Spickzettel („Cheat Sheet“): Single Page Web Apps mit Svelte und SvelteKit

Autor: Dr. Holger Schwichtenberg (www.IT-Visions.de)

www.IT-Visions.de
Dr. Holger Schwichtenberg

Webadressen

Produkt-Website zu Svelte	https://svelte.dev
Produkt-Website zu SvelteKit	https://kit.svelte.dev
GitHub-Projekte	https://github.com/sveltejs
Svelte-Chat auf Discord	https://discord.com/invite/svelte
Svelte-Forum	https://dev.to/t/svelte

Installation

Node.js: nodejs.org

Visual Studio Code: code.visualstudio.com

Svelte-Erweiterungen für Visual Studio Code

- marketplace.visualstudio.com/items?itemName=svelte.svelte-vscode
- marketplace.visualstudio.com/items?itemName=ardenivanov.svelte-intellisense
- marketplace.visualstudio.com/items?itemName=fivethree.vscode-svelte-snippets

Vite global installieren oder aktualisieren: npm install -g vite

Kommandozeilenbefehle

Svelte-Projekt ohne SvelteKit anlegen *npm init vite*

Svelte-Projekt mit SvelteKit anlegen

npm create svelte@latest ProjektName

Alle in package.json gelisteten Pakete installieren

npm install

Alle in package.json gelisteten NPM-Pakete aktualisieren

npm update

Kompilieren und Dev-Webserver mit Hot Reload starten

npm run dev

Kompilieren und Dev-Webserver mit Hot Reload + Browser starten

npm run dev -- --open

Statische Codeanalyse: npm run lint

Unit Tests starten: npm run test:unit

End-To-End-Tests (Browser-UI-Tests) starten: npm run test:e2e

Minifiziertes Kompilat für die Produktion: npm run build

Minifiziertes Kompilat starten: vite preview

Svelte-Komponente MeineKomponente.svelte

```
<!-- Maximal ein Skriptblock pro Komponente! -->
<script lang="ts">
 import { onMount, createEventDispatcher } from 'svelte';
 // --- notwendig für das Auslösen von Events
 const dispatch = createEventDispatcher();

 // --- Lebenszyklusereignis
 onMount(() => { console.log('onMount'); });

 // --- Komponentenparameter und interner Zustand
 export let zaehler: number;

 // --- Berechnete Werte
 $: ergebnis = Math.sqrt(zaehler);
 // --- Reaktiver Codeblock (aufgerufen, wenn Variable sich ändert!)
 $: { if(zaehler%10===0) console.log("Aktueller Wert: ", zaehler); }

 // --- Benutzerinteraktion mit Auslösen der Komponentenereignisse
 // --- export macht diese Funktion auch für Host zugänglich!
 export function Erhoehen(x : number = 1) {
  dispatch('vorErhoehung', zaehler);
  zaehler+=x;
  console.log("zaehler erhöht auf " + zaehler, x);
  dispatch('nachErhoehung', { zaehler, ergebnis });
 }
</script>

<div>
 <input type="number" bind:value={zaehler} />
 <button on:click={()=>Erhoehen(1)}>+1</button>
   <div class="ergebnis">Quadratwurzel: {ergebnis.toFixed(2)}</div>
</div>
<!-- Scoped CSS Style -->
<style>
    .ergebnis { color: blue; }
</style>
```

Svelte-Komponente HostFuerMeineKomponente.svelte

```
<script lang="ts">
 // Versionsnummern einbinden
 import { VERSION } from 'svelte/compiler';
 // Unterkomponente einbinden
 import MeineKomponente from './MeineKomponente.svelte';

 let startWert: number = 10;
 let ausgabe: string = '';

 // Behandlung der Ereignisse der Unterkomponente
 function log(text: string, a: number | null = null, b: number | null =
null) {
  console.log(text, a, b);
  ausgabe += '<li>' + text + '/' + a + '/' + b + '</li>'; }

 // Aufruf einer Methode in der Unterkomponente
 let ErhoehenInMeineKomponenten : (x) => { };
 function callMethodInSubComponent() {
  ErhoehenInMeineKomponenten(10);
 }
</script>

<div class="box">
 <h4>Svelte-Version: {VERSION}</h4>
 <div>
  Startwert: <input type="number" bind:value={startWert} />
  Methode in Komponente aufrufen:
  <button on:click={callMethodInSubComponent}>+10</button>
 </div>

  <!-- Unterkomponente via Tag nutzen -->
  <MeineKomponente
   zaehler={startWert}
   on:vorErhoehung={(e) => log('vorher', e.detail)}
   on:nachErhoehung={
              (e) => log('nachher', e.detail.zaehler, e.detail.ergebnis )}
   bind:Erhoehen={ErhoehenInMeineKomponenten} />

  Meldungen der Unterkomponente (hier legt log() li-Elemente rein):
 <ul>{@html ausgabe}</ul>
</div>
```

Debugging

Anhalten im Debugger, wenn Variable x sich ändert

```
{@debug x}
```

Dateibasiertes Routing in SvelteKit

Dateipfad	Relative URL
/src/routes/+page.svelte	/ (Wurzel-URL)
/src/routes/x/y/z/+page.svelte	/x/y/z
/src/routes/booking/[pid]/[fid]/+page.svelte	/booking/12/34

Datei +layout.svelte mit <slot> ist Masterpage für alle untergeordneten +page.svelte-Dateien. Verschachtelung möglich!

Navigation per Link: `<a href="/x/y/z">Go!</a>`

Navigation per Code:

```
import { goto } from '$app/navigation';
…
<button on:click={()=>goto('/x/y/z')}>Go!</button>
```

Aktuelle URL ermitteln:

```
import { page } from '$app/stores';
Komplette URL: {$page.url}
Relative URL: {$page.url.pathname}
Querystring: {$page.url.search}
```

Optionales Datenladen in +page.ts oder +page.server.ts

+page.ts: wird beim Server-Pre-Rendering & im Browser ausgeführt

+page.server.ts: wird nur beim Server-Pre-Rendering ausgeführt

```
import { error, type LoadEvent } from '@sveltejs/kit';
/** @type {import('./$types').PageLoad} */

export function load(e: LoadEvent) { // bzw: ServerLoadEvent
 const params: any = e.params;
 // Parameter prüfen
 if (params.pid <= 0 || params.fid <= 0) {
  throw error(404, 'Person or Flight not found');  }
 // Datenübergabe an Svelte-Komponente
 return {  pid: params.pid as number,
  fid: params.fid as number,
  text: "Zusatzdaten" };
}
```

Navigation per Redirect in +page.ts oder +page.server.ts

```
import { redirect } from '@sveltejs/kit';
…
throw redirect(302, '/zielseite');
```

Zugriff auf übergebene Daten in +page.svelte

```
import type { PageData } from './$types';
export let data : PageData;
```

Übergebene Daten dann in data.fid, data.pid und data.text

Über den Autor

Dr. Holger Schwichtenberg gehört zu den bekanntesten Experten für die Entwicklung von Web- und .NET-Anwendungen. Er entwickelt seit über 30 Jahren Software, hat über 90 Fachbücher veröffentlicht und spricht regelmäßig auf Fachkonferenzen. Sie können ihn und seine Kollegen für Schulungen, Beratungen und Softwareentwicklung buchen.
Softwareentwicklung: **www.MAXIMAGO.de**
Schulungen und Beratungen: **www.IT-Visions.de**

Spickzettel („Cheat Sheet“): Single Page Web Apps mit Svelte und SvelteKit

Autor: Dr. Holger Schwichtenberg (www.IT-Visions.de)

www.IT-Visions.de
Dr. Holger Schwichtenberg

Variablen im Svelte-Template (Interpolation)

Ausgabe mit Interpolationssyntax (HTML-Encoded)
```
<div>{ x } + { y } =<b>{ x + y }</b></div>
```
Ausgabe (nicht HTML-Encoded)
```
let htmlString = "Das &lt;b&gt;-Tag macht <b>fett</b>!";
…
<span>{@html htmlString}</span>
```
Interpolation mit ternärem Operator
```
<div>{ x > 0 ? x : "ungültiger Wert" }</div>
```
Interpolation mit Ausgabe von Objekteigenschaften
```
<div>{ obj.ID } { obj.Datum.getFullYear() }</div>
```
Escape für geschweifte Klammern { }
```
<div>Mit {'{'} x {'}'} geben sie x aus.</div>
```
Variablen in HTML-Attributen
```
<div title={`Summe aus ${x} und ${y}`}>{x + y}</div>
<a href={linkURL}>{linkText}</a>
```
Abkürzung, wenn Variable heißt wie Attribut. Statt href={href}:
```
 <a {href}>{linkText}</a>
```
Einsatz des Spread-Operators für Attributzuweisungen
```
const linkObj = { target: '_blank', href: 'https://www.IT-Visions.de' };
…
<a {...linkObj}>{linkText}</a>
```
Bei HTML-Attributen ohne Wert entfernt false den Wert
```
<button disabled={x < 0}>x ändern</button>
```

Funktions- und Methodenaufrufe im Template

Interpolation mit synchronem Funktionsaufruf
```
<div>Die Antwort auf alle Fragen: {Berechnung(42)}</div>
```
Interpolation mit Aufruf einer Objektmethode
```
<div>Person {obj.getInfo(true)}</div>
```
Asynchrone Funktionsaufrufe / Promises
```
{#await LangeBerechnung(123)} <span>Aktion läuft...</span>
{:then number}  <span>Das Ergebnis ist: {number}</span>
{:catch error} <span style="color: red">{error.message}</span>
{/await}
```

Bedingte Formatierung

Klassenzuweisung per Variable
```
let eineCSSKlasse = "fett"; let zweiCSSKlassen = ["fett", "rot"];
…
<div class={eineCSSKlasse}>{x}</div>
<div class={zweiCSSKlassen.join(' ')}>{x}</div>
```
Klassenzuweisung mit Ausdruck
```
<div class={x > 0 ? 'gruen' : 'rot'}>{x}</div>
```
Mischung aus statischen und bedingten Klassen
```
<div class="fett {x > 0 ? 'gruen' : 'rot'}">{x}</div>
```
Mischung aus statischen und bedingten Klassen (Alternative)
```
<div class="fett" class:even={x % 2 == 0} class:rot={x % 2 != 0}>
```
Bedingter Style: Wert aus Variable
```
let farbe = 'fuchsia';  … <div style:color={farbe}>{x}</div>
```
Bedingter Style mit Ausdruck
```
<div style:color={x > 0 ? 'green' : 'red'}>{x}</div>
```

Zwei-Wege-Datenbindung bei Eingabesteuerelementen

```
let x = 1234; let b = true; let g1 = 'A'; let g2 = ['A', 'C']; let wahl = 2;
```
Datenbindung an einfaches Eingabefeld
```
<input type="number" bind:value={x} />
```
Datenbindung an Kontrollkästchen
```
<input type="checkbox" bind:checked={b}>
```
Gruppierte Radio-Steuerelemente
```
<input type="radio" bind:group={g1} value="A" />A
<input type="radio" bind:group={g1} value="B" />B
<input type="radio" bind:group={g1} value="C" />C
```
Gruppierte Checkbox-Steuerelemente
```
<input type="checkbox" bind:group={g2} value="A" />A
<input type="checkbox" bind:group={g2} value="B" />B
<input type="checkbox" bind:group={g2} value="C" />C
```
Auswahlsteuerelement
```
<select bind:value={wahl}>
  {#each arrZahlen as z} <option value={z}>Zahl #{z}</option>
  {/each}
</select>
```

Bedingtes Rendering von DOM-Elementen mit #if

```
{#if x > 0}  <div>Gewinn {x} Euro </div>
{:else if x === 0}  <div>Kein Gewinn</div>
{:else}  <div>Verlust {x} Euro </div>
{/if}
```

Schleifen mit #each

Schleife über Array mit Zahlen
```
{#each arrZahlen as z} <li>{z}</li> {/each}
```
Schleife inkl. laufendem Index und Alternativtext
```
{#each arrZahlen as z, index} <li>{index + 1}. Zahl = {z}</li>
{:else}  Keine Zahlen vorhanden
{/each}
```
Schleife über Array von Objekten
```
{#each arrFirmen as o} <li>{o.Name}</li> {/each}
```
Schleife mit Key in Klammern ( ) für Beibehaltung von DOM-Knoten
```
{#each arrFirmen as o (o.id)} <li>{o.ID}: {o.Name}</li> {/each}
```
Schleife mit Destrukturierung
```
{#each arrFirmen as { ID, Name }} <li>{ID}: {Name}</li>{/each}
```
Schleife über Map
```
{#each [...mapFirmen] as [k, wert], index}
   <li>Firma #{index + 1}: {k}, {wert}</li>
{/each}
```
Schleife über alle Eigenschaften eines einzelnen Objekts
```
{#each Object.entries(objFirma) as [propName, wert]}
   <li>{propName} = {wert}</li>
{/each}
```

DOM-Ereignisse behandeln

Ereignis direkt ("inline") behandeln
```
<button on:click={() => counter += 10 }>+10</button>
```
Ganze Befehlsfolgen sind möglich (aber nicht übersichtlich)
```
<button on:click={() => {counter += 10; console.log('counter'); }}>
+10 </button>
```
Ereignis in eigener Funktion behandeln
```
<button on:click={erhoehe}>+1</button>
```
Bedingte Ereignisbehandlung
```
<button on:click={counter < 10 && erhoehe()}>+1</button>
```
Eigene Parameter an die Ereignisbehandlung übergeben
```
<button on:click={() => erhoeheUm(5)}>+5</button>
```
DOM-Ereignisparameter verwenden
```
<div on:mousemove={MausAktion}>  {position} </div>
```
DOM-Ereignisparameter und eigene Parameter verwenden
```
<div on:mousemove={(e) => MausAktion(e, 'Mauszeiger')}>
  {position} </div>
```
Mehrere Ereignisse können zum gleichen Handler führen
```
<div on:mousemove={MausAktion} on:mouseleave={MausAktion}>
  {position} </div>
```
Ereignisbehandlung bei Mausereignissen
```
let position = 'Bewege die Maus hier hin!';
function MausAktion(e: MouseEvent, text: string = 'Maus') {
  if (e.type === 'mouseleave') position = 'Maus außerhalb!';
  else position = text + ': ' + e.x + '/' + e.y;  }
```
Alle DOM-Events: wiki.selfhtml.org/wiki/JavaScript/DOM/Event

Modifikatoren für DOM-Ereignisse (Auswahl)

on:click|stopPropagation="Aktion"	Keine Ereignisweitergabe
on:submit|preventDefault="Aktion"	Keine Standardreaktion
on:mousemove|once="Aktion"	Nur einmalige Reaktion

Lebenszyklusereignisse

```
import { beforeUpdate,onMount,afterUpdate,onDestroy } from 'svelte';
```
z.B.: `beforeUpdate(() => { console.log('vor einem DOM-Update');  });`
Erstes Rendern: beforeUpdate() → onMount() → afterUpdate()
Weiteres Rendern: beforeUpdate() → afterUpdate()
Nur onDestroy() wird auch beim SSPR ausgeführt!

Vorlagenbasierte Komponente mit zwei Slots definieren

Expander.svelte
```
<script lang="ts">
 let eingeklappt = false;
 let umschalten = () => (eingeklappt = !eingeklappt);
</script>
<div class="card"> <div class="card-header">
  <h5 class="card-title" on:click={umschalten}>
   <span  class="oi {eingeklappt ? 'oi-caret-bottom' :
     'oi oi-caret-top'}"  />
    {#if $$slots.kopfzeile} <slot name="kopfzeile" />
    {:else} Kein Titel {/if} </h5>
  </div>
 {#if !eingeklappt}  <div class="card-body">  <slot />  </div> {/if}
</div>
```

Nutzung dieser vorlagenbasierten Komponente
```
import Expander from './Expander.svelte';
…
<Expander>
  <span slot="kopfzeile">Überschrift im Slot "Kopfzeile"</span>
  <p>Inhalt für unbenannten Haupt-Slot</p>  </Expander>
```