

# REFERÊNCIA RÁPIDA DO SVELTE

Componentes, reatividade, stores, transições, SvelteKit

## Componentes

### Componente Básico

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

### Estrutura do Componente

```
<script> Lógica do componente (JS/TS)
Markup Template HTML com `expressões`
```

### <style>

CSS com escopo (automaticamente restrito ao componente)

<script context="module"> Executado uma vez por módulo, não por instância

## Reatividade

### Atribuições Reativas

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

### Declarações Reativas

```
<script>
  let width = 10;
  let height = 5;
  $: area = width * height; // recalcula quando deps mudam
  $: console.log("area is", area); // instrução reativa
</script>
```

\$: marca declarações e instruções reativas

### Regras de Reatividade

#### Atribuição aciona

`count += 1` aciona atualização; `obj.x = 1` também

#### Mutação de array

Use `arr = [...arr, item]` (reatribuir para acionar)

#### \$. declaração

Recalcula automaticamente quando variáveis referenciadas mudam

#### \$. instrução

Executa efeitos colaterais reativamente

## Props

### Declarando e Passando Props

```
<!-- Child.svelte -->
<script>
  export let name;
  export let greeting = "Hello"; // valor padrão
</script>
<p>{greeting}, {name}</p>
<!-- Parent.svelte -->
<Child name="Alice" />
```

### Spread de Props

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

## Eventos

### Eventos DOM

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

### Modificadores de Evento

**preventDefault** Chama `e.preventDefault()`

**stopPropagation** Para a propagação do evento

**once** Handler é disparado apenas uma vez

**self** Apenas se `event.target` for o elemento

**capture** Usar fase de captura

### Eventos de Componente

```
<!-- 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 Bidirecional

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

### Bindings de Elemento e Componente

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

### Tipos de Binding

**bind:value** Valor de input/select/textarea

**bind:checked** Estado da caixa de seleção

**bind:group** Grupo de rádio/caixa de seleção

**bind:this** Referência ao elemento DOM

**bind:clientWidth/Height** Dimensões do elemento (somente leitura)

## Stores

### Store Gravável

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

```
// Componente - assinar automaticamente com $
<script>
  import { count } from "./store.js";
</script>
<button on:click={() => $count += 1}>{$count}</button>
```

### Métodos do Store

```
count.set(10); // definir valor
count.update(n => n + 1); // atualizar a partir do atual
const unsub = count.subscribe(v => console.log(v));
```

### Tipos de Store

**writable(val)** Store de leitura e escrita

**readable(val, fn)** Somente leitura, definido pela função de início

**derived(stores, fn)** Computado a partir de outros stores

**\$store** Sintaxe de assinatura automática em componentes

## Transições

### Transições Internas

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

### Opções de Transição

**fade** Opacidade de 0 a 1

**fly** Anima deslocamento x/y + opacidade

**slide** Desliza para dentro/fora (altura)

**scale** Escala e desvanece

**draw** Animação de traço de caminho SVG

**duration** Duração da transição em ms

**delay** Atraso antes de iniciar

## Slots

### Slots Padrão e Nomeados

```
<!-- Card.svelte -->
<div class="card">
  <slot name="header">Cabeçalho padrão</slot>
  <slot>Conteúdo padrão</slot>
</div>
<!-- Uso -->
<Card>
  <h2 slot="header">Título</h2>
  <p>Conteúdo do corpo aqui</p>
</Card>
```

### Props de Slot

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

## Contexto

### Definir e Obter Contexto

```
<!-- 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>
```

### Contexto vs Stores

**Contexto** Com escopo na árvore de componentes, não reativo por padrão

**Stores** Global, reativo, importável em qualquer lugar

**Contexto + Store** Passar um store via contexto para reatividade com escopo

## Básico do SvelteKit

### Roteamento Baseado em Arquivo

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

### Funções de Carregamento

```
// +page.js (executado no cliente e no servidor)
export async function load({ params, fetch }) {
  const res = await fetch(`/api/posts/${params.slug}`);
  return { post: await res.json() };
}
```

### Arquivos Principais

**+page.svelte** Componente de página

**+page.js / +page.ts** Função de carregamento cliente/universal

**+page.server.js** Carregamento somente no servidor / ações de formulário

**+layout.svelte** Wrapper de layout compartilhado

**+error.svelte** Página de erro

**+server.js** Endpoint de API (GET, POST, ...)