for Web**

The Basic Streaming HTTP Server

Working with HTTPS

Requesting HTTP/HTTPS Data

Working with Routes

Parsing URLs and Query Strings

**Node's Common Built-in Libraries**

Working with the Operating System

Working with the File System

Console and Utilities

Debugging Node.js Applications

**Working with Streams**

Stream All the Things!

Streams 101

Implementing Readable and Writable Streams

Duplex Streams and Transform Streams

**Clusters and Child Processes**

Scaling Node.js Applications

Child Processes Events and Standard IO

The Shell Snytax, exec(), and execFile()

The fork() Function

The Cluster Module

Load-balancing an HTTP Server

Broadcasting Messages to All Workers

Availability and Zero-downtime Restarts

Shared State and Sticky Load Balancing

**Node.js: Creating a Basic Node.js App with Socket.io and Redis**

**Socket.io**

Socket.io

Setting Up socket.io Server-Side

Client socket.io Setup

Listening For Questions

Broadcasting Questions

Saving Client Data

Answering Questions

Answering Question Client

**Persisting Data**

Persisting Data

Simple Redis Commands

Get A Key

Working With Lists 1

Working With Lists 2

Persisting Questions

Emitting Stored Questions

Limiting Questions Stored

**Introduction to the Web Audio API**

**Hello World (Creating Your First Sound)**

Creating an Audio Context Instance

Creating an Oscillator

Stopping the Oscillator

Properties of Oscillators

Adjusting the Volume of an Oscillator

Overlapping Two Oscillators

**Filters and the Node Graph**

Filter Types

Connecting a Filter

Bandwidth

Connecting an Additional Filter

**Loading and Playing a Sound**

Web Audio API Benefits Over HTML5 Audio Element

Asynchronous Loading of Audio Files Explained

Completing the Audio File Buffer Loader

Writing the Code to Play Back the Audio Buffer

**Abstracting Audio File Loading & Playback**

Abstracting File Load and Playback

Adding Parameters to Allow Customization of Sound Playback

What You Should Know

**Adding Reverb Effects**

Using the Convolver Object

Creating Gain Node Objects to Route Our Signals

What You Should Know

**The Delay Node**

Creating an Echo Delay

Filtered Effect

What You Should Know

**Dynamics Compression**

Parameters of the Dynamics Compressor Node

Connecting the Dynamics Compressor Node

Fixing the Meter Error

What You Should Know

**The Panner Node**

Creating an Equal Power Panner

dopplerFactor Property

Overview of the Panner App

What You Should Know

**Time**

Problem of Changing Schedule Events

Creating a Small Drum Machine Application

Creating the User Interface

**Changing Parameter Values Over Time**

Changing Parameter Values Over Time Abruptly

Changing Parameter Values Over Time More Smoothly

What You Should Know

**Integrating HTML5 Audio and Video Elements**

Integrating HTML5 Audio and Video Elements

What You Should Know

**Adding UI Elements**

Introduction

jQuery Knobs

What You Should Know

**HTTP with Node.js**

**Managing HTTP Requests**

Creating Your First Server

The Anatomy of the HTTP Module

Working with the HTTP Method, Headers, and URL

Working with the Request Body

The “body” Package

How About HTTPS?

**Managing HTTP Responses**

Working with Status Codes and Headers

Responding with JSON

The Benefits of Streamed Data

Handling Errors

**Handling File Uploads with Formidable**

What Is Formidable?

Handling File Uploads with a Callback

Configuring Formidable

Handling File Uploads with Events

**Making External API Requests**

Making a GET Request

Making a POST/PUT/DELETE Request

Working with Auth Headers

The “request” Package

**Managing Files with Node.js**

**Reading an Entire File**

Reading an Entire File

Reading a File Asynchronously

Handling Asynchronous Errors

Reading a File Synchronously

Read Asynchronously without Callbacks

**Reading Parts of a File**

Reading Parts of a File

Opening a File to Read

Reading a Chunk at a Time

Reading Parts Synchronously

Importance of Closing

**Writing to a File**

Writing to a File

Writing an Entire File

Appending to a File

Utilizing Options

Writing Parts of a File

Watching Files

**Reading and Writing to Streams**

Reading and Writing Streams

Reading Data from a Stream

Pausing a Stream

Writing to a Stream

Handling Mismatched Streams

**IoT Full Stack Development:** **Getting Started**

**Choosing the Right Hardware Platform**

Connectivity Tradeoffs and Naming Conventions

Cost Tradeoffs and Latency

Faster Development and Testing

Security and Your Scenario Project

OS Installation and Security

VPN Config and Key Generation

**Off-the-shelf vs. Custom and Battery Life**

Prototype Early and Often

Measure Energy Early and Often

Batteries in the Real World

Saltstack Cluster Setup

IoT Cluster Automation with Salt

**Security for Every Link and Node**

Credentials, Firewalls, and VPNs

HTTPS and Alternatives

Demo: Protocol Overhead on Mobile Networks

HTTPS, VPNs, and Latency

Gateways for Low Power Devices

**User Interfaces and Back-end Platforms**

Interface Pitfalls and UX

Databases and Data Paths

Platform Comparisons

**AI, Automation, and Interaction**

Real-time or Historical

AI and Machine Learning On-premise

Set up the MQTT System

Cloud Communication Latency

**Real World Considerations**

Future-proofing IoT Systems

Secure Upgrades and Crypto

Key Rotation and Thorough Testing

Counting Electrons to Maximize Battery Life

Supercluster Setup for Latency Test

Coding up the Latency Test

Real-world Timing Demo

**Analytics and Serverless Operation**

From Server to Serverless

Simulating Devices, GDPR, and Privacy

Machine Learning Here and There

TensorFlow Quick Setup

Other Routes to Smarter Devices

**System Extensibility and Portability**

Cloud Service Evolution

Secure Firmware Updates

Your IoT Future

**Unit Testing with Node.js**

**Introduction**

Course Repository

Why Testing Is Important

Introduction to Unit Testing

**Tools for Unit Testing Node**

Introduction to Mocha and Chai

Setup and Teardown

More Mocha Features

Sinon and Spies

Sinon and Stubs

Sinon and Mocks

Using Gulp to Test Node

Using Grunt to Test Node

**Unit Testing Your Code**

Testing Business Logic

Testing Errors

Finding Edge Cases

IO and Asynchronous Operations

Testing with Mocks

Testing Promises

Test Coverage

**Introduction to Sequelize ORM for Node.js**

**Features of Sequelize**

Introduction and Overview

Prerequisites

What Is Sequelize?

SQL Database Explained

Sequelize Uses Promises

Benefits of Sequelize

**Sequelize Setup**

SQLite Installation

Project Setup

Connecting to Database

Models

Sync Database

Test with Data

Data Types

Primary Keys

**Customize and Validate**

Customize Model

Attribute Qualifiers

Validations

Hooks

Review

**Performing CRUD Operations**

Add Sample Data

Create Method

Find All and Filter

Find One Entry

Update Entry

Delete Entry

**Associations**

Associations Explained

Add belongsTo() Association

Foreign Keys

Model Alias

Types of Associations

One-to-Many Association

Many-to-Many Association

Many-to-Many Example

**Demo Application Using Angular**

Project Setup

Database Setup

Create Models

Seed Database

Show All Books

Single User Route

Save User Book

Display User Books

Save User Favorite

Add New User

**Securing Your Node.js Web App**

**Introduction to Hackers Hall**

**Proper User Authentication**

Overview

The Problem with Password Storage

Introduction to Bcrypt

Implementing Bcrypt

Password Strength, the Missing Ingredient

Enforcing Validation Rules with Express-validator

Enforcing Validation at the Database level with Mongoose

Brute-force Safeguards

Brute-force Mitigation with Delayed Responses

Tracking Failed Logins

Mitigating Parallel Brute-force Attacks

Transport Layer Security

**Session Management**

Overview

The Problem with Session ID's

Protecting the Session ID

Time-limited Sessions

Session Fixation

Regenerating Sessions on Authentication

The Problem with Session Cookies

Protecting Session Cookies with the HTTPOnly Flag

Using Transport Layer Security to Secure Session Cookies

The Problem with Mixed Content

Protecting Session Cookies with the Secure Flag

Re-authorization on Key Access Areas

**Securing MongoDB from Injection Attacks**

Overview

SQL Injection Attacks

Injection Demonstration with Burp

NoSQL and the Risk of Injection Attacks

MongoDB Injection Attacks

MongoDB and the Risk of JavaScript Expressions

**Handling Untrusted Data**

Handling Untrusted Data

Overview

Fuzzing Data with Zed Attack Proxy

Identifying Untrusted Data

Where and When to Handle Untrusted Data

Whitelist Versus Blacklist Approaches

Validating Untrusted Data

Escaping Untrusted Data

Why Sanitizing Isn't So Sanitary

**Access Controls**

Overview

Principle of Least Privilege

The Problem with Database Access

Overview of MongoDB Access Control

Implementing MongoDB Access Control

Role Based Access Control

Function Level Controls (with RBAC)

Server-side Function Level Control Failure

Access Control Misconfiguration

**Defending Against Cross-site Scripting (XSS)**

Overview

Demo: Cross-site Scripting

Identifying XSS with Netsparker

Anatomy of Cross-site Scripting Attack

Reflective Cross-site Scripting

Persistent Cross-site Scripting

DOM Based Cross-site Scripting

Introduction to Content Security Policies

Implementing Content Security Policies

Enabling Cross-site Scripting Protection Filter

Cookies Protection

Escaping Untrusted Data

Sanitizing and Validation of Untrusted Data

**Securing Your Connection**

Overview

Acronym Soup: TLS, SSL, and HTTPS

The Importance of TLS

Setting up a Secure Server

Login Forms from the Top

Introducing to HTTP Strict Transport Security

Implementing the HSTS Header

Introducing to Content Security Policy

Implementing the CSP Header

**Building Command Line Applications in Node.js**

**Setting up Your CLI Project**

A Brief Overview of Command Line Applications

Demo: Registering Your Sample Application with Twitter

What Makes a Node.js Project a Command Line Application?

Demo: Initializing Your Node.js Project

**Managing Configuration**

A Brief Overview of Configuration Information

Demo: Storing the Twitter API Key and Secret in Your Project

Mocking and Testing Command Line Applications

Demo: Adding Tests to Your Sample Project

Handling Sensitive Configuration Data

Demo: Adding Your Twitter API Secret to the Keychain

**Interacting with the User**

CLI Option and Command Patterns

Demo: Adding Commands to Your Project, with Test Coverage

Implementing Twitter's PIN-based OAuth

Demo: Adding Another Command

Demo: Adding Another Command, Part 2

Demo: Using the Node Debugger

**Interacting with the Environment**

Handling Errors and Setting an Exit Status

Demo: Add Error Handling and Proper Exit Statuses

A Pattern for Environment Variable Overrides

Demo: Implementing Environment Overrides for Credentials

File Descriptors and Dealing with Unbounded Input

Demo: Lookup Users via Standard In or Command Line

Demo: Piping from Your CLI to JQ and Back Again

Future Integration Opportunities and Summary

**Packaging and Distribution**

Scoped Packages, Publishing to npm, and Using npx

Demo: Publishing Your CLI to npm, Running It with npx

Adding Update Notifications, Travis CI Automation

Demo: Automating Your npm Publish with GitHub and Travis CI

Creating a Docker Image of Your Command Line Application

Demo: Creating a Docker Image, Updating Your Travis Automation

**API Design in Node.js Featuring Express & Mongo**

**Node.js**

Node.js Refresher

CommonJS

Executing Node

Express

Using Express

Exercise 1

Exercise 1 Solution

**REST APIs**

Getting RESTful

Anatomy of a REST API

Exercise 2

Exercise 2 Solution

Exercise 2 Solution, Continued

**Middleware**

What Is Middleware?

Types of Middleware

Middleware Examples

Exercise 3

Exercise 3 Solution

Routers

Exercise 4

Exercise 4 Solution

Error Handling

Audience Q&A

Audience Q&A, Continued

**Testing**

Testing in Node

Node Environment Variables

Exercise 5

Exercise 5 Solution

Testing Q&A

**Organization and Configuration**

Application Organization

Configuration

Exercise 6

Exercise 6 Solution

**Mongo**

Mongo Introduction

Using Mongo with Node

Exercise 7

Exercise 7 Solution

**Data Modeling**

Schemas

Schema Types

ObjectId

Blog Schema Representation

Exercise 8

Exercise 8 Solution

**Querying**

Querying Data with Mongoose

Populations

Exercise 9

Exercise 9 Solution

Exercise 9 Solution, Continued

Creating Promises

Consuming Promises

Nested Promises

**Authentication**

JSON Web Tokens

Using JWT

Usernames and Passwords

Authentication with Middleware

Exercise 10

Exercise 10 Solution

Executing CRUD Operations

Authentication Configuration

Exercise 11

Exercise 11 Solution

Testing the Authentication

**Securing Routes**

Identifying Sensitive Routes

Understanding CORS

Testing the UI

Exercise 12

Exercise 12 Solution

Exercise 12 Solution, Continued

**Deployment**

Deployment Overview

Deploying to Heroku

Configuring the Deployment

**Q&A**

Q&A Part 1

Q&A Part 2

Q&A Part 3

**Using the Serverless Framework with Node.js on AWS**

**The State of Serverless**

Serverless Context - Why Do You Need Frameworks?

Why Use the Serverless Framework?

Key Course Concepts

Planning Course Projects

**Up and Running with the Serverless Framework on AWS**

Module Concepts and Installation Demo Outline

Getting Started with AWS

Installing the Serverless Framework on Windows

Installing the Serverless Framework on OS X

Installing the Serverless Framework on Ubuntu

The Serverless Framework Application Model

Developing Serverless Services

Creating Your First Serverless Framework Service

**Deploying a Text Notification Website with a Serverless HTTP API**

Module Overview and Serverless Plugins

Exploring Serverless APIs

Creating Your Text Notification Website

**Scaffolding and Deploying Serverless CRUD APIs**

DynamoDB Overview and Resource Configuration

Configuring HTTP CRUD APIs with the Serverless Framework

Reviewing Your API Function Code

Deploying Your CRUD API

Essential Debugging Tools and Resources

**What Is Multi-platform Serverless?**

IaaS Outages and Risks

Multi-platform Architectures

Other Resilient Architectures

**Code School: Real-time Web with Node.js**

Intro to Node.js

Events

Streams

Modules

Express

Socket.io

Persisting Data

**Building AngularJS and Node.js Apps with the MEAN Stack**

Introduction

Prerequisites

Operating System

The Multivision Application

Versions

Course Updates

**Creating a Walking Skeleton**

Introduction

Creating the package.json file

Adding Git

Installing Clientside Dependencies with Bower

Creating the Node Application

Adding Stylus

Creating a Layout

Creating the Angular Application

Adding MongoDB

Displaying Data in a View

**Deploying to Heroku**

Preparing for Heroku

Heroku and MongoDB

Pushing to Heroku

General Heroku Troubleshooting

Managing and Troubleshooting Heroku Keys

**Designing the UI**

Creating the Header and Footer

Creating the Main Page Content

Adding Data to the Main Page

**Authentication**

Adding the Login Display

Refactoring server.js

Preparing for Login

Creating the Server Login Code

Creating the Client Login Code

Improving the Client Login Code

Improving the Server Login Code

**Security and Authorization**

Implementing Secure Passwords

Adding Signout Functionality

Persisting Login between Page Refreshes

Refactoring Server Authentication Code

Implementing Clientside Authorization

Implementing Serverside Authorization

Protecting Admin Pages on the Client

**Client-side Testing**

Installing Testing Tools

Testing the mvUser Service

**Implementing Signup - Creating Entities**

Implementing the Client Code

Creating the Server-side Controller

Refactoring the Server-side Model

Implementing Model Validation

**Use Profile - Updating Data**

Creating the Display

Creating the Client-side Code

Creating the Server-side Code

**Course List - Sorting, Filtering and Caching**

Creating the Client-side Code

Creating the Server-side Code

Filtering and Sorting

Getting Main Page Data from the Server

Caching Data

**Course Details - Selecting and Caching Single Entities**

Creating the Client-side Code

Creating the Server-side Code

Getting Cached Data

**Understanding OAuth with Node.js**

**OAuth 2.0 in Action**

How the Finished Application Works

Reviewing the Finished Code

**Building the OAuth Client**

Reviewing the Client Code

Common OAuth Client Vulnerabilities

**Building the Protected Resource**

Reviewing the Protect Resource Code

Common Protected Resource Vulnerabilities

**Building the Authorization Server**

Reviewing the Authorization Server Code

Common Authorization Server Vulnerabilities

**Zero to Production Node.js on Amazon Web Services**

**The Web Server**

The Sample Application

Exercise 0: Setting up Shop

Serving HTTP Requests with Express

The Web Server Code Demo

Alternatives to Express

Exercise 1: Hacking on Express

Exercise 1 Solution

**Build Tools**

NPM Scripts and Grunt

NPM Scripts and Elastic Beanstalk

Build Tools Code Demo

Alternatives and Friends of Grunt

Exercise 2: Enhancing Your Build Tools

Exercise 2 Solution

**The Database**

Sequelize and PostgreSQL

The Database Code Demo

Alternative Databases

Exercise 3: Enhancing Your Data Model

Exercise 3 Solution, Part 1

Exercise 3 Solution, Part 2

**Production Environment**

Elastic Beanstalk and RDS

Provisioning an Environment

Production Environment Demo, Part 1

Production Environment Demo, Part 2

Production Environment Demo, Part 3

Production Environment Demo, Part 4

Alternative Production Environments

Exercise 4: Deploying the Application, Part 1

Exercise 4: Deploying the Application, Part 2

Exercise 4: Deploying the Application, Part 3

**Front End Toolchain**

Day Two Agenda

Browserify

SASS and SASS Alternatives

Browserify Code Demo

SASS Code Demo

Exercise 5: Building on Front End Tools

Exercise 5 Solution

**Vue.js**

Building with Vue.js

Vue.js Code Demo

Exercise 6: Building a Rich Client

Exercise 6 Solution, Part 1

Exercise 6 Solution, Part 2

**Realtime User Interfaces**

Using Socket.io

Chat Application Code Demo, Part 1

Chat Application Code Demo, Part 2

Exercise 7: Realtime Features

Exercise 7 Solution

**Production Monitoring**

Locust and Rollbar

Load Testing Code Demo

Exercise 8: Production Monitoring with Rollbar

Exercise 8 Solution

**Web Analytics**

Google Universal Analytics

Google Analytics: Overview and Custom Reports

Google Analytics: Goals

Universal Analytics in Node

Exercise 9: Implement Google Analytics

Exercise 9 Solution

**Mocking Node.js with Sinon**

**Spying on Functions**

What Is a Spy?

Getting Set Up

Creating a Spy

Using a Spy

Restoring Spies

Why Use Spies?

**Replacing Functions with Stubs**

Replacing Functions with Stubs

Creating a Stub

Stubbing out Exceptions

Stub Specific Calls

Stubbing Asynchronous Functions

Other Stub Functionality

**Simplifying with Fakes**

Simplifying with Fakes

Replacing a Spy with a Fake

Replacing a Stub with a Fake

How Are Fakes Different?

**Using Mocks to Isolate Code**

Using Mocks to Isolate Code

Creating a Mock

Mocking Specific Calls

Verifying a Function Wasn't Called

Other Mocking Functions

Comparing Test Doubles

**Using MongoDB with Node.js**

**Getting MongoDB Running**

Setting the Stage

Installing MongoDB

Starting Your Project

Loading Data

**Working with CRUD**

Building a Test Harness

Get

Filter

Get by ID

Add

Update

Remove

**Working with the Aggregation Pipeline**

Resources

Using Aggregation

More Complicated Aggregation

**Automating Node.js with NPM Scripts**

**Getting Started Creating NPM Scripts**

Initialize Package/Project

NPM Run

Lifecycle Scripts vs. Custom Scripts

NPM Test

NPM Start

**Scripting for Building**

CLI Tools via NPM Scripts

TypeScript: Compile with NPM Scripts

TypeScript: Passing Arguments in NPM Scripts

TypeScript: Start Script

TypeScript: Building before Running with NPM Pre Hook

Environment Variables in NPM Scripts

Using NPM Config as Environment Variables

CSS Pre-processors: Building with NPM Scripts

Bundling and Minifying: Webpack Build Tasks

Bundling and Minifying: Automating Webpack Build Tasks

Front-end Frameworks: Angular Build Tasks

Front-end Frameworks: Automating Angular Build Tasks

Containers: Docker Build Tasks

Containers: Automating Docker Build Tasks

**Scripting for Developing**

Node.js Development Tasks

Automating Node.js Development Tasks

TypeScript Development Tasks

Automating TypeScript Development Tasks

Bonus TypeScript Development Tool

Webpack Development Tasks

Automating Webpack Development Tasks

Automating CSS Pre-processor Tasks

Automating Frontend Development Tasks

Docker Development Tasks

Automating Docker Development Tasks

Automating Node.js Debugging

**Scripting for Testing**

Running Tests

Watching for Changes

Creating Coverage Reports

Opening Coverage Reports with a Post Script

**Scripting for Deploying**

NPM Scripts for Production

Build and Release Process

Using NPM Scripts in a Pipeline

**VS Code**

Getting Started

Adhoc Tasks

Build and Run as Manual Tasks

Automate Build and Run Tasks

**Testing JavaScript for Node.js with Mocha**

**Testing Your JavaScript**

Types of Testing

Node Versions

Things Change

**Testing with Mocha**

Testing Mocha

Our First Test

What Is an Assert

Testing Something Real

Scripts

Testing Async

Working with Timeouts

Hooks

Scoping in Hooks

When Hooks Go Bad

Pending Tests

**BDD Style Assertions**

Installing Chai

Expecting Behaviors

Natural Language Asserts with Should

Testing Objects

Dealing with Nulls

Working with Promises

**Spys, Stubs, and Mocks**

Setting up Get Index

Working with Spys

Watching Existing Functions

Stubbing Functions

Mocks

**Testing Real Things**

GitChecker

Setting up Tests

Mocking HTTP

Mocking Second Call

Validating Parameters

Testing the Controller

Code Coverage with Istanbul

**Data Access in Node.js Using Knex**

**Introduction**

Course Outline

What Is Knex?

Starting a Knex Project

Your First Query

Database Options

Tips for Development

**Building Queries**

Querying with Promises

Selecting Data

Object Graphs and Eager Loading

Inserts & Updates & Deletes (Oh My!)

Transactions

**Schema Building, Migrations, and Seeding**

The Movie Database

Project and CLI Setup

Schema Migrations

Seeding Data

**Real-world Data Access**

Project Setup

Querying Data

Modifying Data

**Integrating Node Applications with GitHub**

**Integrating NodeJS Applications with GitHub**

Before Beginning the Course

Why Integrate with GitHub?

Business Reasons for Integrating NodeJS Applications with GitHub

**Integrating Node Applications with Travis CI and GitHub**

What is Travis CI?

Is Continuous Integration Right for Your Project?

A Look at the App

Preparing the App for Travis

Uploading the App to GitHub

Integrating a GitHub Account with Travis

Exploring Travis Features

Adding Build Status to GitHub Repositories

Travis CI Conclusion

**Deploying GitHub Branches Automatically with Cloud-based Deployment Software (Heroku)**

What is Heroku?

Getting Set up with Heroku

Deploying Node Applications with Heroku Explained

Manual Deployment via Heroku

Integration Between Heroku & Travis

Automating App Deployment

Automatic App Deployment Demo

Manual Pull Request Review Process

Enable Automatic Review of Pull Requests with Review Apps

GitHub Heroku Summary

**Automatically Creating Issues and Managing Repos with Discussion Board Software (Trello)**

Understanding Trello

Discussion Board Software and Development

How GitHub and Trello Work Together

How Can Discussion Board Software be Integrated with GitHub?

Note on Trello Business Class

Connecting GitHub and Trello - Demo

Advantages to Trello Integration

Integrating A Pull Request with Trello - Demo

Additional Trello-GitHub Integration Demonstration

Trello Github Review

**Integrating Custom Node.js Webhooks with GitHub**

What Are Webhooks?

Webhooks and GitHub Explained

Creating a Simple Webhook in Node.js

Enabling Webhooks

Webhook Applications

Webhooks Summary

GitHub Integrations in Review

Continued Learning

**Getting Started with Node.js Security with Express and Angular**

**Introduction**

Application Security & Risk

The Role of Application Developers in Security

Prerequisites

The Vulnerable & A

ttacker Sample Applications

Running & Exploring the Vulnerable Application

**Protecting Data from Extraction**

Attack Reconnaissance

Attack Execution

Cross-site Scripting

Attack Prevention

Prevention with Input Handling

Input Handling Examples

Prevention with a Content Security Policy

Correcting in-line Styles

Correcting in-line Scripts

Content Security Policy Example

Content Security Policy Browser Support

Defense with Input Handling in Angular

Defense with Input Handling in Node.js

Defense with a Content Security Policy

Content Security Policy in Action

**Hide Network Traffic from Snooping**

Attack Reconnaissance & Execution

Man in the Middle

Client/Server Overview

Attacker Positioning

Prevention with Transport Layer Security

Transport Layer Security

Generating a Self-signed Certificate

Implementing HTTPS with a Certificate

Defense with Transport Layer Security in Place

**Ensure Legitimacy of Requests**

Attack Reconnaissance

Attack Setup

Attack Execution

Cross-site Request Forgery

Attack Prevention

Importance of Cross-site Scripting Mitigation

Prevention with HTTP Header Checks

Origin and Referer Header Example

Prevention with the Synchronizer Token Pattern

Synchronizer Token Pattern Example

Implementation of HTTP Origin and Referer Header Checks

Defense with HTTP Origin and Referer Header Checks

Defense with the Synchronizer Token Pattern

**Block Content Hijacking**

Attack Reconnaissance & Setup

Attack Execution

Clickjacking

Attack Prevention

Prevention with HTTP Headers

Browser Support

Prevention Through Denying Framing

Prevention Through Limiting Framing from the Same Origin

Prevention Through Limiting Framing to Specific Origins

High-level Topic Review

Recap of Cross-site Scripting

Recap of Man in the Middle

Recap of Cross-site Request Forgery

Recap of Clickjacking

Resources & Conclusion

**Play by Play: Build a Serverless Node Web API**

**Getting Started with Serverless**

What Is Serverless?

Getting Started

Custom Function Creation

Scenario: Migrate an Existing Node Express API to Serverless

**Explore Local Development and Create First Endpoint**

Creating a Function App Inside VSCode

Build Read Endpoint and Debug Locally

Read Heroes Using MongoDB API

Create Custom Route

Build Create Endpoint

Build Update Endpoint

Build Delete Endpoint

**Publish Function and Test Application**

Publishing Functions to the Cloud

Make Your App Work

**Deployment Options**

Version Control

Setting up Continuous Deployment

**Creating Apps With AngularJS, Node, and Token Authentication**

**Introduction**

Introduction Part 2

Life of a JWT

**Register in Front End Views**

Environment Setup

Generated Angular Project

Register View

ui-router

App Theme

ui-sref active

Register View Fixes

**Register Front End Controllers and Services**

Match Password Directive

Register Controller

HTTP Post

Alert Service

**Register WebAPI Manual JWT**

API Environment Setup

API Register Post

MongoDB Save User

Encrypt Password With Hash

Hide Password

JWT Encoding From Scratch

Token Passed With Register

Authtoken Factory

isauthenticated

**Register WebAPI JWT Library**

Logout

Jobs and Greetings

View Animations

Securing the Jobs Resource

Auth Interceptor

JWT Decoding from Scratch

Verifying the Signature

Design Break Optimize Alerts

Switching to Node JWT Simple

**Login**

Login Endpoint

Login View

Login Controller

Login Auth Service

Redirect

Register Auth Service

Passport Login

Passport Register

Duplicate Email Check

**Auth Provider**

Google Project

Window Popup

Google Authorization Code Request

Getting Authorization Code

Sending Authorization Code

Authorization Code in Main Window

Close Popup

Authorization Code to Backend

Token Exchange With Google

Google Plus API Profile

Create jwt From Google Profile

Frontend Authentication

**Satellizer**

Satellizer Google Login

Is Authenticated

Logout

Local Login

Local Register

Facebook Auth Backend

Facebook Auth Frontend

**Email Validation**

Cleanup

Email Verification Token

Template

Template Theme

Template With Underscore

Nodemailer

Endpoint Handler

Front End Check

**Authentication With Sails**

Sails Overview

Getting Jobs

Authenticating With Policy Middleware

Local Login

**Unity Multiplayer Game Development with Node**

**Introduction and Project Overview**

**Unity Multiplayer Game Development with Node**

Setting up Unity and Node for Multiplayer Development

Connecting Our First Client to the Server

Spawning Our First Player

Spawning Existing Players

Navigating Our Player

Moving Our Player with Clicks

Animating Our Player

Run Animation for Our Player

Sending a Move Request to the Server

Passing Our Move Position

Identifying Our Players

Associating Our Player Objects with ID's

Moving Other Players

Moving All Players

Disconnecting Players

Updating Position with New Clients

Updating Movement with New Clients

Interacting with Players Using Interfaces

Creating a Spawner Factory

Following Players

Sending a Follow Target to the Server

Synchronizing Player Follows

Stopping Our Player Follow

Optimizing Our JSON

Refactoring Our Network Class

Creating a Targeting Class

Attacking Players

Synchronizing Player Attacks

Implementing Health and Damage

Adding Attack Animations

Implementing Player Death Logic

Respawning Players

Improving Anti-hacking with Server Authoritative Code

Estimating Player Travel Time

Determining if a Player Is Hacking Their Position

**Supercharging Your GitHub Workflow**

**The Modern Software Development Pipeline**

Different Workflows with Git

Introduction to the GitHub Workflow

The Basic GitHub Workflow

Thinking Through Your Software Delivery Pipeline

Tools for Securing Your GitHub Workflow

The Role of Continuous Integration

The Role of Continuous Delivery and Deployment

The Role of Issues and Pull Requests

Put It All Together for a Supercharged Workflow

**The Supercharged GitHub Workflow in Action**

Enable Continuous integration

Set up Your Build with a config.yml File

Add Validations to your config.yml File

Add Branch Protections

Issue and Pull Request Templates

Use Webhooks through a GitHub App

Deploy using GitHub Pages

Customizing Your GitHub Workflow

Different Merge Strategies

The Perils of Long Running Branches

**Customizing Your Supercharged GitHub Workflow**

Considerations for Designing a Workflow

Pitfalls of a Non-InnerSource Culture

Adopting InnerSource

**Code School: Mastering GitHub**

**Fork Based Workflows**

Forking and Cloning

Submitting a Pull Request

Updating Your Fork

**Single Repository Workflows**

Collaborating on Pull Requests

Merging Pull Requests

**Tags, Branches, and Releases**

Tags

Release Branches

Releases in GitHub

**Issues, Wikis, and GitHub** Pages

Issues

Wikis

GitHub Pages

**Managing and Securing GitHub**

Configuring a New Repository

GitHub Authentication/Authorization

**Automating GitHub**

Integrating Existing Services

The GitHub API

**Command Line GitHub**

Getting Started with Hub

**GitHub Fundamentals**

**An Introduction to Git**

Module Introduction

What's in This Course for You?

Hello Git

Base Concepts of Git

Getting Your Machine Ready

Demo: Setting up a Windows Environment

Working with Git

Demo: Working with Git

**Getting Started with GitHub**

Module Introduction

Hello GitHub

Demo: A Guided Tour of GitHub

Creating Your GitHub Account

Demo: Setting up Your GitHub Account

The Base Building Block: Repositories

Demo: Creating a Repository on GitHub

Using SSH to Connect with GitHub

Demo: Using SSH

Finding Stuff on GitHub

Demo: Searching GitHub

**Working with Repositories in GitHub**

Module Introduction

Repositories in GitHub

Demo: Repositories on GitHub

Special Files on GitHub

Demo: Creating Files on GitHub

Working with GitHub Locally

Demo: Interacting with GitHub

Adding and Editing Files on GitHub

Demo: Adding and Editing Files on GitHub

Archiving Repositories

Demo: Archiving Repositories

Repository Features

Demo: Exploring the Features of a Repository

Bringing in More People

Demo: Working with Collaborators

Repository Insights

Demo: Exploring Repository Insights

**Branching, Merging, and Pull Requests in GitHub**

Module Introduction

An Overview of Branches

Demo: Creating a Branch on GitHub

Commands for Branching

Demo: Creating Files on GitHub

Pull Requests and the GitHub Flow

Demo: Working with Pull Requests

Demo: Setting the Default Branch

**Creating Tags and Releases**

Module Introduction

Tags: Locally and on GitHub

Demo: Working with Tags

Working towards a Release on GitHub

Demo: Creating a New Release on GitHub

**The Social Aspect of GitHub**

Module Introduction

Working with Forks

Demo: Working with Forks

Pull Requests Revisited

Demo: Working with Pull Requests

Using Gists

Demo: Using Gists

Creating GitHub Pages

Demo: GitHub Pages

Enabling Notifications

Demo: Working with Notifications

**Managing Your Project through GitHub**

Module Introduction

Working with Issues

Demo: Working with Issues

Creating a Milestone

Demo: Creating a Milestone

Working with Projects

Demo: Working with Projects

Setting up a Wiki

Demo: Setting up a Wiki

**Working with Organizations and Teams**

Module Introduction

Creating an Organization

Demo: Creating an Organization

Working with Teams

Demo: Setting up Teams in GitHub

**Interacting with GitHub through the GitHub Desktop Client**

Module Introduction

What Is GitHub Desktop?

Installation and Requirements

Demo: Configuring GitHub Desktop

Working with GitHub Desktop

Demo: Working with Repositories

Demo: Working with Branches

Demo: Working with Pull Requests

**GitHub for Windows Developers**

**Brief Introduction to Git**

About Git

CVC versus DVC

Basic Git Commands

Command Line Demo

**Using the GitHub Website**

Intro to GitHub

Installing Git

Working With Repositories

README, .gitignore, and Live Editing

Organizations, Teams, and Collaborators

Issues, Wikis, and Pages

**GitHub for Windows Basics**

Configuration Options

Setting Up Repositories

Local and Remote Repositories

Committing and Syncing

Reverting and Rolling Back

**Teamwork With GitHub for Windows**

What Merging and Rebasing Are

Dangerous Rebasing

Good Examples of Merging and Rebasing

Branching and Merging Demo

Stashing Changes

Resolving Conflicts

Configuring .gitattributes Files

**Social Coding With GitHub**

Forking Repositories

Pull Requests

Reporting and Managing Issues

Repository Graphs and Data

Using Gists

Profiles and Contributions

**Web Development with ExpressJS**

Introduction to node

Installation and tooling

First node application

Code organization

Writing code 'node-style'

A node Http Server

Packages

Connect Middleware

**An Introduction to Express.js**

Express.js Overview

Our first Express Application

Routing

Route parameters and queries

Configurations

Enhancing apps with middleware

**Views**

Jade View Engine

HTML Elements

Embedding JavaScript and Code

Using Blocks for Layout

Stylesheets

**Working with Data**

Displaying Data in Views

Working with Forms

Content Negotiation

Uploading Files

Serving Files

Locals and Helpers

**Common and Advanced Scenarios**

Working with cookies

Working with sessions

Custom middleware

Authentication and Authorization

Error Handling

Using express to create web apps

Using WebStorm to create web apps

Using WebStorm to run and debug web apps

**Node.js Testing Strategies**

**Introduction**

Installations

Optional Installations

Setting up Webstorm

A First Test

**Simple Tests with Node.js and Mocha**

The Membership Application

The Membership Application, Part 2

The Sad Path

Testing Dates with Moment

Testing Process Behaviors

SinonJS Spy

Using Async

**Data Access Considerations**

A Look at Node.js ORMs

Stubbing with SinonJS

Setting up a Repository

Making Things Easier with Helpers

When Helpers Hurt

Listening to Your Tools

Finishing up Review Process

**External APIs**

A Look at Stripe

Making a Test Call

The Billing Process

Stubbing Stripe

Handling Stripe Failure

Mocks with SinonJS

Simplifying Things with Nock

Involving the Database

Summary and Parting Thoughts

**SinonJS Fundamentals**

**The State of Testing at Globomantics**

Welcome to Globomantics

Installing Prerequisites

Getting the Sample Project

Path Ahead

**Spying on Functions**

Spying on Functions

What Are Spies?

Verifying a Spy Was Called

Demo: Verify a Spy Was Called

Testing Pro-tip

Verify Multiple Calls

Examining Parameters Passed to a Function

Demo: Verify Right Book Was Created

Verify Exceptions

Spies as Building Blocks

**Swapping a Stub for a Function**

Swapping a Stub for a Function

What Is a Stub?

Returning Fake Values

Handling Promises

Demo: Stubbing Resolved Values

Cleaning up After a Stub

Additional Return Options

Spies vs. Stubs

**Verifying Behavior with Mocks**

Verifying Behavior with a Mock

What Is a Mock?

Using Mocks

Checking Mocks

Demo: Verify Payment

Mocks vs. Spies vs. Stubs

**Controlling How Parameters Are Matched**

Verifying Creation Date

What Are Matchers?

What Can Be Matched?

Completing Date Verification

Array Matchers

Custom Matchers

**Faking out XHR and Timer Calls**

Making External API Calls

Running Tests in the Browser

Making a Fake Request

Demo: Fake XHR

Faking a Server

Faking a Date

Demo: Cleaning up Date Verification

Coming Up

**Easier Set up and Clean up with Sandboxes**

Easier Set Up

What Is Sandbox?

Demo: Using Sandbox

Other Sandbox Functions

**What’s Next?**

Big Picture

Additional Libraries

Final Thoughts

**Getting Started with OAuth 2.0**

**API Security 101**

A Problem of API Authorization

A Solution: Credential Sharing

A Solution: Cookies

A Solution: API Keys

The Solution: OAuth 2.0

Demo: A Very Typical OAuth Flow

OAuth: A Misunderstood Protocol

**OAuth in Detail**

Protocol Endpoints

What Is a Scope?

Authorization Code for Web Applications

Demo: Authorization Code for Web Applications

Implicit Flow for Single Page Applications

Demo: Implicit Flow for Single Page Applications

Client Credentials for Machines

Demo: Client Credentials for Machines

Resource Owner Password Credentials for No One

Demo: Resource Owner Password Credentials for No One

Long-lived Access with Refresh Tokens

Demo: Long-lived Access with Refresh Tokens

Choosing the Right Response Mode

When Things Go Wrong

**Best Practices for Native Applications**

The Unique Issues of Native Applications

Dealing with Stolen Tokens Using PKCE

Choosing the Best Redirect URI

Not All Browsers Are Created Equally

Demo: OAuth for Native Applications in Action

**Extending OAuth**

OAuth + Identity with OpenID Connect

Demo: Identity with OpenID Connect

Automatically Configuring Clients with OAuth Metadata

Securely Authorizing the IoT with the OAuth Device Flow

Demo: Device Flow in Action

Combining SAML and OAuth with the SAML Assertion Grant

**Introduction to OAuth2, OpenID Connect and JSON Web Tokens (JWT)**

**Overview**

Agenda

**The Security Stack for Modern Applications**

Overview

Enterprise Security

The Mobile Revolution

Mobile Enterprise Apps

OAuth2 for Authorization

OpenID Connect for Authentication

**JSON Web Tokens (JWT)**

The Purpose of Security Tokens

JWT Structure and Format

Producing and Consuming JWTs

**Introduction to OAuth2**

OAuth2 Overview

The Valet Parking Analogy

The Players

**OAuth2 Flows**

Code Flow: Authorization Request

Code Flow: Token Request

Code Flow: Access and Refresh Token

Code Flow: Client Management

Code Flow: Summary

Code Flow: Demo

Implicit Flow

Implicit Flow: Demo

Resource Owner Credentials Flow

Resource Owner Credentials Flow: Demo

Client Flow

**OpenID Connect**

OAuth2 and Authentication

The Problems With OAuth2 Authentication

OpenID Connect

OpenID Code Flow

**OAuth2 Concerns**

Eran Hammer

Specification Bloat

Bearer Tokens

Security Theater

Attack Surface

**Resources**

**Web API v2 Security**

**Overview**

**HTTP Security Primer**

Transport Security

X.509 Certificates

SSL Handshake

Developers and SSL

Where to get Certificates from?

Creating Certificates

Demo: Building an SSL Development Environment

Demo: Command Line Tools and Self-hosting

Demo: Fiddler and SSL Tracing

Validating Certificates using .NET APIs

Resources

**ASP.NET Web API Security Architecture**

The Security Pipeline

OWIN/Katana Hosting

OWIN Middleware

Message Handler

Authentication Filter

Authorization Filter

Accessing Client Identity

Demo: Security Pipeline

Demo: Hosting Options

Resources

**Classic Authentication and Katana Authentication Middleware**

Windows Authentication

Demo: Windows Authentication

Basic Authentication

Excursion: Katana Authentication Middleware

Demo: Basic Authentication

X.509 Client Certificates

Demo: Client Certificates and Combining Authentication Methods

Demo: Self Hosting

**JavaScript and Browser-based Clients**

Same Origin Policy

Implicit Browser Authentication

Cross Site Request Forgery (CSRF)

CSRF Mitigation

Demo: Implicit Authentication and CSRF

Demo: CSRF Mitigation using Anti-Forgery Tokens

Cross Origin Resource Sharing (CORS)

Demo: CORS support in Web API

**Token-based Authentication - Part 1**

Modern Applications

Requirements and Complexity

OAuth2

Authorization Servers

Demo: Thinktecture AuthorizationServer

Trusted Applications

Demo: Resource Owner Credential Flow

Adding Refresh Tokens

Demo: Adding Refresh Tokens

Demo: AuthorizationServer and Resource Owner Flow

Excursion: JSON Web Tokens

**Token-based Authentication - Part 2**

Native or Browser-based Clients

Demo: Implicit Flow

Confidential Clients

Demo: Authorization Code Flow

Federation and Delegation

Demo: Assertion Flow

Resources

**Authorization**

Overview

Client vs. User Authorization

Authorization Options

AuthorizeAttribute Usage and Internals

Custom Authorization Logic

Imperative Authorization

Demo: Authorization

**Code School: Building Blocks of Express.js**

**First Steps**

**Middleware**

How Middleware Works

Writing Our Own Middleware

**User Params**

Reading from the URL

Massaging User Data

**Body Parser**

POST Requests

DELETE Requests

**Refactoring Routes**

Route Instances

Route Files

**Acceptance Testing in Node.js**

**Course Overview**

**What Is Acceptance Testing?**

Automated Tests

Types of Automated Tests

What Is Acceptance Testing?

Interpretations of Acceptance Tests

Why Should You Write Acceptance Tests?

**Mechanics of Acceptance Testing**

How to Write an Acceptance Test

Arrange, Act, Assert

Red, Green, Refactor

Acceptance Test Driven Development (ATDD)

How Many Acceptance Tests Should You Write?

Acceptance Tests and Continuous Integration

**Introduce and Set up Demo** **Application**

Prerequisites

Demo Application

Installing Git and Node

Installing MongoDB

Get the Code and Run the App

Familiarize Yourself with the Code - App Startup and Routing

Familiarize Yourself with the Code - Business Logic and Data Access Layers

Familiarize Yourself with the Code - Integration Tests

Familiarize Yourself with the Code - Unit Tests

**Adding Acceptance Tests to Existing Code**

First Acceptance Test

Start Test Instance Automatically

Acceptance Test to Catch a Bug - Extract GET Helper

Acceptance Test to Catch a Bug - Extract POST and DELETE Helpers

Acceptance Test to Catch a Bug - Failing Acceptance Test

Acceptance Test to Catch a Bug - Fix Bug and Passing Acceptance Test

Refactor Acceptance Test

**SuperTest**

What Is SuperTest?

SuperTest: Setup

SuperTest: API Calls

SuperTest: Assertions with .expect()

SuperTest: Assertions with .then() or .end()

Demo: Refactor First Test to SuperTest

Demo: Different Structure for First Test

Demo: Refactor Second Test Block to SuperTest

Demo: Test Cleanup with SuperTest

SuperTest vs. Hand Rolled

**Acceptance Test Driven Development**

Failing Acceptance Test - Two Lists with Names

Passing Acceptance Test - Two Lists with Names

When Can You Ship?

Failing Acceptance Test - Getting a List by Id

Passing Acceptance Test - Getting a List by Id

Acceptance Test - Getting All Lists for a User

Make UI Changes to Support Multiple To Do Lists

Retrospect

**PayPal Integration Using Node.js and Express**

**Course Overview**

**Introduction to PayPal**

Docs and Setup

**Creating the Server**

Creating the Express Server

Creating the View

Creating a MongoDB Service

**Integrating PayPal**

Creating the Configuration

Integrating Single Payments

Integrating Single Payments Repository

Integrating Recurring Payments

Recurring Payments Repository

**Using ES6 in Your Node.js Web Application**

**Course Overview**

**ECMAScript and Node.js**

What is EcmaScript?

What is ES6?

Getting to Know KangaX

ES6 Adoption in Node.js

**Some Syntactic Sugar**

Function Defaults

Rest Parameters

Spread Operator

Object Literal Notation

For of Loop

Template Literals

Destructuring

**New Bindings**

Block Scope

**Function Changes**

Arrow Functions

Lexical Bindings for this

Prototypes

Classes

Extends and Super

Generators

Repository as a Generator

**Built-ins**

Sets

Maps

Promises

**Visualizing Data with MongoDB Charts**

**Course Overview**

Agenda

What Is MongoDB Charts?

Set up Atlas Cluster

User Access and Permissions

Loading Sample Dataset

Working with Dashboards

Import JSON Data

**Get Started with Charts**

Agenda

Structure of Datasets

Encoding Channels

Creating Donut Chart

Visualizing Using Bar Chart

Building Column Chart

Building Text Chart

**Working with Grid, Line, and Geospatial Charts**

Agenda

Creating Grid Charts

Building Line and Area Charts

Visualize Geo Data Using Geospatial Charts

Learning More About Dashboards

**Customization and Embedding of Charts**

Agenda

Customizing Charts

Embedding Charts - Unauthenticated Sharing

Embedding Charts - Verified Signature

Where to Go Next

**Introduction to MongoDB**

**Introduction**

Overview

Scalability

Locking

Scaling Out

Denormalization

Consistency 1 Server

Consistency Multi Server

Inconsistency

Schema Responsibility Shift

**Introduction to MongoDB**

Downloading

Package Content

First Run

Command Line Options

Install As Service

Verify Server

Database is Namespace

Getting Help in the Shell

Replica Sets

Configuring a Replica Set

Verifying Replication Works

Replica Set Failover

**The Mongo Shell**

Shell Role in Ecosystem

Shell Modes

Shell Invoking Syntax

Using Eval

Substantial Scripts

Execute Script Before Enter

Shell Keys and Shortcuts

Multiline Editing

External Editor Integration

Load Script from Within

User RC File

Common Safety Usage Tip

Shell Alternatives

**Saving Data**

Overview

Storage

BSON

Saving Documents

Collections

Document Id

ObjectId

Insert

Insert with Id

Complex Document

Save Danger

Update Command

Update Demo

Set Operator

Unset Operator

Rename Operator

Push Operator

Pull Operator

Pop Operator

Array Type

Multi Update

Find And Modify

Query With Sort

Demo Find And Modify

Documentation

**Finding Documents**

find()

Equality

Projection

Comparison

$not

$in

Arrays

$all

$nin

Dot Notation

Sub-Document

null and $exists

And

More Projection

Cursor

sort()

limit()

skip()

findOne()

**Indexing**

Scan Is Bad

Index Theory

Sort Uses Index

Index Types

Create Index

system.indexes collection

explain()

nscanned

Multi-Term Query

Comparison

Scanned vs. Returned

dropIndex()

Nested Fields

Array Field

Sort

Unique

Sparse

Compound

Sort Direction

Covering Index

Dead Weight

Background build

Index Name

**MongoDB Administration**

**Introduction**

**Installation**

Packaged Distributions

Package Content

Linux Service

File System Type

Disable Last Access Time

apt-get Install

apt-get Outcome

Install on Windows

Data Directory

File Permissions

Windows Service

Steps Review

Safe Stop (Linux)

shutdownServer()

Safe Kill

Dangerous Kill

Start/Stop Windows

Kill Danger (Windows)

Dirty Shutdown Recovery

Shutdown From Shell

**Configuration**

Minimal Configuration

Storage Engines

Journal

Flush Interval

Directory Per DB

IP and Port

HTTP Interface

Upgrading Notes

**Query Crash Course**

Save

Insert

Save Danger

Find

Update

Delete

BSON

ObjectID

Temporal, Ascending

Sort

Limit

Cursor

**Data In, Data Out**

mongodump

simple dump

--oplog

Housekeeping DBs

Credentials

File Access

Deprecated Direct File Access

mongorestore

--drop

Restore Specific Collection

Target DB

--oplogReplay

Practice!

mongoimport

--upsert

--upsertFields

CSV

--fieldFile

mongoexport

Complex Documents

Selective Export

**Indexing**

\_id Index

ensureIndex

Nested Documents

explain()

Sort

Multiple Fields

Sorting Direction

Multy-key

Yielding

Covering Index

Sparse Index

Unique

TTL Index

Rebuild & Compact

Good Index

Limitations

**Replica Sets**

Capped Collection

Simple Replica Set

Reconfigure

Runtime Status

Elections

Quorum Tolerance

More Members

Failover

Recovery from No Quorum

Priority

Step Down

Freeze

Hidden

Network Partition

Chaining

Durability

Acknowledged write-concern

Journaled write-concern

Multi-member write-concerns

Timeouts

write-concern Demo

Timeout Demo

**Sharding**

Sharding Concepts

Simplistic Topology

Using a Sharded Cluster

Production Topology

Picking a Shard Key

Shard Key Cardinality

Shard Key Evaluation

Key Choice Conclusion

Tag Aware Theory

Tag Aware Demo

Moving Chunks

Verifying Tag Aware Works

Caveats

**Monitoring**

Log File

Setting Verbosity Configuration

Query Profiler

Querying Slow Queries

mongostat

mongotop

db.stats()

db.serverStatus()

MMS

**Security**

Surface Area

SSL Support

Replica Set with SSL

Data File Protection

Authentication & Authorization

Roles

Creating a User

Logging In

User Authorization Summary

Intra Cluster Authentication

Key Files in a Cluster

**Miscellaneous**

Introduction

Hardware Choices

CPU

Memory

Disk

Cloud

Storage Engines

wiredTiger

Directory for Indexes

**Big Data & Reporting with MongoDB**

**Introduction**

Limitations of find()

Aggregation Choices

**Getting Started with the Aggregation Framework**

Statefulness

Pipeline Architecture

$group

$unwind

$project

$limit

$skip

$sort

$match

$geoNear

**The $group Pipeline Operator**

Simple $group

$min $max $avg

Group by Field

Key Rules

Value Rules

Value-Of-Field

Field Name

Arrays and Sets

$addToSet

$push

$first

$last

Order Needed

Command behind aggregate()

Result Structure

Limitations

**Document Selection**

$match

$match and Arrays

Compound $match

$match for Efficiency

$sort

$sort and Indexes

$limit

$skip

Outliers

Paging

Paging Cost

**Shaping Documents**

Shaping Needs

$unwind

Thinking Pipeline

$project

Field Selection

Un-Project \_id

Field Renaming

Computation

String Functions

Date Functions

Compound Conditional

$ifNull

Building Blocks

Creating Sub-Document

Complex Aggregation

Complex Scenario

Multiple Step Demo

**Other Operators**

Boolean Operators

Comparison Operators

Arithmetic Operators

String Operators

Date Operators

Conditional Operators

**Performance (Aggregate)**

Key Principals

Thrift

Rules of Thumb

$match Early

$project Trim

Index Awareness

$sort Early

$sort-$limit Pair

$skip-$limit

$project Optimization

Optimization Cycle

**Map/Reduce**

Working on Big Data

What is Map/Reduce?

Map

Reduce

The Process

Distributed Work

Flexibility

Basic Syntax

Map Basics

Reduce Basics

Finalize

Demo

**Map/Reduce - Digging Deeper**

Output Options

Replace

Merge

Reduce

Finalize

Result Type

mapReduce for Ops

Complex Scenario

Mapping for Book

Reducing for Book

Running mapReduce on Book

The Author Collection

Mapping for Author

Reduce for Author

Running mapReduce on Author

Idempotency

Other Scenarios

Usage Considerations

**MongoDB Change Streams: Driving Real Time Events and Streaming Applications**

**Course Overview**

**Streaming and Events – the Backbone of the Modern Architecture**

About This Course

About This module

The Architecture Need

Separate Messaging Infrastructure

Where Change Streams Fit In

The Oplog and Change Streams

**Change Streams**

Starting MongoDB as a Replica Set

Consuming a Change Stream with Aggregation

Change Stream Operation Types

Changes Occur Only on Write

Operation Type for Upserts

Operation Type "Delete"

Multiple Notifications from One Command

Operation Type "Invalidate"

Getting the Full Document

**Advanced Scenarios**

Consumer Demo

Filtering at the Server

Filter on Field Value Change

Event Field Selection

Pipeline Stage Rules

Choosing Separate or Consolidated Consumer

The watch() Syntax

**Understanding Heuristics and Limitations**

Robust Handling

Resuming Work

The Mind-shift

Missed Change Events

Resume at Operation Time

Scaling Out

**Fundamentals of Mongoose for Node and MongoDB**

**Course Overview**

**Course Introduction**

Prerequisites

Why Mongoose?

Demo Application Introduction

Demo: Preview of the Demo Application

**Setting up the Demo Application**

Frameworks and Libraries - Server

Frameworks and Libraries - Client

Helpful Tools

Demo: Develop the Express API

Demo: Vue.js and Vuetify

**Setting up MongoDB**

MongoDB Hosting Options

The Virtual Standups Database

**Building the Mongoose Schema**

Everything Starts with a Schema

Comparison - Relational Database Schema

Simple Schema Example

Complex Schema Example

Demo: Building the Stand-up Notes Schema

**Building a Model**

Our First Model

Documents

Sub-documents

Demo: Saving Documents

**Retrieve Data Using Queries**

Documents and Helper Methods

Updating and Removing Documents

Demo: Querying Documents

**Using Validation Middleware**

Built-in Validators

Introduction to Middleware

Custom Validators

Handling Validation Errors

Adding Validation

Demo: Adding Validation

**Next Steps**

Finishing Touches

Migrate to the Cloud

Review the Client Project

Other Resourses

**Moving Forward with Mongoose.js**

**Course Overview**

**Course Introduction**

What We Will Cover

Prerequisites

Review

Tools

Preview of Demo Application

**Review of Breaking Changes**

Breaking Changes

Top Offenders

New Versioning

**New Features**

Schemas

Validators

Middleware

Other Improvements

Demo

**Extending Functionality with Plugins**

Custom Plugin

Plugin Library

Demo

**Query Middleware and Hooks**

Introduction to Middleware

Document Middleware

Query Middleware

Error Handling Middleware

**Discriminators**

Inheritance Overview

Extending a Base Schema

Discriminator Demo

User Interface Considerations

UI Considerations Demo

**Promises**

What Is a Promise?

Promise

Demo - Using Promises

Using Other Promise Libraries

Demo - Using BlueBird or Q