

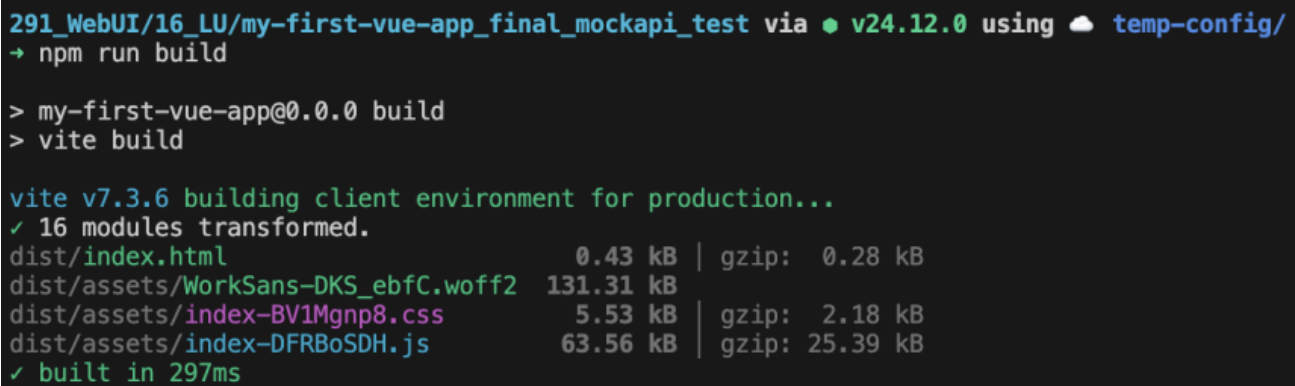
LU16c - Bundling, Minify & Deploy

Bevor eine Vue.js-App veröffentlicht werden kann, muss sie **gebündelt** werden. Dieser Schritt optimiert alle Dateien für den Produktionseinsatz und erstellt einen `dist`-Ordner, der auf einem Server bereitgestellt werden kann.

npm run build

Der Befehl `npm run build` startet den Vite-Build-Prozess.

```
npm run build
```



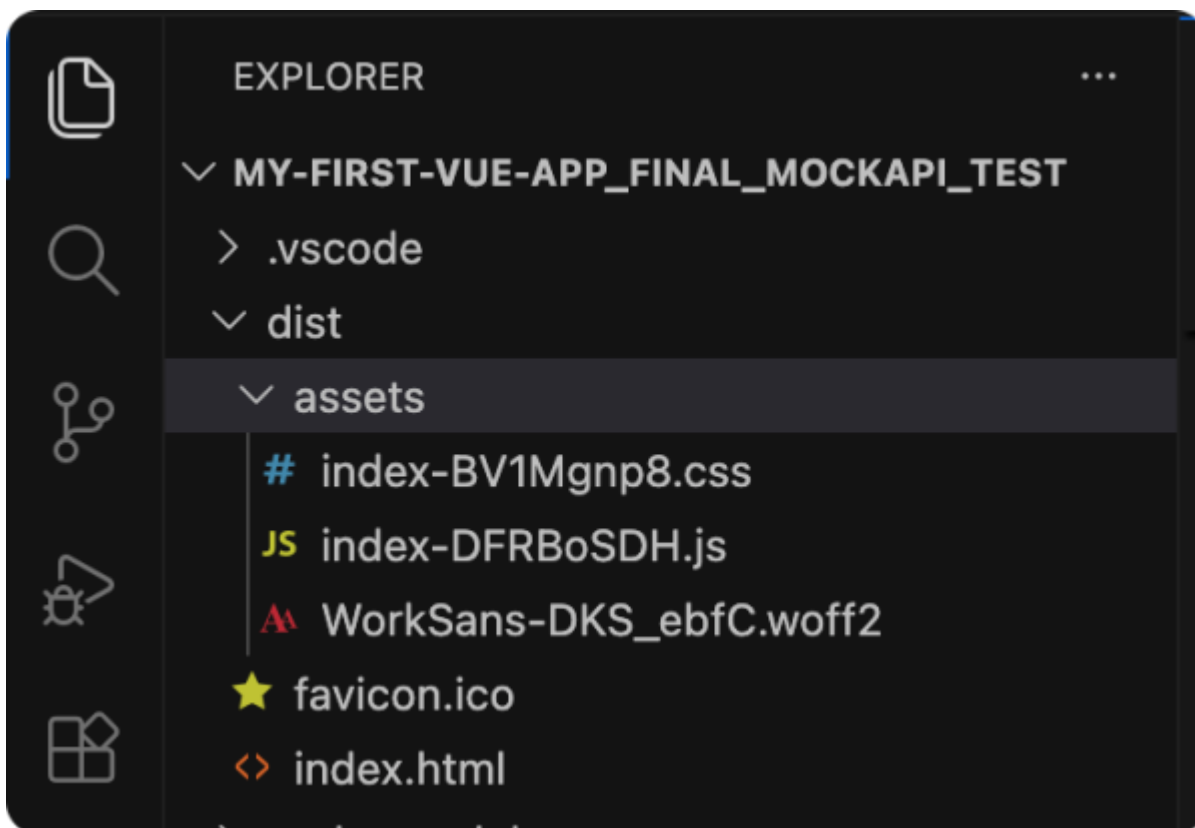
```
291_WebUI/16_LU/my-first-vue-app_final_mockapi_test via ● v24.12.0 using ☁ temp-config/
→ npm run build

> my-first-vue-app@0.0.0 build
> vite build

vite v7.3.6 building client environment for production...
✓ 16 modules transformed.
dist/index.html                0.43 kB | gzip: 0.28 kB
dist/assets/WorkSans-DKS_ebfC.woff2 131.31 kB
dist/assets/index-BV1Mgnp8.css   5.53 kB | gzip: 2.18 kB
dist/assets/index-DFRBoSDH.js   63.56 kB | gzip: 25.39 kB
✓ built in 297ms
```

Platzhalter: Screenshot Terminal mit Build-Output - Dateigrößen, Chunk-Namen, Dauer

Nach dem Build liegt ein neuer Ordner `dist/` im Projekt:



Screenshot VS Code Explorer mit dist-Ordner: index.html, assets/-Unterordner mit .js und .css Dateien

Dieser dist/-Ordner enthält alles, was für den Betrieb der App nötig ist – kein Node.js, kein Vite, keine Vue-Komponenten.

Minification

Minification bedeutet: überflüssige Zeichen aus dem Code entfernen, ohne die Funktion zu verändern.

Vorher (Entwicklung)	Nachher (Production)
Zeilenumbrüche, Einrückungen	Alles in einer Zeile
Lange Variablennamen	Kurze Namen (a, b, fn)
Kommentare	Entfernt

```

src > components > AccordionItem.vue > {} template > @ div.accordion-item > @ button.accordion-
1 <script setup>
2 import { ref } from 'vue';
3 import { truncateText } from '../utils/textHelpers.js';
4
5 const props = defineProps({
6   faq: Object,
7 });
8
9 let isOpen = ref(false);
10
11 function toggleOpen() {
12   isOpen.value = !isOpen.value;
13 }
14 </script>
15
16 <template>
17   <div class="accordion-item">
18     <button
19       class="accordion-btn"
20       v-on:click="toggleOpen"
21       :aria-expanded="isOpen"
22     >
23       {{ isOpen ? faq.question : truncateText(faq.question, 35) }}
24     </button>
25
26     <div class="panel accordion-body" v-bind:class="{ open: isOpen }">
27       {{ faq.answer }}
28     </div>
29   </div>
30 </template>
31
32 <style scoped>
33 .panel {
34   height: 0;
35   opacity: 0;
36   overflow: hidden;
37   transition: all 800ms ease-in-out;
38   font-size: 1rem;
39   font-weight: 400;

```


```

dist > assets > JS index-DFRBoSDH.js > ...Minified Vue-Projekt
1 (function(){const t=document.createElement("link").relList;if(t&&t.supports&&t.supports("modulepreload"))return;for(const r of document.querySelectorAll('link[rel="modulepreload"]'))n(r);new MutationObserver(r=>{for(const i of r)if(i.type==="childList")for(const o of i.addedNodes)o.tagName==="LINK"&&o.rel==="modulepreload"&&n(o)}.observe(document,{childList:!0, subtree:!0});function s(r){const i={};return r.integrity&&(i.integrity=r.integrity),r.referrerPolicy&&(i.referrerPolicy=r.referrerPolicy),r.crossOrigin==="use-credentials"?i.credentials="include":r.crossOrigin==="anonymous"?i.credentials="omit":i.credentials="same-origin",i}function n(r){if(r.ep)return;r.ep=!0;const i=s(r);fetch(r.href,i)}();function Fs(e){const t=Object.create(null);for(const s of e.split(","))t[s]=1;return s=>s in t}const V={},tt=[],Ee={}>{}},Fn=(i=>{!1,Qt=e=>e.charCodeAt(0)==111&&e.charCodeAt(1)==110&&(e.charCodeAt(2)>122||e.charCodeAt(2)<97),Xt=e=>e.startsWith("onUpdate:"),ee=Object.assign,Ds=(e,t)>{const s=e.indexOf(t);s>=1&&s.splice(1,1),Vr=Object.prototype.hasOwnProperty,j=(e,t)>Vr.call(e,t),P=Array.isArray,st=>Pt(e)===[object Map]">Pt(e)===[object Set]">Pt(e)===[object Date]">Ie=>typeof e==="function",Ge=>typeof e==="string",Ae=>typeof e==="symbol",Ne=>e===null&&typeof e==="object",jn=>(N(e)||I(e))&&I(e.then)&&I(e.catch),Hn=Object.prototype.toString,Pt=>Hn.call(e),Wr=>Pt(e).slice(8,-1),Ln=>Pt(e)===[object Object]",js=>G(e)&&e!="NaN"&&e[0]!="-"&&""+parseInt(e,10)==e,_t=Fs("key,ref,ref_for,ref_key,onVnodeBeforeMount,onVnodeMounted,onVnodeBeforeUpdate,onVnodeUpdated,onVnodeBeforeUnmount,onVnodeUnmounted"),Zt=>{const t=Object.create(null);return(s=>t[s]||(t[s]=e(s)))},Br=/-\/w/g,Fe=Zt(e=>e.replace(Br,t=>t.slice(1).toUpperCase())),qr=/\B([A-Z])/g,Ze=Zt(e=>e.replace(qr,"-$1").toLowerCase()),Nn=Zt(e=>e.charCodeAt(0).toUpperCase()+e.slice(1)),fs=Zt(e=>e?'on$Nn(e)':''),Te=(e,t)=>!Object.is(e,t),us=(e,...t)>{for(let s=0;s<e.length;s++)e[s](...t)},$n=(e,t,s,n=!1)>{Object.defineProperty(e,t,{configurable:!0,enumerable:!1,writable:n,value:s})},Gr=>{const t=parseFloat(e);return isNaN(t)?e:t;let rn;const kt=()=>rn||(rn=typeof globalThis<"u"?globalThis:typeof self<"u"?self:typeof window<"u"?window:typeof global<"u"?global:{});function es(e){if(P(e)){const t={};for(let s=0;s<e.length;s++){const n=e[s],r=G(n)?Qr(n):es(n);if(r)for(const i in r)t[i]=r[i]}return t}else if(G(e)||N(e))return e}

```

in der - links normal, rechts minifiziert

Ziel: Kleinere Dateigröße → kürzere Ladezeit für die Nutzerin / den Nutzer.



Vite minifiziert automatisch bei `npm run build`. Sie müssen nichts zusätzlich konfigurieren.

Chunking & Tree Shaking

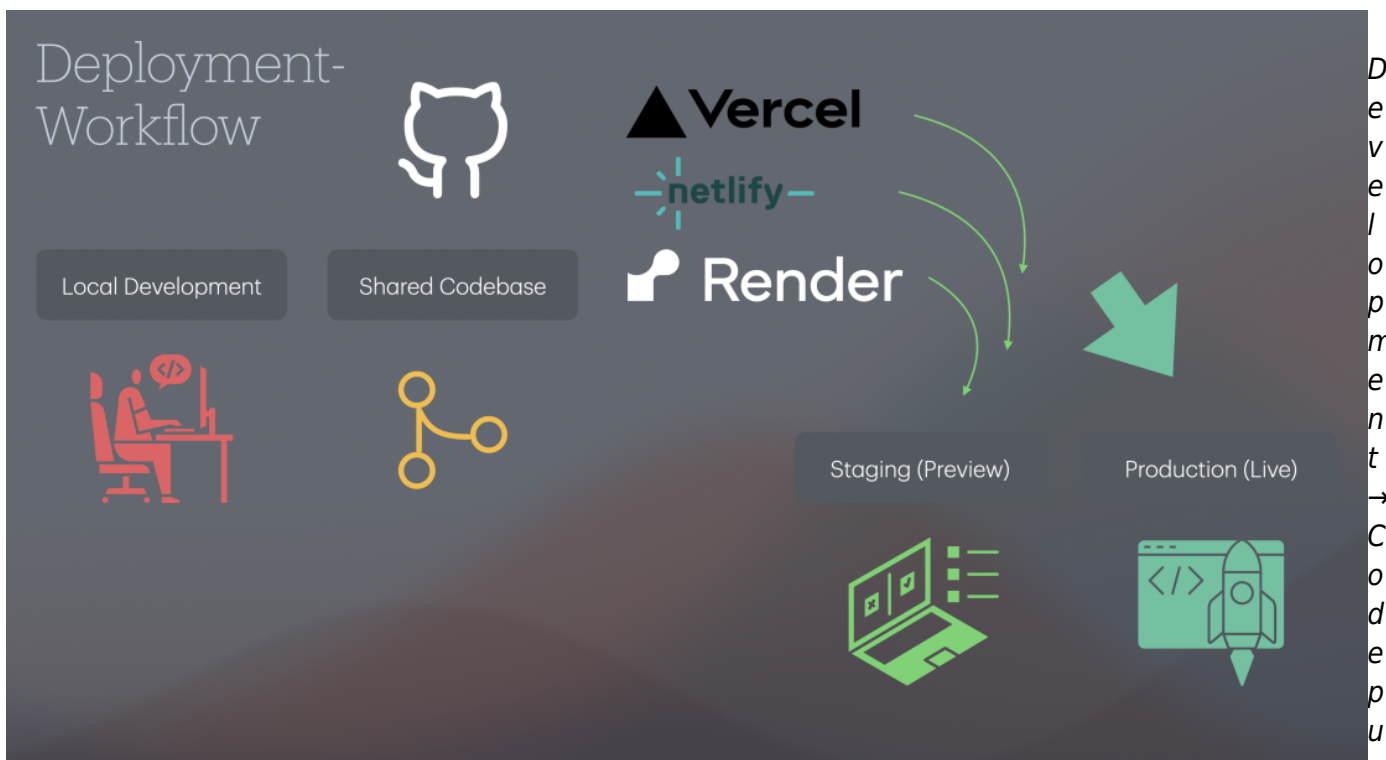
Vite teilt den JavaScript-Code automatisch in **mehrere kleine Dateien (Chunks)** auf, anstatt alles in eine grosse Datei zu schreiben.

Chunking: Der Browser lädt nur den Code, den er gerade braucht - nicht alles auf einmal.

Tree Shaking: Code, der im Projekt importiert aber nie verwendet wird, wird automatisch aus dem Build entfernt.

Deploy: Mit Git (Vercel, Netlify, Render)

Die meisten modernen Hosting-Plattformen verbinden sich mit einem GitHub-Repository und deployen automatisch, sobald neuer Code gepusht wird.



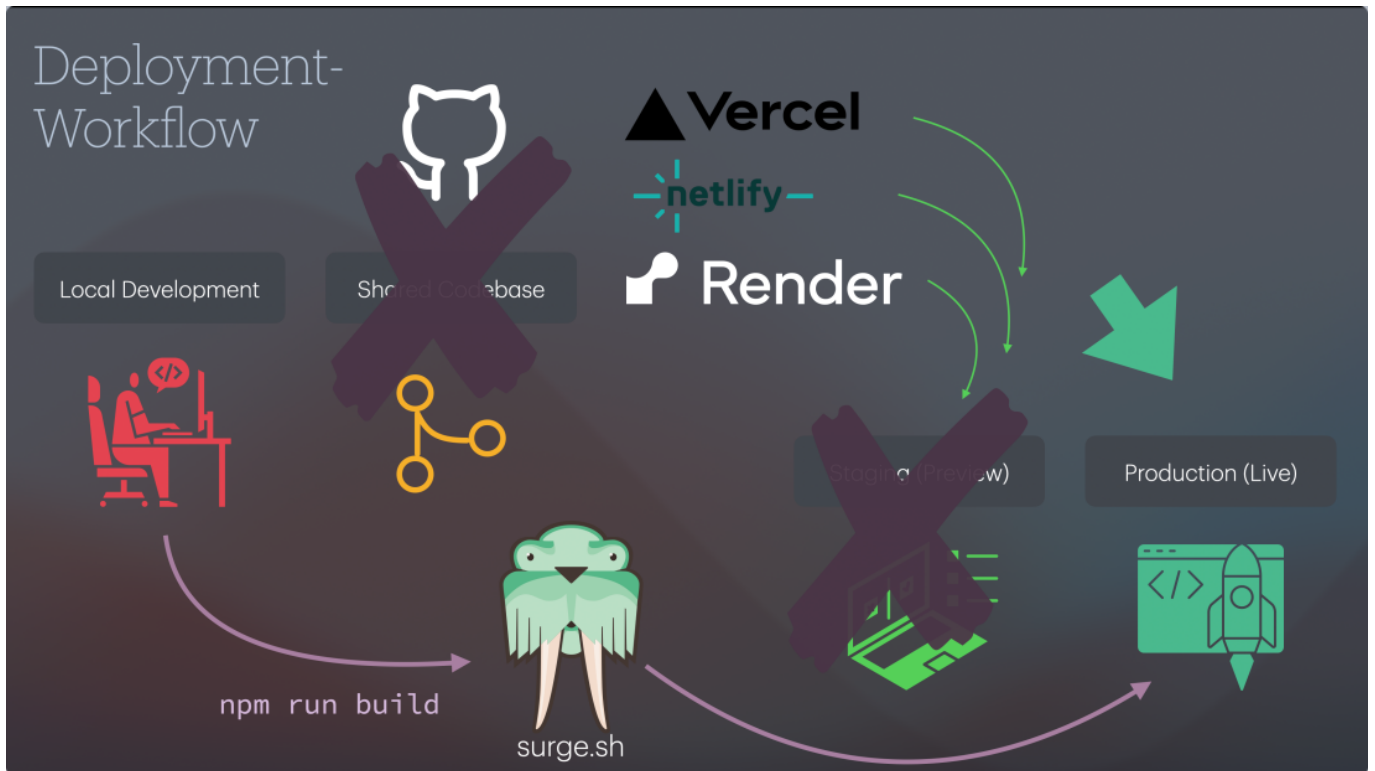
h zu GitHub → Vercel/Netlify/Render → Preview → Production (Live)

Anbieter	Kostenlos	Besonderheit
Vercel	☐ (Hobby-Plan)	Sehr schnell, optimiert für Vue/React/Next.js
Netlify	☐ (Starter-Plan)	Netlify Drop für direktes Hochladen möglich
Render	☐ (mit Einschränkungen)	Auch für Backend-Services geeignet

Voraussetzung: Ein GitHub-Account und ein Repository mit dem Projektcode. Diese Plattformen eignen sich ideal für Teams, die mit Git arbeiten.

Deploy: Ohne Git (Surge.sh & Netlify Drop)

Für den Unterricht - ohne Git-Kenntnisse - gibt es zwei einfachere Wege:



Vue.js mit npm run build und surge.sh deployen

Option A: Surge.sh (Terminal-Weg)

```
npm run build
cd dist
npx surge
```

Surge fragt beim ersten Start nach E-Mail und Passwort (kostenlose Registrierung direkt im Terminal).
Anschliessend wählen Sie eine Domain:

```
domain: vorname-faq.surge.sh
```



Screenshot Terminal – surge-Befehl, Registrierung, gewählte Domain, grüne Erfolgsanzeige

□ Die App ist live unter <https://vorname-faq.surge.sh>

Option B: Netlify Drop (Drag-and-Drop)

1. npm run build ausführen
2. app.netlify.com/drop im Browser öffnen
3. Den dist/-Ordner per Drag-and-Drop ins Browserfenster ziehen

4. Fertig - Netlify erstellt sofort eine öffentliche URL

Hinweis: Ohne Konto ist die Netlify-Drop-URL nur ca. 1 Stunde aktiv. Mit kostenlosem Konto (keine Kreditkarte nötig) bleibt sie dauerhaft erreichbar.

Vergleich: Deploy-Optionen

	Surge.sh	Netlify Drop	Vercel / Netlify / Render
Git nötig?	Nein	Nein	Ja
Terminal nötig?	Ja	Nein	Nein
Aufwand	Gering	Sehr gering	Mittel (Einrichtung)
Geeignet für	Unterricht, Demos	Unterricht, schnelle Tests	Echte Projekte
Automatisches Re-Deploy	Nein	Nein	Ja (bei Git-Push)
Gratis	☐	☐	☐ (mit Limits)
Eigene Domain	Ja	Nein	Ja

Zusammenfassung: Der komplette Ablauf

```
# 1. App für Production optimieren
npm run build

# 2. In den dist-Ordner wechseln
cd dist

# 3. Deployen mit Surge
npx surge
```

Nach diesen drei Befehlen ist die Vue.js-App unter einer echten, öffentlichen URL erreichbar.

From:
<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/de/modul/m291/learningunits/lu16/theorie/c_deploy

Last update: **2026/06/28 23:55**

