

Updated on 03/17/2026

Sign up

Svelte Training

3 days (21 hours)

Overview

The [Svelte](#) training course will introduce you to this open-source JavaScript framework, which allows you to develop an application using a minimalist syntax and then deploy it.

Build your skills by mastering Svelte's concepts and syntax. You'll discover why Svelte is the framework to choose over its competitors.

During this course, you'll learn to manage states and various stores. This will enable you to apply the skills needed to build your own custom store.

Learn how to implement animations and transitions to improve the user experience of the application.

Like all our training courses, this one will introduce you to **the latest stable version** of the technology and its new features.

Objectives

- Understand how to set up the development environment and install Node.js
- Gain an in-depth understanding of the basic syntax of Svelte 5 and creating projects with Vite
- Master the reactivity system
- Apply the concepts of conditions, loops, and event handling
- Explore the use of components, snippets, and parent/child communication via callback props
- Understand state management with context, stores, and runes
- Master animations, transitions, and form handling
- Develop full-stack applications with SvelteKit

Target Audience

- Developers
- Lead Developer

Prerequisites

- Knowledge of JavaScript
- Knowledge of HTML and CSS

Our Svelte Training Program

Introduction to Svelte

- Setting up the development environment
- Installing or updating Node.js
- Useful extensions in VS Code
- The official Svelte website and interactive documentation
- Getting Started with Svelte via the Playground
- Creating your first Svelte project with Vite
- Structure of a Svelte project: .svelte files, scripts, and styles
- Adding your first component

The basic syntax of Svelte

- Creating a standard Svelte skeleton with Vite
- Interpolations (expressions in the markup)
- Plain text or HTML interpolations
- Dynamic attributes
- Attribute shortcuts
- Conditional classes
- Conditional styles

Components

- Introduction to components
- Declaring props
- Default values and optional props
- Bindable props
- Child-to-parent communication using callback props (functions passed as props) •
Replacing createEventDispatcher with callback props
- Chaining communication between components
- Exercise

Conditions, loops, and event handling

- Conditional execution of HTML code
- Each loops
- Two parameters in an each loop
- Iterating over JSON objects with each
- Key blocks to force re-rendering
- Handling promises in the template
- Event handling
- Event modifiers

Runes and reactivity

- Overview of the Runes system in Svelte 5
- Declaring reactive state with \$state
- Deep vs. Shallow Reactivity: \$state vs. \$state.raw
- Derived values with \$derived
- Complex derived calculations with \$derived.by
- Side effects with \$effect
- Pre-rendering effects with \$effect.pre
- Debugging reactivity with \$inspect
- Comparison with the old reactive syntax (\$:)

Snippets and composition

- Introduction to snippets (replacing "slots")
- Declaring snippets and rendering snippets
- The special children snippet
- Named snippets
- Optional snippets and conditional rendering
- Passing data to snippets
- Error handling with <svelte:boundary>
- The {@attach} directive for element actions

Propagated context

- Sharing data with the context (setContext / getContext / createContext) • Context scope and limitations

Svelte stores

- Fundamental concept of a store
- Creating a writable store
- Read-only stores (readable)
- Derived stores
- Auto-subscriptions with the \$ prefix
- Subscription control and memory leak prevention
- Moving business logic into the store

Form management

- Data binding with bind:value
- Text inputs and number inputs
- Radio and checkbox inputs
- Select and option
- Other input tags (textarea, range, etc.)
- Binding with a component (\$bindable)
- Front-end form validation

Animations and transitions

- HTML element transitions (fade, fly, slide, scale, blur, draw)
- Customizing transition settings
- Capturing the start and end of a transition
- Using different fade-in and fade-out transitions (in: / out:)
- Animating numbers and values with the Spring and Tween classes
- Animating an inline CSS property with Spring
- Animating lists with the animate: directive

Introduction and architecture

- Overview and architecture of a SvelteKit application
- Building a SvelteKit application from scratch

Routing and navigation

- File-based routing system
- Pages and layouts (+page.svelte, +layout.svelte)
- Nested layouts and layouts specific to a group of pages
- Defining global style rules
- Dynamic routes and route parameters

Data loading

- Loading data on the server side with load functions (+page.server.ts, +layout.server.ts)
- Loading data on the client side with the load functions (+page.ts, +layout.ts)
- Multiple data sources
- API routes with +server.ts

Forms and error handling

- Error handling (+error.svelte, error())
- Form actions for server-side form processing

Configuration and deployment

- Hooks (handle, handleError, handleFetch in hooks.server.ts)
- Page options (ssr, csr, prerender, trailingSlash)
- Deployment adapters (adapter-auto, adapter-node, adapter-static, adapter-vercel, etc.)
- SPA mode with adapter-static and ssr=false

Target Audience

This training is intended for both individuals and companies, large or small, wishing to train their teams in new advanced IT technology or to acquire specific professional knowledge or modern methods.

Assessment upon enrollment

The pre-training assessment complies with Qualiopi quality standards. Upon final registration, the learner receives a self-assessment questionnaire that allows us to evaluate their estimated proficiency in various types of technologies, as well as their expectations and personal goals for the upcoming training, within the limits imposed by the selected format. This questionnaire also allows us to anticipate certain connection or internal security issues within the company (intra-company or virtual classroom) that could pose challenges for monitoring and ensuring the smooth running of the training session.

Teaching Methods

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

Organization

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

Certification

At the end of the session, a multiple-choice quiz is used to verify that the skills have been properly acquired.

Certification

A certificate will be issued to each trainee who has completed the entire training program.