Svelte Compiled UI framework

START HERE

`npx sv create <dir>`
New Svelte(Kit) app

`let { a } = $props()`
Component inputs

`npm run dev`
Dev server on :5173

`onclick={fn}`
Handle an event

`let n = $state(0)`
Reactive state

`{#each list as x}`
Loop in markup

`let x = $derived(n * 2)`
Computed value

`src/routes/+page.svelte`
A page

CONCEPTS

Svelte is a compiler: .svelte files hold script, markup and scoped style. Reactivity uses runes ($-prefixed), which are language keywords and need no import.

SvelteKit routing, load, SSR, adapters
Vite dev server and build
Svelte components, runes

This sheet follows Svelte 5.57 and SvelteKit 3. On SvelteKit 2, config is in svelte.config.js, $lib and $app/stores still exist, and matchers live in src/params/; npx sv migrate sveltekit-3 upgrades.

PROJECT (SV CLI)

`npx sv create <dir>`
Scaffold a new project

`npx sv create --template minimal --types ts <dir>`
Skip the prompts for template and types
Templates: minimal, demo, library, addon. Types: ts or jsdoc.

`npx sv add`
Pick add-ons interactively

`npx sv add tailwindcss`
Add an official add-on
Official: eslint, prettier, vitest, playwright, tailwindcss, drizzle, mdsvex, paraglide, storybook, sveltekit-adapter, enhanced-img, better-auth.

`npm i -D svelte-check`
Install the checker once

`npx sv check`
Type and a11y check the project

`npx sv check --watch`
Re-check on every change

`npx sv check --threshold error`
Show errors only

`npx sv check --fail-on-warnings`
Exit non-zero on warnings (CI)

`npx sv migrate`
Pick a migration interactively

`npx sv migrate sveltekit-3`
Upgrade a Kit 2 app to Kit 3

DEV AND BUILD

`npm run dev`
Start the Vite dev server

`npm run dev -- --open`
Dev server and open the browser

`npm run dev -- --host`
Expose the server on the network

DEV AND BUILD (cont.)

`npm run build`
Production build (runs the adapter)

`npm run preview`
Serve the production build locally

A SvelteKit app is a Vite project: scripts are plain vite dev, vite build and vite preview. Output depends on the adapter, preview is not a production server.

COMPONENTS

`<script lang="ts">`
Enable TypeScript in a component

`<style>`
Scoped CSS for this component

`<script module>`
Code run once per module, not per instance

`<Child {prop} />`
Pass a variable named prop

`<Child {...props} />`
Spread props

`{@render children?.()}`
Render nested content

`{#snippet name(arg)}…{/snippet}`
Reusable markup chunk

`{@render name(arg)}`
Render a snippet

`<svelte:head>`
Add to the document head

RUNES

`let n = $state(0)`
Reactive state
Arrays and plain objects become deep proxies; class instances do not.

`let o = $state.raw({…})`
Shallow state, reassign only

`$state.snapshot(o)`
Plain copy of a state proxy

`let d = $derived(expr)`
Value computed from state

`let d = $derived.by(() => {…})`
Derived with statements

`$effect(() => {…})`
Run after DOM updates, on changes
Not run during SSR. Reads after await or in timers are not tracked.

`$effect(() => { return () => {…} })`
Cleanup before re-run or destroy

`let { a, b = 1 } = $props()`
Props with a default

RUNES (cont.)

`let { a, ...rest } = $props()`
Collect remaining props

`let { v = $bindable() } = $props()`
Allow bind: from the parent

Prefer $derived over setting state inside $effect. Reach for $effect only for side effects like DOM, timers or subscriptions.

TEMPLATE SYNTAX

`{expr}`
Output a value

`{#if c}…{:else if d}…{:else}…{/if}`
Conditional block

`{#each items as item, i (item.id)}`
Loop with index and key

`{#await p}…{:then v}…{:catch e}…{/await}`
Await a promise

`onclick={fn}`
Event attribute, no on: prefix

`onclick={() => n++}`
Inline handler

`class={{ active: on, big }}`
Conditional classes from an object

BINDINGS

`bind:value={v}`
Two-way input value

`bind:checked={b}`
Checkbox state

`bind:this={el}`
Reference to an element

SVELTEKIT ROUTES

`src/routes/+page.svelte`
Page for this URL

`src/routes/+layout.svelte`
Wrapper; must render children

`+page.js / +page.server.js`
load function (universal / server)

`+layout.js / +layout.server.js`
Load data for a layout and children

`+server.js`
API endpoint (GET, POST, ...)

`+error.svelte`
Error page for load errors

`src/routes/blog/[slug]`
Dynamic segment

`src/hooks.server.js`
handle, handleError, handleFetch

SVELTEKIT ROUTES (cont.)

`static/`
Files served as-is

`src/lib/`
Shared code, server/ is server-only

`import x from '#lib/x.js'`
Import from lib (Kit 3; $lib in Kit 2)

DATA AND FORMS

`export const load = ({ params, fetch }) => {…}`
Load data for the page

`let { data } = $props()`
Receive load data in the page

`export const actions = { default }`
Form actions in +page.server.js

`<form method="POST" action="?/name">`
Post to a named action

`use:enhance`
Progressive form submission

`redirect(303, '/path')`
Redirect from load or action

`goto('/path')`
Navigate from code ($app/navigation)

`import { page } from '$app/state'`
Reactive page url, params, data

RECIPES

New app, ready to check

```
npx sv create --template
minimal --types ts my-app
cd my-app
npm i -D svelte-check
npm run dev -- --open
```

Counter component

```
<script>
  let count = $state(0);
  let double = $derived(count *
2);
</script>
<button onclick={() => count++}
>{count} / {double}</button>
```

Page with server load

```
// src/routes/post/[id]/+page.
server.js
export const load = async
({ params, fetch }) => {
  const res = await fetch(`/
api/posts/${params.id}`);
  return { post: await
res.json() };
};
```

Form action with validation

```
// +page.server.js
export const actions = {
  default: async ({ request })
=> {
    const email = (await
request.formData()).get(
'email');
    if (!email) return
fail(400, { missing: true });
  }
};
```