Spickzettel („Cheat Sheet“): Single Page Web Apps mit Vue.js 3 und Vue CLI oder Vite

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

www.IT-Visions.de
Dr. Holger Schwichtenberg

Installation

Node.js
nodejs.org
Visual Studio Code
code.visualstudio.com
Volar für Visual Studio Code
marketplace.visualstudio.com/items?itemName=johnsoncodehk.volar
Vue Developer Tools für Chrome und Firefox
github.com/vuejs/devtools
Vue CLI global installieren oder aktualisieren
npm install -g @vue/cli
oder: vite global installieren oder aktualisieren
npm install -g vite

Befehle der Vue CLI | Vite sofern abweichend

Versionsnummer der Vue CLI bzw. von Vite abfragen
vue --version | *vite --version*
Projekt anlegen (Projektname nur Kleinbuchstaben sowie – und _!)
vue create projektname | *npm init vue@3*
Alle in package.json gelisteten Pakete installieren
npm install
Alle in package.json gelisteten NPM-Pakete aktualisieren
npm update
Kompilieren und Dev-Webserver starten mit Hot Reload
npm run serve | *npm run dev*
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

Single-File-Component MeineKomponente.vue

```
<template>
 <div>
  <input type="number" v-model="zaehler" />
  <button @click="Erhoehe">+1</button>
  Quadratwurzel: {{ ergebnis.toFixed(2) }}
 </div>
</template>

<!-- Scoped CSS Style -->
<style scoped>  div { color: blue; }  </style>

<script setup lang="ts">
// Alle define…() sind Makros und müssen nicht importiert werden.
import { ref, onMounted, watch, computed } from 'vue';

//#region ### Öffentliche Schnittstelle der Komponente
// Komponentenparameter
const props = defineProps({
 startWert: Number })

// Optionale Reaktion auf durch Host geänderten Parameterwert
watch(() => props.startWert,
            (neu, alt) => { zaehler.value = neu || 0; });

// Komponentenereignisse für den Host
const emit = defineEmits<{
 (e: 'vorErhoehung', zaehler: number): void
 (e: 'nachErhoehung', zaehler: number, ergebnis: number): void}>()

// Veröffentlichung einer Methode
defineExpose({ Erhoehe });
//#endregion

// Reaktive Variable, initialisiert mit Parameter
let zaehler = ref(props.startWert || 0);
// Berechnete Werte
let ergebnis = computed(() => Math.sqrt(zaehler.value));

// Lebenszyklusereignis
onMounted(() => {
 console.log("OnMounted"); });

// Benutzerinteraktion und Auslösen der Komponentenereignisse
function Erhoehe() {
 emit("vorErhoehung", zaehler.value);
 zaehler.value++;
 console.log(zaehler.value);
 emit("nachErhoehung", zaehler.value, ergebnis.value); }
</script>
```

Single-File-Component HostFuerMeineKomponente.vue

```
<template>
 <h4>Vue.js {{ vueVersion }} -
       Anwendungsversion {{ appVersion }}</h4>
 Startwert: <input type="number" v-model="data.startWert" />
 Methode in Komponente aufrufen:
 <button @click="callMethodInSubComponent">+1</button>

 <!-- Unterkomponente via Tag nutzen -->
 <!-- alternativ lower Camel Casing: <meineKomponente> -->
 <!-- oder Kebab-Casing: <meine-komponente> (groß oder klein) -->
 <MeineKomponente
  ref="mk"
  :startWert="data.startWert"
  @vorErhoehung="(x) => log('vorher', x)"
  @nachErhoehung="(x, y) => log('nachher', x, y)"
  style="border-style: dotted;" />
 Meldungen der Unterkomponente (hier legt log() li-Elemente rein):
 <ul v-html="data.ausgabe"/>
</template>

<script setup lang="ts">
// Versionsnummern einbinden
import { version as vueVersion, ref, reactive } from 'vue';
import { version as appVersion } from '../../package.json'

// Unterkomponente einbinden
import MeineKomponente from
'@/components/MeineKomponente.vue';

// Reaktives Datenobjekt als Alternative zu einzelnen ref()-Variablen
const data = reactive({
 startWert: 10,
 ausgabe: ""   });

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

// Verweis auf Komponente (optional, nur für Methodenaufruf!)
const mk = ref<typeof MeineKomponente>();

// Aufruf einer Methode in der Unterkomponente
function callMethodInSubComponent() {
 if (mk.value) mk.value.Increment();      }
</script>
```

Routen in Vue.js-Router (/src/router/index.ts)

```
import HostFuerMeineKomponente
        from '../views/HostFuerMeineKomponente.vue'
…
const routes: Array<RouteRecordRaw> = [ {
    path: '/CheatSheet/HostFuerMeineKomponente', // Haupt-URL
    name: 'HostFuerMeineKomponente', // optional
    component: HostFuerMeineKomponente,
    alias: ['/Start','/a/b/c'], // alternative URLs
  }, … ]
```

Optional: Eigener Chunk (dabei kein *import* zu Dateibeginn!)

```
    component: () => import(
   /* webpackChunkName: "HostFuerMeineKomponente" */
   '../views/HostFuerMeineKomponente.vue'),
```

Navigation zwischen Komponenten

Im Template: `<router-link to="/a/b/c">Go!</router-link>`
Im Code: `import router from '@/router';`
`import { useRoute } from 'vue-router';`
Navigation per URL: `router.push("/a/b/c");`
Navigation per Name: `router.push("HostFuerMeineKomponente");`
Aktuelle URL ermitteln: `useRoute().path`

Webadressen

Produkt-Website zu Vue.js 3	v3.vuejs.org
GitHub-Projekte	github.com/vuejs
Vue.js Chat auf Discord	chat.vuejs.org
Vue.js Forum	forum.vuejs.org
Deutsches Vue.js-Buch	www.IT-Visions.de/VueBuch

Spickzettel („Cheat Sheet“): Single Page Web Apps mit Vue.js 3 und Vue CLI oder Vite

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

www.IT-Visions.de
Dr. Holger Schwichtenberg

Variablen im Vue.js-Template (Interpolation)

Ausgabe mit Interpolationssyntax (HTML-Encoded)
```
<div>{{ x }} + {{ y }} =<b>{{ x + y }}</b></div>
```
Alternative Schreibweise mit Direktive (HTML-Encoded)
```
<b v-text="x + y"></b>
```
Ausgabe (nicht HTML-Encoded)
```
let htmlString = "Das &lt;b&gt;-Tag macht <b>fett</b>!";
…
<span v-html="htmlString"></span>
```
Interpolation mit ternärem Operator
```
<div>{{ x > 0 ? x : "ungültiger Wert" }}</div>
```
Escape für {{ }}
```
<div v-pre>Mit {{ x }} geben sie x aus.</div>
```
Interpolation mit Nutzung von JavaScript-Objekten
```
<div>{{ new Intl.NumberFormat().format(x + y) }}</div>
```
Interpolation mit Ausgabe von Objekteigenschaften
```
<div>#{{ obj.ID }} {{ obj.Datum.getFullYear() }}</div>
```
Interpolation mit Funktionsaufruf (await nicht möglich hier!)
```
<div>Die Antwort auf alle Fragen:{{ Berechnung(123) }}</div>
```
Interpolation mit Aufruf einer Objektmethode
```
<div>Objektinfo: #{{ obj.getInfo(false) }}</div>
```
Variablen in HTML-Attributen
```
<a v-bind:href="linkURL">{{ linkText }}</a>
<div v-bind:title="`Summe aus ${x} und ${y}`">{{ x + y }}</div>
```
Kurzform
```
<a :href="linkURL">{{ linkText }}</a>
<div :title="`Summe aus ${x} und ${y}`">{{ x + y }}</div>
```
Bei HTML-Attributen ohne Wert entfernt false den Wert
```
<button :disabled="(x < 0)">x ändern</button>
```
Zwei-Wege-Datenbindung an Wert von Eingabesteuerelementen
```
<input type="number" v-model="x">
```

Bedingte Anzeige mit v-if und v-show

v-if: bei false wird nicht gerendert
```
<div v-if="x > 0">Gewinn {{ x }}</div>
<div v-else-if="x == 0">Kein Gewinn</div>
<div v-else>Verlust {{ x }}</div>
```
v-show: false führt zu style="display: none;"
```
<div v-show="x < 0">Verlust {{ x }}</div>
```

Bedingte Formatierung

Klassenzuweisung per Variable
```
let eineCSSKlasse = "fett"; let zweiCSSKlassen = ["fett", "rot"];
…
<div v-bind:class="eineCSSKlasse">{{ x }}</div>
<div v-bind:class="zweiCSSKlassen">{{ x }}</div>
```
Klassenzuweisung mit Ausdruck
```
<div v-bind:class="{ 'gruen': x % 2 == 0, 'rot': x % 2 != 0 }">{{ x
}}</div>
```
Kurzform
```
<div :class="{ 'gruen': x % 2 == 0, 'rot': x % 2 != 0 }">{{ x }}</div>
```
Bedingter Style: Wert aus Variable
```
<div v-bind:style="{ color: farbe }">{{ x }}</div>
```
Kurzform
```
<div :style="{ color: farbe }">{{ x }}</div>
```
Bedingter Style mit Ausdruck
```
<div :style="{ color: (x % 2 == 0 ? 'green' : 'red') }">{{ x }}</div>
```
Mischung aus statischer und bedingter Formatierung
```
<div class="num"
     :class="{ 'gruen': x % 2 == 0, 'rot': x % 2 != 0 }">{{ x }}</div>
<div style="font-weight:800;"
     :style="{ color: (x % 2 == 0 ? 'green' : 'red') }">{{ x }}</div>
```
Mischung aus statischer und bedingter Formatierung (Alternative)
```
<div :class="'fett ' + (x % 2 == 0 ? 'gruen' : 'rot')">{{ x }}</div>
<div :style="'font-weight: 800;color:' + (x % 2 == 0 ? 'green' :
'red')">{{ x }}</div>
```

DOM-Ereignisse behandeln

Ereignis direkt behandeln
```
<button v-on:click="() => x+=10">+10</button>
```
Kurzform
```
<button @click="() => x+=10">+10</button>
```
Ganze Befehlsfolgen sind möglich (aber nicht übersichtlich)
```
<button @click="() => { x+=10; console.log(x);}">+10</button>
```
Ereignis in eigener Funktion behandeln
```
<button v-on:click="Erhoehe">+1</button>
```
Kurzform immer möglich
```
<button @click="Erhoehe">+1</button>
```

Mögliche Ereignisse: Alle HTML-DOM-Events, siehe wiki.selfhtml.org/wiki/JavaScript/DOM/Event

Bedingte Ereignisbehandlung
```
<button @click="x < 10 && Erhoehe()">+1</button>
```
Eigene Parameter an die Ereignisbehandlung übergeben
```
<button @click="ErhoeheBy(5)">+5</button>
```
DOM-Ereignisparameter verwenden
```
<div style="border-style: dotted;height: 100px; width: 200px;"
@mousemove="MausAktion">{{ pos }}</div>
```
DOM-Ereignisparameter und eigene Parameter verwenden
```
<div @mousemove="(e) => MausAktion(e, 'Mauszeiger')" >
   {{ pos }}</div>
```
Mehrere Ereignisse können zum gleichen Handler führen
```
<div @mousemove="MausAktion"
     @mouseleave="MausAktion"> {{ pos }}
</div>
```
Ereignisbehandlung den Mausereignissen
```
let position = ref("Bewege die Maus hier hin!");
function MausAktion(e: MouseEvent, text: string = "Maus") {
 if (e.type === "mouseleave") position.value = "Maus außerhalb!";
 else position.value = text + ": " + e.x + "/" + e.y;   }
```

Schleifen mit v-for (:key ist Pflichtangabe!)

Schleife über Array mit Zahlen
```
<li v-for="z in arrZahlen" :key="z">{{ z }}</li>
```
Schleife über Array mit Zahlen inkl. laufendem Index (auch als :key)
```
<li v-for="(z, index) in arrZahlen" :key="index">
{{ index + 1 }}. Zahl: {{ z }}</li>
```
Schleife über Array mit Objekten
```
<li v-for="o in arrFirmen" :key="o.ID">{{ o.Name }}</li>
```
Schleife über Map
```
<li v-for="[k, wert] in map" :key="k">{{ k }}, {{ wert }}</li>
```
Schleife über ein einzelnes Objekt
```
<li v-for="(propName, wert) in objFirma" :key="propName">
{{ propName }} = {{ wert }}</li>
```

Modifikatoren für DOM-Ereignisse

@click.stop="Aktion"	Keine Ereignisweitergabe
@submit.prevent="Aktion"	Keine Standardreaktion
@mousemove.once="Aktion"	Nur einmalige Reaktion
@keyup.enter="Aktion"	Reaktion auf ENTER
@keyup.alt.c="Aktion"	Reaktion auf alle mit ALT+C
@keyup.alt.c.exact="Aktion"	Reaktion nur auf ALT+C
@click.ctrl="Aktion"	Reaktion nur STRG + Mausklick

Vorlagenbasierte Komponente (mit Slots) definieren

Expander.vue
```
<template>
 <div class="card">  <div class="card-header">
  <h5 class="card-title" @click="Umschalten">
   <span class="oi"
    :class="Eingeklappt ? 'oi-caret-bottom' : 'oi-caret-top'" ></span>
   <slot name="kopfzeile"></slot>
  </h5>  </div>
 <div v-show="!Eingeklappt" class="card-body">
 <slot></slot> <!-- Haupt-Slot -->
 </div> </div> </template>

<script setup lang="ts">
 import { ref} from 'vue';
 let Eingeklappt = ref(false);
 let Umschalten = () => Eingeklappt.value = !Eingeklappt.value;
</script>
```

Vorlagenbasierte Komponente (mit Slots) nutzen

```
import Expander from './Expander.vue';
…
<Expander>
  <template v-slot:kopfzeile>Überschrift</template>
  <p>Inhalt</p> <!-- Inhalt für Haupt-Slot -->
</Expander>
```

Über den Autor

Dr. Holger Schwichtenberg gehört zu den bekanntesten Experten für die Entwicklung von Web- und .NET-Anwendungen. Er hat zahlreiche Bücher zu diesen Themengebieten (u.a. zu Vue.js, ASP.NET und Blazor) 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**