Node.js Training: Server-side JavaScript with Node.js and Express

Course Summary

Description

ProTech's Server-side JavaScript with Node.js and Express training teaches experienced JavaScript developers how to create server-side applications with JavaScript and Node.js, culminating with an MVC application built on the Express framework that queries databases and calls back-end web services.

Objectives

At the end of this course, students will be able to:

- Learn why server-side JavaScript is useful
- Install Node.js
- Learn how Node.js is architected to allow high scalability with asynchronous code
- Create basic web applications with Node.js
- Automate tasks with Gulp
- Build an HTTP server using the core modules in Node.js
- Use stream I/O to efficiently serve the web pages
- Create modules to organize the server
- Test the reliability of the application with unit tests
- Convert the application to an MVC framework using Express
- Interface to a MongoDB database and a web service

Prerequisites

Node.js training attendees should have a thorough knowledge of JavaScript. They should be familiar with web server application design concepts (such as accessing databases and SOA concepts), as well as basic HTML and CSS.

Software Needed on Each Student PC

- A recent version of Google Chrome or Mozilla Firefox
- A local installation of Node.js
- Admin/root or sudoer privileges to install additional features during the class
- A JavaScript development tool of your choice
- Additional lab files that Accelebrate will provide

Duration

Two day

Node.js Training: Server-side JavaScript with Node.js and Express

Course Outline

I. **Introduction**

II. **Foundation**
   A. The Node.js framework
   B. Installing Node.js
   C. Using Node.js to execute scripts

III. **Node Projects**
   A. The Node Package Manager
   B. Creating a project
   C. The package.json configuration file
   D. Global vs. local package installation
   E. Automating tasks with Gulp.

IV. **HTTP**
   A. The HTTP protocol
   B. Building an HTTP server
   C. Rendering a response
   D. Processing query strings
   E. Using Representational State Transfer
   F. Configuring TLS

V. **File System**
   A. Synchronous vs. asynchronous I/O
   B. Path and directory operations
   C. __dirname and __filename
   D. Asynchronous file reads and writes

VI. **Buffers, Streams, and Events**
   A. Using buffers for binary data
   B. Flowing vs. non-flowing streams
   C. Streaming I/O from files and other sources
   D. Processing streams asynchronously
   E. Configuring event handlers

VII. **Modules and Unit Testing**
   A. Modularization
   B. The CommonJS and RequireJS specifications
   C. Defining modules with exports
   D. Modules are singletons
   E. Creating a package
   F. Module scope and construction
   G. Unit testing frameworks
   H. What to test and how to test it
   I. Building unit tests with Mocha

VIII. **Express**
   A. The model-view-controller pattern
   B. Defining Jade and Handlebars templates
   C. Building a front-end controller
   D. Defining routes
   E. Creating actions
   F. Configuring Express to use Handlebars
   G. Using REST
   H. Reading POST data
   I. Building Handlebars helpers
   J. Adding middleware

IX. **Data Sources**
   A. How Node.js connects to databases
   B. RDBMS databases and NoSQL databases
   C. Connecting to RDBMS and NoSQL databases
   D. Performing CRUD operations
   E. Building client requests to web services

X. **Conclusion**