

LU10a - Statische Webseiten veröffentlichen (Nachtrag)

Definitionen / Unterscheidung

SSR vs. Entkoppelte Frontend/Backend-Architektur

In modernen Webprojekten wird häufig eine klare Trennung von Frontend und Backend umgesetzt:

Frontend: z. B. mit Angular, React oder Vanilla JS Backend: REST- oder GraphQL-API (z. B. in Node.js, Java, Python)

Diese Architektur bietet klare Vorteile:

- Lose Kopplung (Frontend und Backend unabhängig entwickelbar)
- Austauschbarkeit (z. B. Frontend-Technologie wechseln)
- Einfache Deployments (statische Dateien + API)
- Gute Skalierbarkeit

Static vs dynamic hosting

Ob etwas statisch deploybar ist, hängt nicht von den Features ab, sondern davon, ob der Server zur Laufzeit am Rendering beteiligt ist.

Es ist also eine Architekturfrage, ob man Server-Side-Rendering (SSR) einsetzt oder nicht.

Salopp gesagt, können die meisten Webapplikationen in ein statisches Frontend und ein (allfälliges) dynamisches Backend aufgeteilt werden.

Statische Webseiten sind möglich bei

- Vanilla / Plain Html und JS
- Angular
- React
- Vite
- Astro (default)

Ausnahmen sind

- PHP
 - .php
- Next.js mit API/SSR
 - `getServerSideProps`
 - Routing (`route.js`, `route.ts`, „`/api/*`“)
 - `NextResponse`
 - `cookies()`
 - `headers()`

- Express mit Templates
 - .ejs
 - .pug
 - .hbs

Next.js

next.config.js

```
const nextConfig = {  
  output: 'export',  
};  
  
module.exports = nextConfig;
```

Nach dem erfolgreichen Ausführen von folgendem Befehl, sind die fertigen Dateien unter /out verfügbar.

```
npm run build
```

```
npx serve out
```

Mögliche Fehler:

- „Route Handlers are not supported with output: export“
- „Dynamic server usage“

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

Permanent link:
<https://wiki.bzz.ch/de/modul/ffit/3-jahr/cicd/learningunits/lu10/a?rev=1777981469>

Last update: **2026/05/05 13:44**

