SVELTE クイックリファレンス

コンポーネント、リアクティビティ、ストア、トランジション、SvelteKit

コンポーネント

基本コンポーネント

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

コンポーネント構造

`<script>`	コンポーネントロジック（JS/TS）
`Markup`	`{expressions}` を含む HTML テンプレート
`<style>`	スコープ付き CSS（コンポーネントに自動スコープ）
`<script context="module">`	インスタンスごとではなく、モジュールごとに一度実行

リアクティビティ

リアクティブな代入

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

リアクティブ宣言

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

$: はリアクティブ宣言と文をマークします

リアクティビティのルール

`Assignment triggers`	`count += 1` で更新がトリガーされます。`obj.x = 1` も同様
`Array mutation`	`arr = [...arr, item]` を使う（再代入でトリガー）
`$: declaration`	参照する変数が変わると自動再計算
`$: statement`	リアクティブに副作用を実行

プロップ

プロップの宣言と渡し方

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

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

プロップのスプレッド

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

イベント

DOM イベント

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

イベント修飾子

`preventDefault`	`e.preventDefault()` を呼び出す
`stopPropagation`	イベントのバブリングを停止
`once`	ハンドラーが一度だけ発火
`self`	`event.target` が要素自身のときのみ
`capture`	キャプチャフェーズを使用

コンポーネントイベント

```
<!-- 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)} />
```

バインディング

双方向バインディング

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

要素とコンポーネントのバインディング

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

バインディングの種類

`bind:value`	input/select/textarea の値
`bind:checked`	チェックボックスの状態
`bind:group`	ラジオ/チェックボックスグループ
`bind:this`	DOM 要素の参照
`bind:clientWidth/Height`	読み取り専用の要素サイズ

ストア

書き込み可能ストア

```
// 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>
```

ストアのメソッド

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

ストアの種類

`writable(val)`	読み書き可能ストア
`readable(val, fn)`	読み取り専用、開始関数でセット
`derived(stores, fn)`	他のストアから計算
`$store`	コンポーネント内の自動サブスクライブ構文

トランジション

組み込みトランジション

```
<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}
```

トランジションオプション

`fade`	不透明度 0 から 1
`fly`	x/y オフセットと不透明度をアニメーション
`slide`	スライドイン/アウト（高さ）
`scale`	スケールとフェード
`draw`	SVG パスのストロークアニメーション
`duration`	トランジション時間（ms）
`delay`	開始前の遅延

スロット

デフォルトスロットと名前付きスロット

```
<!-- 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>
```

スロットプロップ

```
<!-- 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>
```

コンテキスト

コンテキストのセットと取得

```
<!-- 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>
```

コンテキストとストアの比較

`Context`	コンポーネントツリースコープ、デフォルトでリアクティブでない
`Stores`	グローバル、リアクティブ、どこでもインポート可能
`Context + Store`	コンテキスト経由でストアを渡しスコープ付きリアクティビティを実現

SvelteKit の基本

ファイルベースルーティング

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

ロード関数

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

主要ファイル

`+page.svelte`	ページコンポーネント
`+page.js / +page.ts`	クライアント/ユニバーサルロード関数
`+page.server.js`	サーバー専用ロード/フォームアクション
`+layout.svelte`	共有レイアウトラッパー
`+error.svelte`	エラーページ
`+server.js`	API エンドポイント（GET、POST など）