

# Tham Khảo Nhanh Svelte

Components, reactivity, stores, transitions, SvelteKit

## Components

### Component cơ bản

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

### Cấu trúc Component

<b>&lt;script&gt;</b>	Logic component (JS/TS)
<b>Markup</b>	Template HTML với <b>{expressions}</b>
<b>&lt;style&gt;</b>	CSS có phạm vi (tự động giới hạn trong component)
<b>&lt;script context="module"&gt;</b>	Chạy một lần mỗi module, không phải mỗi instance

## Reactivity

### Gán giá trị có tính phản ứng

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

### Khai báo có tính phản ứng

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

\$: đánh dấu các khai báo và câu lệnh có tính phản ứng

### Quy tắc Reactivity

<b>Assignment triggers</b>	<b>count += 1</b> kích hoạt cập nhật; <b>obj.x = 1</b> cũng vậy
<b>Array mutation</b>	Dùng <b>arr = [...arr, item]</b> (gán lại để kích hoạt)
<b>\$. declaration</b>	Tự động tính lại khi các biến phụ thuộc thay đổi
<b>\$. statement</b>	Chạy side effects có tính phản ứng

## Props

### Khai báo & Truyền 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

<b>preventDefault</b>	Gọi <b>e.preventDefault()</b>
<b>stopPropagation</b>	Ngừng event bubbling
<b>once</b>	Handler chỉ kích hoạt một lần
<b>self</b>	Chỉ khi <b>event.target</b> là chính phần tử đó
<b>capture</b>	Dùng 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

### Binding hai chiều

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

### Binding phần tử & Component

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

### Các loại Binding

<b>bind:value</b>	Giá trị input/select/textarea
<b>bind:checked</b>	Trạng thái checkbox
<b>bind:group</b>	Nhóm radio/checkbox
<b>bind:this</b>	Tham chiếu phần tử DOM
<b>bind:clientWidth/Height</b>	Kích thước phần tử chỉ đọc

## 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>
```

### Phương thức Store

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

### Các loại Store

<b>writable(val)</b>	Store đọc-ghi
<b>readable(val, fn)</b>	Chỉ đọc, thiết lập bởi hàm start
<b>derived(stores, fn)</b>	Tính toán từ các stores khác
<b>\$store</b>	Cú pháp auto-subscribe trong components

## Transitions

### Transitions tích hợp

```
<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}
```

### Tùy chọn Transition

<b>fade</b>	Opacity từ 0 đến 1
<b>fly</b>	Hoạt ảnh offset x/y + opacity
<b>slide</b>	Trượt vào/ra (chiều cao)
<b>scale</b>	Phóng to và mờ dần
<b>draw</b>	Hoạt ảnh stroke SVG path
<b>duration</b>	Thời gian transition tính bằng ms
<b>delay</b>	Độ trễ trước khi bắt đầu

## Slots

### Slot mặc định & có tên

```
<!-- 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>
```

# Tham Khảo Nhanh Svelte

## Context vs Stores

<b>Context</b>	Phạm vi theo cây component, không có tính phản ứng mặc định
<b>Stores</b>	Toàn cục, có tính phản ứng, có thể import mọi nơi
<b>Context + Store</b>	Truyền store qua context để có reactivity theo phạm vi

## Cơ bản về SvelteKit

### Routing theo file

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

### Hàm Load

```
// +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() };  
}
```

### Các file quan trọng

<b>+page.svelte</b>	Component trang
<b>+page.js / +page.ts</b>	Hàm load phía client/universal
<b>+page.server.js</b>	Load chỉ phía server / form actions
<b>+layout.svelte</b>	Wrapper layout dùng chung
<b>+error.svelte</b>	Trang lỗi
<b>+server.js</b>	API endpoint (GET, POST, ...)