

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

<script>	Logic component (JS/TS)
Markup	Template HTML với {expressions}
<style>	CSS có phạm vi (tự động giới hạn trong component)
<script context="module">	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

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

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

bind:value	Giá trị input/select/textarea
bind:checked	Trạng thái checkbox
bind:group	Nhóm radio/checkbox
bind:this	Tham chiếu phần tử DOM
bind:clientWidth/Height	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

writable(val)	Store đọc-ghi
readable(val, fn)	Chỉ đọc, thiết lập bởi hàm start
derived(stores, fn)	Tính toán từ các stores khác
\$store	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

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

Context	Phạm vi theo cây component, không có tính phản ứng mặc định
Stores	Toàn cục, có tính phản ứng, có thể import mọi nơi
Context + Store	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

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