Referencia Rápida de Svelte

Componentes, reactividad, stores, transiciones, SvelteKit

Componentes

Componente Básico

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

Estructura del Componente

`<script>`	Lógica del componente (JS/TS)
`Markup`	Plantilla HTML con **`{expresiones}`**
`<style>`	CSS con ámbito (auto-delimitado al componente)
`<script context="module">`	Se ejecuta una vez por módulo, no por instancia

Reactividad

Asignaciones Reactivas

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

Declaraciones Reactivas

```
<script>
  let width = 10;
  let height = 5;
  $: area = width * height;       // recalcula cuando cambian las
dependencias
  $: console.log("area is", area); // instrucción reactiva
</script>
```

$: marca declaraciones e instrucciones reactivas

Reglas de Reactividad

`La asignación dispara`	**`count += 1`** dispara actualización; **`obj.x = 1`** también
`Mutación de arrays`	Usar **`arr = [...arr, item]`** (reasignar para disparar)
`Declaración $:`	Se recalcula automáticamente cuando cambian las variables referenciadas
`Instrucción $:`	Ejecuta efectos secundarios de forma reactiva

Props

Declarar y Pasar Props

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

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

Props Expandidas

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

Eventos

Eventos del DOM

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

Modificadores de Eventos

`preventDefault`	Llama a **`e.preventDefault()`**
`stopPropagation`	Detiene la propagación del evento
`once`	El manejador se dispara solo una vez
`self`	Solo si **`event.target`** es el 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

Enlace Bidireccional

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

Bindings de Elementos y Componentes

```
<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 del checkbox
`bind:group`	Grupo de radio/checkbox
`bind:this`	Referencia al elemento DOM
`bind:clientWidth/Height`	Dimensiones del elemento (solo lectura)

Stores

Store Escribible

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

// Componente — suscripción automática con $
<script>
  import { count } from "./store.js";
</script>
<button on:click={() => $count += 1}>{$count}</button>
```

Métodos del Store

```
count.set(10);              // establecer valor
count.update(n => n + 1);   // actualizar desde el valor actual
const unsub = count.subscribe(v => console.log(v));
```

Tipos de Store

`writable(val)`	Store de lectura-escritura
`readable(val, fn)`	Solo lectura, establecido por función de inicio
`derived(stores, fn)`	Calculado a partir de otros stores
`$store`	Sintaxis de suscripción automática en componentes

Transiciones

Transiciones Incorporadas

```
<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>Vuela al entrar, desaparece al
salir</div>
{/if}
```

Opciones de Transición

`fade`	Opacidad de 0 a 1
`fly`	Anima desplazamiento x/y + opacidad
`slide`	Desliza entrada/salida (altura)
`scale`	Escalar y desvanecer
`draw`	Animación de trazo de ruta SVG
`duration`	Tiempo de transición en ms
`delay`	Retraso antes de iniciar

Slots

Slots por Defecto y con Nombre

```
<!-- Card.svelte -->
<div class="card">
  <slot name="header">Cabecera por defecto</slot>
  <slot>Contenido por defecto</slot>
</div>

<!-- Uso -->
<Card>
  <h2 slot="header">Título</h2>
  <p>El contenido del cuerpo va aquí</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

Establecer y Obtener 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>
```

Referencia Rápida de Svelte

Contexto vs Stores

`Contexto`	Con ámbito de árbol de componentes, no reactivo por defecto
`Stores`	Global, reactivo, importable desde cualquier lugar
`Contexto + Store`	Pasar un store vía contexto para reactividad con ámbito

Conceptos Básicos de SvelteKit

Enrutamiento Basado en Archivos

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

Funciones de Carga

```
// +page.js (se ejecuta en cliente y servidor)
export async function load({ params, fetch }) {
  const res = await fetch(`/api/posts/${params.slug}`);
  return { post: await res.json() };
}
```

Archivos Clave

`+page.svelte`	Componente de página
`+page.js / +page.ts`	Función de carga universal/cliente
`+page.server.js`	Carga solo en servidor / acciones de formulario
`+layout.svelte`	Envoltorio de diseño compartido
`+error.svelte`	Página de error
`+server.js`	Endpoint de API (GET, POST, ...)