ProTech protechtraining.com *"Charting the Course ... ... to Your Success!"*

# Intermediate Development with Node.js

## Course Summary

### Description

This course is a follow-on to the Intro to Node course for developers. This course focuses on more hands on development using Express and related packages to make developers productive and to help follow best practices.

This course combines lecture and hands-on exercises to give students confidence in creating solutions using Node. Students will explore popular modules such as Express. During the course students will create the different parts of a web application to allow records to be Created, Read, Updated, and Deleted (CRUD). This course uses Postgres as a relational database but can be customized based upon the needs of the client to use other relational or NoSQL databases.

### Objectives

After taking this course, students will be able to:

- Work with common core modules
- Leverage Express.js middleware
- Create a CRUD app with a REST API
- Use Mocha and chai Unit testing tools
- Create custom packages and require in projects

### Topics

- Warm-up / Review of Core Concepts
- Setting up Express
- Database Connections
- Handling User Input and Requests
- Error Handling
- Validation Strategies
- Session Management in Node
- Production Security Concerns
- Testing Node
- Production Practices

### Audience

This course is designed for experienced JavaScript developers who wish to understand Node JS and to execute JavaScript outside of the browser.

### Prerequisites

Attendees should have experience with using Node, npm install, understand concepts of asynchronous programming and promises. Knowledge of REST based web services is helpful but not required.

### Duration

Three days

ProTech protechtraining.com *"Charting the Course ... ... to Your Success!"*

# Intermediate Development with Node.js

## Course Outline

**I. Warm-up / Review of Core Concepts**
- A. Using ES6 (ES2015) syntax
- B. Using promises and Bluebird
- C. Using the CommonJS interface for modules
- D. Using require with native modules and relative files
- E. Practice with packages: lodash, various moment packages, Async.js,, Request.js
- F. “Linting” with eslint
- G. Usin Node for builds using Grunt.js
- H. Using SuperAgent for client-side AJAX requests

**II. Setting up Express**
- A. Review of Express and the Express Application Generator
- B. Configuring Express for our course web server
- C. Creating Routing files
- D. Using express.static to access static assets

**III. Database Connections**
- A. Creating database connections using configuration files
- B. More on using knex and migration files
- C. Using Bookshelf - a JavaScript ORM for Node

**IV. Handling User Input and Requests**
- A. Passing data (locals) to templates
- B. Use express-promise-router and unhandled-error
- C. Use a State object, for stateful things like the error reporter object
- D. Use Wrapper functions around module contents, for receiving state object
- E. Object destructuring for 'unpacking' items from state object
- F. Usage of body-parser module
- G. Server-side caching

**V. Error Handling**
- A. Using a custom regular middleware
- B. Using Error-handling Express middleware
- C. Custom error types
- D. Exposing and logging using conditional error logic

**VI. Validation Strategies**
- A. Add express-validation package
- B. Validate user input on backend
- C. Validate user input on frontend
- D. Creating validation schemas with Joi

**VII. Session Management in Node**
- A. Allowing User Registration
- B. Creating a Login Route
- C. Authenticating Login with database
- D. Forcing logins with route-level middleware (requireLogin)
- E. Sharing State of User
- F. Logging out to end the Session
- G. Sessions + express-session usage
- H. Using Promises + promisifying
- I. Hash user's password
- J. Returning auth cookie
- K. Using passport with local strategy

**VIII. Production Security Concerns**
- A. Avoided deprecated and vulnerable versiosn of Express
- B. Transmitting sensitive data, with Transport Layer Security (TLS)
- C. Securing apps by setting HTTP headers with helmet
- D. Best Practices with cookies
- E. Addressing Cross-Site Request Forgery (CSRF)
- F. Using tools to check that your dependencies are secure

**IX. Testing Node**
- A. Review of using Mocha and Chai
- B. Using Supertest for a test-driven approach to development
- C. Mocking in tests using Nock and Sinon
- D. Using Istanbul for code coverage

**X. Production Practices**
- A. Monitoring
- B. Metrics of apps performance
- C. Error reporting