Updated 05/02/2025

Sign up

Node.JS training

4 days (28 hours)

Presentation

Nodejs is an open-source, event-driven software platform based on Google Chrome's V8 engine, which enables the development of network applications in JavaScript while guaranteeing excellent performance.

The major advantage of Node.js® lies in the ability to use a single programming language, JavaScript, across all layers of a software architecture, thus facilitating rationalization of the code base and communication within the technical team.

The tool is used in production by a large number technology companies (LinkedIn, PayPal and Netflix).

It is supported by all major cloud providers (AWS, Google App Engine, Microsoft Azure).

As with all our programs, our training covers the latest stable release this platform, as well as discovering what's new in version 23 of Node.js.

Objectives

- Master the main features of NodeJS
- Know how to use NodeJS, NPM and its ecosystem in the latest versions
- Configuring a NodeJS server
- Developing a web application with NodeJS and ES2022
- Mastering event-driven & asynchronous programming
- Creating and managing APIs with NodeJS
- Secure, industrialize, test & deploy your application

Target audience

Web developers

Prerequisites

- Knowledge of JavaScript
- Knowledge a client-side framework or other object-oriented programming language (java, php, etc.)
- Test My Knowledge

Technical requirements

- An IDE
- Node.js installed
- A MongoDB database
- Administrator access to avoid permission restrictions

Node.JS training program

Day 1 - Introduction to the fundamentals

JavaScript reminder

- The history of language
- Language fundamentals
- The Event Loop
- JavaScript engines
- Focus on Google's ES2022 V8

engine

- Introduction to ECMAScript
- Variable declaration and scope
- Object literals
- JSON format
- The classes
- Destructuring
- Rest and Spread
- Template strings
- Arrow functions
- ES modules
- Native Node compatibility
- Using the latest version of JavaScript with Babel

Asynchronous programming

- Callbacks
- NodeJs callbacks
- The "callback hell" problem
- Use async.js to avoid callback hell
- The promises
- Async control flow with async /

await Introduction to NodeJs

- The genesis of NodeJs
- Executing server-side JavaScript
- Installing the Nodejs server
- A first program
- Run a file
- Overview of the Node.Js API
- Comparison with other technologies

Overview of the main Node.JS components

- Node CLI (command-line tools)
- Different development environments (IDEs)
- NPM - The node.js package manager
- package.json
- Node Modules
- Tools: Development Tools and

Frameworks Global objects

- Focus on Node API documentation
- The global object and the difference with window
- Using the setTimeout, setInterval and setImmediate functions
- logging on process.stdout with console
- Access the file context with _____dirname and filename
- Access to server process and bone hardware configuration

Day 2 - Manipulating the Node API

Node module management

- What is a Node module?
- Core modules
- Module import with require and import
- Module configuration and initialization
- Use of utility modules (util, path, queryString, url)
- Module creation

Discover NPM

- The package manager
- The npm command-line tool
- The yarn alternative
- Command line module search
- The npmjs.com website
- Module search on the site
- Local or global installation
- Module packaging
- The package.json file
- Declaration of dependencies
- Version conflict management
- Dependency management by environment

File manipulation

- Introducing the fs module
- Synchronous file playback
- Asynchronous file reading
- Asynchronous file creation
- Asynchronous folder deletion

Event-driven programming

- Why event programming
- Presentation of the events module
- Using EventEmitter
- Practical example

Day 3 - Web application development

Network access from NodeJs

- Network recall
- Network-oriented Node core modules
- Using the udp and net modules
- Using the http and http2 modules
- Using the dns module
- Focus on the HTTP protocol

Creating a web server with the Node.JS API

- What is an HTTP server?
- Launching a Node web server
- HTTP request/response management
- Setting up a route manager
- Asynchronous query processing

Creating a web server with Express

- Introduction to Express
  - Comparison with Fastify and NestJS
- Express server launch
- Configuring Express applications with middleware
- Using the Express route manager
- Templating engines
- Pug template creation and HTML page rendering
- HTML form processing

Database connection

- Compatible databases
- Introduction to MongoDB
- Using the mongoose package: model creation and querying
- Linking a route to a mongoose model
- Restify a data model with express-restify-mongoose
- Using the sequelize package: model creation and querying
- Linking a route to a sequelize model
- Restify a data model with finale-rest Real-

time bidirectional communication

- Introduction to Websocket
- Introducing socket.io
- Server-side communication management
- Customer communication management

Day 4 - Industrializing a Node.JS application

Build your project

- Why build a nodeJs project?
- Build tools
- Writing your own scripts
- Starting a boilerplate project (Yeoman

style) Testing and debugging

- Node core modules for testing and debugging (console, debugger, inspector, repl, assert)
- The npm package ecosystem for unit and integration testing
- Assertion modules: assert and Chai
- Testing your module with

Mocha The NPM package

ecosystem

- Choosing the right npm package: viability analysis
- The main API development frameworks
- Who are the npm package developers?
- How do I contribute to an npm package?

Securing a Node/Express application

- Core security modules (crypto, https, tls)
- Password encryption with bcrypt
- Helmet package
- Authentication with Passport

Facilitate Node application development in teams

- Cleanly version your code with git
- Documenting code with docco
- Documenting an API using Swagger
- Harmonizing a code base with ESLint
- Impose typing via Typescript or Flow

Optional modules (+ 1 day)

Stream and buffer management

- What is a stream?
- Comparison between the use of unix streams and nodejs
- Stream types: readable, writable, duplex and transform
- The Buffer class
- An example of high-level use

Node application deployment - Add-on module (+1 day)

- Deploying code on Heroku
- Deploying code on AWS
- Node application containerization with Docker
- Process management in a production environment with PM2
- Continuous integration with Jenkins and TravisCI

Companies concerned

This course is aimed at both individuals and companies, large or small, wishing to train their teams in a new advanced computer technology, or to acquire specific business knowledge or modern methods.

Positioning on entry to training

Positioning on entry to training complies with Qualiopi quality criteria. As soon as

On final registration, the learner receives a self-assessment questionnaire which enables us to assess his or her estimated level of proficiency in different types of technology, as well as his or her expectations and personal objectives for the forthcoming course, within the limits imposed by the selected format. This questionnaire also enables us to anticipate any connection or security difficulties within the company (intra-company or virtual classroom) which could be problematic for the follow-up and smooth running of the training session.

Teaching methods

Practical course: 60% Practical, 40% Theory. Training material distributed in digital format to all participants.

Organization

The course alternates theoretical input from the trainer, supported by examples, brainstorming sessions and group work.

Validation

At the end of the session, a multiple-choice questionnaire verifies the correct acquisition of skills.

Sanction

A certificate will be issued to each trainee who completes the course.