

Svelte 快速参考

组件、响应式、Store、过渡动画与 SvelteKit

组件

基础组件

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

组件结构

<script>	组件逻辑 (JS/TS)
Markup	含 {表达式} 的 HTML 模板
<style>	作用域 CSS (自动限定到组件)
<script context="module">	每个模块只运行一次, 不随实例重复

响应式

响应式赋值

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

响应式声明

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

\$: 标记响应式声明和语句

响应式规则

Assignment triggers	count += 1 触发更新; obj.x = 1 也会触发
Array mutation	使用 arr = [...arr, item] (重新赋值以触发更新)
\$: declaration	引用变量变化时自动重新计算
\$: statement	响应式地执行副作用

Props

声明与传递 Props

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

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

展开 Props

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

事件

DOM 事件

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

事件修饰符

preventDefault	调用 e.preventDefault()
stopPropagation	阻止事件冒泡
once	处理器只触发一次
self	仅当 event.target 为该元素时触发
capture	使用捕获阶段

组件事件

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

绑定

双向绑定

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

元素与组件绑定

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

绑定类型

bind:value	input/select/textarea 值
bind:checked	复选框状态
bind:group	单选/复选框组
bind:this	DOM 元素引用
bind:clientWidth/Height	只读元素尺寸

Store

可写 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 方法

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

Store 类型

writable(val)	可读写 Store
readable(val, fn)	只读, 由 start 函数设置
derived(stores, fn)	从其他 Store 派生计算
\$store	组件中的自动订阅语法

过渡动画

内置过渡

```
<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}
```

过渡选项

fade	透明度 0 到 1
fly	x/y 偏移 + 透明度动画
slide	滑入/滑出 (高度)
scale	缩放并淡入淡出
draw	SVG 路径描边动画
duration	过渡时长 (毫秒)
delay	开始前的延迟

插槽

默认与具名插槽

```
<!-- 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>
```

插槽 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

设置与获取 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 Store

Context	限定于组件树作用域, 默认不响应式
Stores	全局、响应式, 可在任何地方导入
Context + Store	通过 Context 传递 Store, 实现作用域响应式

Svelte 快速参考

SvelteKit 基础

基于文件的路由

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

Load 函数

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

关键文件

+page.svelte	页面组件
+page.js / +page.ts	客户端/通用 load 函数
+page.server.js	仅服务端 load / 表单 action
+layout.svelte	共享布局包装器
+error.svelte	错误页面
+server.js	API 端点 (GET、POST 等)