Test Driven Development Using React.js and ES6

Course Summary

Description

React.js is a popular, high-performance JavaScript library for building rapidly responsive user interfaces. EcmaScript 6 is the latest version of JavaScript, which includes support for classes and modules. Test Driving Development Using React.js teaches experienced JavaScript developers the skills they need to immediately implement React.js in their applications using professional test driven development tools and methodologies.

Objectives

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

- Install, configure, and use modern web tooling
- Understand Test Driven Development
- Learn to create test suites using Jasmine
- Write ES6 code and compile it using Babel
- Understand what React.js is and what problem it solves
- Explore the basic architecture of a React.js application
- Gain a deep knowledge of React.js components and JSX
- Build a working application that uses React.js's components
- Use Redux for maintaining state in a React.js application
- Learn React.js best practices

Topics

- Introduction
- Development Eco-System
- Configure a Local Web Server
- Manual Testing
- Test Driven Development
- Modularity
- Building with webpack
- ES6
- Redux
- The Document Object Model
- React.js

Audience

This course is designed for experienced JavaScript developers.

Prerequisites

All attendees must be experienced JavaScript developers with a fairly advanced understanding of JavaScript, including prototypes and functions as first class citizens. If your group doesn't yet have this experience, we could readily add one or two days to the beginning of your course to appropriately prepare them.

Duration

Five days

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

Test Driven Development Using React.js and ES6

Course Outline

I. Introduction
- A. Intro to modern front-end development
- B. What is React.js?
  1. What problem does it solve?

II. Development Eco-System
- A. Core Tools
  1. Code Editor
  2. node.js
  3. git
  4. Command Prompt
- B. Creating a Project
- C. Reproducible Builds
  1. Version control with git
  2. Build Tools
     a) installing and uninstalling with npm
     b) Gulp, Grunt, Jake
     c) Installing a Build Tool
  3. Dependency management
     a) Strategies
  4. Shell scripts and file permissions
  5. Managing external dependencies
- D. Code Analysis
  1. Lint tools
     a) JSLint
     b) JSHint
     c) ESLint
  2. "use strict"
  3. iffe function
  4. Add linting to automated build
  5. Configuring JSHint
  6. JavaScript Pitfalls

III. Configure a Local Web Server

IV. Manual testing

V. Test Driven Development
- A. Assertions
- B. JS exception handling
- C. Jasmine overview
- D. TDD vs BDD
- E. Automated Cross-browser Testing with Karma

VI. Modularity

VII. Building with webpack

VIII. ES6
- A. Variable scoping with const and let
- B. Functional programming with arrow functions
- C. Creating and consuming generator functions
- D. Spread operator
- E. Classes and Modules
- F. Using Babel to compile back to ES5

IX. Redux
- A. Redux architecture
- B. Stores & Immutable State Tree
- C. Redux Actions
- D. Using ES6 to create Pure Functions
- E. Reducers
- F. Reducer Composition
- G. Redux with Ajax

X. The Document Object Model

XI. React.js
- A. Hello World Application
- B. JSX
  1. What is JSX?
  2. Using JSX
  3. Using React with JSX
  4. Using React without JSX
  5. Precompiled JSX
- C. Components
  1. Component Life-Cycle
  2. Virtual DOM
  3. Events
  4. State Machines
  5. Compositions
  6. Communication Between Components
  7. Reusable Components
  8. Testing React components
- D. Forms
  1. Controlled Components
  2. Uncontrolled Components
- E. Node.js and Server DOM Manipulation
- F. React.js Best Practices