

Svelte Quick Reference

Components, reactivity, stores, transitions, SvelteKit

Components

Basic Component

```
<script>
  let name = "world";
</script>
<h1>Hello {name}</h1>
<style>
  h1 { color: purple; }
</style>
```

Component Structure

<script>	Component logic (JS/TS)
Markup	HTML template with {expressions}
<style>	Scoped CSS (auto-scoped to component)
<script context="module">	Runs once per module, not per instance

Reactivity

Reactive Assignments

```
<script>
  let count = 0;
  function increment() { count += 1; } // triggers re-render
</script>
<button on:click={increment}>{count}</button>
```

Reactive Declarations

```
<script>
  let width = 10;
  let height = 5;
  $: area = width * height; // recomputes when deps change
  $: console.log("area is", area); // reactive statement
</script>
```

\$: marks reactive declarations and statements

Reactive Rules

Assignment triggers	count += 1 triggers update; obj.x = 1 does too
Array mutation	Use arr = [...arr, item] (reassign to trigger)
\$: declaration	Auto-recomputes when referenced vars change
\$: statement	Runs side effects reactively

Props

Declaring & Passing Props

```
<!-- Child.svelte -->
<script>
  export let name;
  export let greeting = "Hello"; // default value
</script>
<p>{greeting}, {name}</p>
```

```
<!-- Parent.svelte -->
<Child name="Alice" />
```

Spread Props

```
<script>
  const props = { name: "Alice", greeting: "Hi" };
</script>
<Child {...props} />
```

Events

DOM Events

```
<button on:click={handleClick}>Click</button>
<input on:input={e => value = e.target.value} />
<form on:submit|preventDefault={handleSubmit}>
```

Event Modifiers

preventDefault	Calls e.preventDefault()
stopPropagation	Stops event bubbling
once	Handler fires only once
self	Only if event.target is the element
capture	Use capture phase

Component Events

```
<!-- Child.svelte -->
<script>
  import { createEventDispatcher } from "svelte";
  const dispatch = createEventDispatcher();
</script>
<button on:click={() => dispatch("greet", { text: "hi" })}>
```

```
<!-- Parent.svelte -->
<Child on:greet={(e) => alert(e.detail.text)} />
```

Bindings

Two-Way Binding

```
<input bind:value={name} />
<input type="checkbox" bind:checked={agreed} />
<select bind:value={selected}>
  <option value="a">A</option>
</select>
```

Element & Component Bindings

```
<div bind:this={element}></div>
<canvas bind:clientWidth={w} bind:clientHeight={h}></canvas>
<Child bind:value={childValue} />
```

Binding Types

bind:value	Input/select/textarea value
bind:checked	Checkbox state
bind:group	Radio/checkbox group
bind:this	DOM element reference
bind:clientWidth/Height	Read-only element dimensions

Stores

Writable Store

```
// store.js
import { writable } from "svelte/store";
export const count = writable(0);

// Component - auto-subscribe with $
<script>
  import { count } from "../store.js";
</script>
<button on:click={() => $count += 1}>{$count}</button>
```

Store Methods

```
count.set(10); // set value
count.update(n => n + 1); // update from current
const unsub = count.subscribe(v => console.log(v));
```

Store Types

writable(val)	Read-write store
readable(val, fn)	Read-only, set by start function
derived(stores, fn)	Computed from other stores
\$store	Auto-subscribe syntax in components

Transitions

Built-in Transitions

```
<script>
  import { fade, slide, fly } from "svelte/transition";
  let visible = true;
</script>
{#if visible}
  <div transition:fade>Fades in/out</div>
  <div in:fly={{ y: 200 }} out:fade>Fly in, fade out</div>
{/if}
```

Transition Options

fade	Opacity 0 to 1
fly	Animate x/y offset + opacity
slide	Slide in/out (height)
scale	Scale and fade
draw	SVG path stroke animation
duration	Transition time in ms
delay	Delay before start

Slots

Default & Named Slots

```
<!-- Card.svelte -->
<div class="card">
  <slot name="header">Default header</slot>
  <slot>Default content</slot>
</div>
```

```
<!-- Usage -->
<Card>
  <h2 slot="header">Title</h2>
  <p>Body content goes here</p>
</Card>
```

Slot Props

```
<!-- List.svelte -->
{#each items as item}
  <slot {item} index={item.id} />
{/each}
```

```
<!-- Usage -->
<List {items} let:item let:index>
  <p>{index}: {item.name}</p>
</List>
```

Context

Set & Get Context

```
<!-- Parent.svelte -->
<script>
  import { setContext } from "svelte";
  setContext("theme", { color: "dark" });
</script>
```

```
<!-- Descendant.svelte -->
<script>
  import { getContext } from "svelte";
  const theme = getContext("theme"); // { color: "dark" }
</script>
```

Context vs Stores

Context	Component-tree scoped, not reactive by default
Stores	Global, reactive, importable anywhere
Context + Store	Pass a store via context for scoped reactivity

Svelte Quick Reference

SvelteKit Basics

File-Based Routing

```
src/routes/  
+page.svelte      <!-- / -->  
about/+page.svelte <!-- /about -->  
blog/[slug]/+page.svelte <!-- /blog/:slug -->
```

Load Functions

```
// +page.js (runs on client & server)  
export async function load({ params, fetch }) {  
  const res = await fetch(`/api/posts/${params.slug}`);  
  return { post: await res.json() };  
}
```

Key Files

+page.svelte	Page component
+page.js / +page.ts	Client/universal load function
+page.server.js	Server-only load / form actions
+layout.svelte	Shared layout wrapper
+error.svelte	Error page
+server.js	API endpoint (GET, POST, ...)