

**Introduction to Svelte**

**Course Number:** SVLT-100
**Duration:** 3 days

**Overview**

[Svelte](https://svelte.dev/) is a JavaScript library for building user interfaces, but instead of running in the browser (like React and Vue), Svelte runs at *build time*, surgically updating the DOM. As a result, developers can write higher-performance applications that run faster and more efficiently.

This Svelte course teaches attendees the fundamentals of this innovative JavaScript framework. Attendees learn how to set up a development environment and implement the core features of SvelteKit, including routing, server-side rendering, and unit testing. Participants create static and dynamic pages, understand template reactivity, and work with Svelte components. The course covers event handling, forms, lifecycle, state management, and routing. Attendees also explore advanced topics, including error handling and asynchronous data. By the end of this course, participants will confidently build robust, high-performance web applications with Svelte.

**Prerequisites**

All students must have JavaScript and HTML programming experience. Experience with CSS is helpful but not required.

**Materials**

All Svelte training attendees receive comprehensive courseware.

**Software Needed on Each Student PC**

Students need a free, personal GitHub account to access the courseware and permission to install Node.js and Visual Studio Code on their computers. Students also need permission to install NPM Packages and Visual Studio Extensions. Accelebrate provides a cloud-based environment if students cannot configure a local environment.

**Objectives**

* Understand the fundamentals of Svelte and how it compares to other frameworks
* Set up a development environment for Svelte using SvelteKit
* Work with Svelte files and extensions for popular IDEs
* Learn about SvelteKit’s features, including Vite tooling, routing, server-side rendering, and unit testing
* Create static and dynamic pages in Svelte and understand their structure, content, and functionality
* Master Svelte’s template reactivity and change data through assignments and reactive statements
* Work with Svelte components and understand their structure
* Compose props and events
* Understand event handling in Svelte, including DOM events, event handlers, modifiers, and dispatching component events
* Explore advanced Svelte topics, including forms, lifecycle, state management, routing, error handling, and asynchronous data

**Outline**

* Introduction
	+ What is Svelte?
	+ What problem does Svelte solve?
	+ Svelte vs. Other Frameworks
	+ Svelte Compiler
* Development Environment
	+ Requirements
	+ SvelteKit
	+ Svelte Files
	+ Svelte Extension for Visual Studio Code
	+ Run/Debug Svelte App in Visual Studio Code
	+ Svelte Extension for WebStorm
	+ Run/Debug Svelte App in WebStorm
* SvelteKit Overview
	+ Vite Tooling
	+ Development Server
	+ Routing
	+ Deployment
	+ Server-side rendering
	+ Unit Testing
* Getting Started
	+ Exploring the REPL
	+ Svelte Layout
	+ Svelte Page
	+ Svelte Component
	+ Svelte Architecture
	+ Svelte Element Directives
	+ Compiling Svelte Files
* Static Pages
	+ What is a Static Page?
	+ What problem do Static Pages solve?
	+ Static Page File Structure
	+ Setting Head Content
	+ HTML Content
	+ CSS Content
	+ Comments
	+ Scoped CSS
	+ Handling Images
	+ Hot Module Reloading
	+ Server Pre-rendering
	+ Page Routing
* Dynamic Pages
	+ What is a Dynamic Page?
	+ What problem do Dynamic Pages solve?
	+ Client-Side Rendering
	+ Dynamic Page File Structure
	+ JavaScript Content
	+ Using Variables
	+ Using Expressions
	+ Data Binding
	+ Class and Style Directive
	+ Event Binding
	+ Logic Blocks
	+ Debug Tag
* Template Reactivity
	+ Principles
	+ Changing Data through Assignments
	+ Reactive Statements
	+ Updating Arrays and Objects
* Component Basics
	+ What is a Component?
	+ What problem does it solve?
	+ Calling Components vs HTML Elements
	+ Component File Structure
	+ Component Props
	+ Component Events
* Component Composition
	+ What is Component Composition?
	+ What problem does it solve?
	+ Nested Components
	+ Passing Data to Child Components
	+ Handling Events and Receiving Data from Child Components
	+ Component Tree Best Practices
* Event Handling
	+ Event Handling Element Directives
	+ DOM Events
	+ Adding Event Handlers
	+ In-line Handlers
	+ Event Modifiers
	+ Dispatching Component Events
	+ Forwarding Events
* Data binding
	+ Top-down data binding by default
	+ Communication with props and events
	+ Using two-way data binding
* Forms
	+ HTML Form Element
	+ Named Form Actions
	+ Form Validation
	+ Form Submission
	+ Progressive Enhancement
* Lifecycle
	+ Mount
	+ Destroy
	+ Before Update
	+ After Update
	+ Tick
* State Management
	+ Stores
	+ Writable Stores
	+ Auto-subscriptions
	+ Readable Stores
	+ Derived Stores
	+ Custom Stores
	+ Store Bindings
	+ Page Store
	+ Navigation Store
	+ Updated Store
* Routing
	+ What is Routing?
	+ What problem does Routing solve?
	+ Pages
	+ Layout
	+ Route Parameters
	+ API Routes
* Errors and Redirects
	+ Handling Errors and Redirects
	+ Error Pages
	+ Fallback Errors
	+ Redirects
* Asynchronous Data
	+ Promises & async/await
	+ Fetching data from a REST API
	+ Subscriptions
	+ Stores
* Conclusion