+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Updated on 20/06/2025

Sign up

Formation Bun

3 days (21 hours)

Presentation

Our Bun training course will help you discover and master this JavaScript runtime. designed to replace Node.js, npm and even your favorite bundler. You'll learn how to develop modern fullstack applications, test and deploy them efficiently, while exploiting Bun's exceptional performance. You'll start by understanding Bun's unique architecture, installation and minimalist CLI, as well as its runtime engine capable of natively handling JavaScript and TypeScript. The aim: to quickly become autonomous and productive with a fast, all-in-one tool. We'll then get to the heart of Bun: its integrated bundler, ultra-fast dependency management and native test runner. You'll see how to build, organize and maintain robust projects without resorting to Babel, Webpack or Jest. An entire module will be dedicated to creating HTTP servers, REST APIs and handling WebSockets. You'll also learn how to interact with databases, secure your applications and deploy your projects using Docker or Railway. As with all our training courses, this one will be presented with the latest stable version of Bun.

Objectives

- Understand the inner workings of Bun and how it differs from Node.js and Deno
- Create, test and run TypeScript applications without external transpilation
- Master package management, bundling and hot reloading with Bun's integrated tools
- Design high-performance HTTP servers and secure REST APIs with Bun.serve
- Integrate databases, WebSockets and middleware in a fullstack architecture
- Be able to deploy and supervise a Bun application in production

Target audience

- JavaScript developers
- Fullstack developers

Prerequisites

- Knowledge of JavaScript and a client-side framework
- Knowledge of a typed language

Bun training program

Introduction to Bun

- Comparison with Node.js and Deno
- Bun installation (Linux, macOS, Windows WSL)
- Project initialization (bun init)
- Project structure Bun
- bun run, bun install, bun bun
- Execute TypeScript/JavaScript scripts directly Useful commands: bun upgrade, bun pm, etc.

Bun as Runtime JS/TS

- .js, .ts, .tsx, .jsx files
- No explicit compilation with tsc
- fetch, Request, Response, WebSocket, FormData
- API Blob and TextEncoder/TextDecoder
- Standard modules (fs, http, crypto, etc.)
- Limitations and compatibility monitoring
- CommonJS modules vs. ESModules

Package management with Bun

- Using and comparing with npm/yarn Generating bun.lockb Standard node_modules tree Managing native or compiled packages Limitations and error resolution URL imports, global packages, integrated polyfills

bundler and stacker

- Bundling a project into a file
- Minification, tree-shaking and sourcemaps
- Supported file types (.ts, .jsx, .tsx)
- No Babel or TypeScript Compiler required
- Introduction to future bundling plugins
- System scalability

HTTP servers and web development

- Using Bun.serve
- Management of routes, headers, JSON
- Creating a complete API with Bun
- Parameters, body, status, JSON
- Using bun --watch
- Automatic server restart

Tests with Bun

- Writing unit tests with the integrated framework Syntax (test(), expect()) Files *.test.ts Groups, setup/teardown

Mocking & snapshot testing

- Mock functions and modules
- Snapshot support (under development)

Bun + Frontend

- React, Preact, Vue, Svelte (bundler support)
- Using JSX/TSX with Bun
- CSS, JSON, SVG files
- Import resolution behavior
- First steps with SSR en Bun
- Current limits and best practices

Bun: Backend & Fullstack

- Integrated SQLite (via bun:sqlite)
- Using Prisma or Drizzle
- Creating WebSocket servers
- Real-time communication
- Cookies, sessions, headers
- Simple middleware creation

Packaging and deployment

- Compiling with bun build
- Deployment on Vercel, Railway, Docker

Monitoring and logging

- Native logs and JSON format
- Integration with external tools (Sentry, Logtail)

Performance and benchmarks

- Bun application profiling
- Comparative analysis vs. Node.js

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 at the start of training complies with Qualiopi quality criteria. As soon as registration is finalized, 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 training to come, 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, with 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.