Course Outline

Beginning Vue.js

Course Summary

Description

Vue.js is an open source JavaScript library for building modern, interactive web applications. Its component-based approach, intuitive API, blazingly fast core, and compact size make Vue.js a great solution to craft your next frontend application.

This is a 3-day course, balancing theory and hands-on applications that are focused on practical takeaways. This course will provide you with practical solutions to common problems when building an application using Vue. We start off by exploring the fundamentals of Vue.js: its reactivity system, data-binding syntax, and component-based architecture, through practical examples. After that, we delve into integrating Webpack to enhance your development workflow using single file components. Finally, we take an in-depth look at Vuex for state management and Vue Router to route in your single-page applications.

Hardware:

- For successful completion of this course, students will require computer systems with the following:
- Processor: Intel Core i3 or equivalent
- Memory: Minimum 4 GB RAM
- Hard disk: Minimum 35 GB
- Operating System: Windows (7 SP1 64-bit or higher)
- An internet connection

Software:

- Browser: Google Chrome or Mozilla Firefox (with the latest updates installed)
- Visual Studio 2017 (the latest version)
- Vue.js and vue-devtools Chrome or Firefox browser extension
- Node.js and npm
- Vuex library
- Feathers API player
- Axios promise-based HTTP client
- .NET Framework 4.6 or higher

Objectives

After taking this course, students will be able to:

- Create dynamic and animated lists
- Identify how to use computed properties
- Animate with JavaScript instead of CSS
- Package reusable transitions into components
- Create basic AJAX requests with Axios
- Utilize Jasmine for testing Vue and Karma workflow
- Utilize single-page applications and Webpack

Topics

- Getting Started With Vue.Js
- Basic Vue.Js Features
- Transitions And Animations
- All About Components
- Vue Communicates With The Internet
- Single-Page Applications
- Unit Testing And End-To-End Testing
- Organize + Automate + Deploy = Webpack

Audience

This course is for web developers who have little or no prior experience with Vue.js. It mainly targets JavaScript enthusiasts who want to learn a modern and simple JavaScript framework.

Duration

Three days

Course Outline

Beginning Vue.js

Course Outline

I. Getting Started With Vue.Js
- A. A Simple Vue.js Program
- B. Lists and their Types
- C. Creating a Dynamic and Animated List
- D. Reacting to Events Such as Clicks and Keystrokes
- E. Choosing a Development Environment
- F. Formatting Your Text with Filters

II. Basic Vue.Js Features
- A. Learning How to Use Computed Properties
- B. Filtering a List with a Computed Property
- C. Sorting a List with a Computed Property
- D. Formatting Currencies with Filters
- E. Formatting Dates with Filters
- F. Displaying and Hiding an Element Conditionally
- G. Adding Styles Conditionally
- H. Adding Some Fun to Your App with CSS Transitions
- I. Outputting Raw HTML
- J. Creating a Form with Checkboxes
- K. Creating a Form with Radio Buttons
- L. Creating a Form with a Select Element

III. Transitions And Animations
- A. Integrating with Third-Party CSS Animation Libraries
- B. Adding Your Own Transition Classes
- C. Animating with JavaScript Instead of CS
- D. Transitioning on the Initial Render
- E. Transitioning Between Elements
- F. Letting an Element Leave Before the Enter Phase in a Transition
- G. Adding Entering and Leaving Transitions for Elements of a List
- H. Transitioning Elements That Move in a List
- I. Animating the State of Your Components
- J. Dynamic Transitions

IV. All About Components
- A. Creating and Registering a Component
- B. Passing Data to Your Components with Props
- C. Making Components Talk to Each Other
- D. Making Components Talk with Vuex
- E. Reading a Child's State
- F. Using Components in Your Own Components
- G. Content Distribution with Slots
- H. Single File Components with Webpack
- I. Loading Your Components Asynchronously
- J. Having Recursive Components

V. Vue Communicates With The Internet
- A. Sending Basic AJAX Requests with Axios
- B. Validating User Data before Sending It
- C. Recovering from an Error during a Request
- D. Creating a REST Client (and Server!)
- E. Implementing Infinite Scrolling
- F. Processing a Request before Sending It Out
- G. Preventing XSRF Attacks on Your App

VI. Single-Page Applications
- A. Creating an SPA with Vue-Router
- B. Fetching Data before Switching Route
- C. Managing Errors for Your Routes
- D. Adding a Progress Bar to Load Pages
- E. Using Named Dynamic Routes
- F. Having More Than One Router-View in Your Page
- G. Composing Your Routes Hierarchically
- H. Adding Transitions between Your Routes
- I. How to Redirect to Another Route
- J. Saving Scrolling Position When Hitting Back

VII. Unit Testing And End-To-End Testing
- A. Using Jasmine for Testing Vue
- B. Adding Karma to Your Workflow
- C. Testing Your Application State and Methods
- D. Testing DOM Asynchronous Updates
- E. End-to-end testing with TestCafe
- F. Stubbing External API Calls with Sinon.JS
- G. Measuring the Coverage of Your Code

VIII. Organize + Automate + Deploy = Webpack
- A. Extracting Logic from Your Components to Keep the Code Tidy
- B. Organizing Your Dependencies with Webpack
- C. Using External Components in Your Webpack Project
- D. Developing with Continuous Feedback with Hot Reloading
- E. Running a Code Linter While Developing
- F. Releasing Your Components to the Public