

VUE.JS & NUXT

16. April 2025

Milena Kübler & Simon Wimmer & Max Tyrchan
Edv: 143404a Entwicklung von Web-Anwendungen

ZIEL & GLIEDERUNG



Unser Ziel ist es euch in den nächsten beiden Vorlesungen die Grundlagen von Vue.js und Nuxt zu vermitteln.

- 01** Einführung
- 02** Theoretische Grundlagen & Übung
- 03** Einführung in Nuxt & Übung
- 04** Routing und Seitenverwaltung in Nuxt & Übung
- 05** Abschluss und Ausblick

WARUM FRAMEWORKS?

Strukturierung und Wartbarkeit

- Frameworks helfen, Code zu modularisieren

Performance und Ladezeiten

- Frameworks nutzen virtuelle DOMs und reaktive Updates

State Management

- Moderne Apps haben viele Zustände
- Frameworks bieten zentrale Lösungen (z.B. Pinia, Vuex)

Search Engine Optimization und Server-Side Rendering

- SPAs haben ein Problem mit SEO, da sie erst nach dem Laden der Seite JavaScript ausführen

 Server-Side Rendering mit Nuxt.js

WARUM VUE.JS?



💡 Einfacher Einstieg

- Sanfte Lernkurve und intuitive Syntax

⚡ Progressiv

- Kann als einfaches Script oder als volles Framework genutzt werden

💡 Leistungsstark und flexibel

💻 Aktive Community und starke Unterstützung

- Vor allem aus chinesischem Raum

GRUNDLEGENDE KONZEPTE

❖ Komponenten-basierte Architektur

- Basiert auf modularer Architektur – UI wird in wiederverwendbare Komponenten aufgeteilt
- Komponente ist in sich geschlossene UI-Einheit mit eigenem HTML, CSS und JavaScript
- Komponenten können miteinander kommunizieren und wiederverwendet werden

☀ Nutzen:

- ✓ Wiederverwendbarkeit – einmal schreiben, überall nutzen
- ✓ Bessere Wartbarkeit – Code bleibt übersichtlich
- ✓ Einfache Zusammenarbeit – verschiedene Teams können an einzelnen Komponenten arbeiten

BEISPIEL KOMPONENTEN

```
src > components > ▼ Reactivity.vue > ...
1  <script setup>
2  import { ref } from 'vue'
3
4  defineProps({
5    msg: String,
6  })
7
8  const count = ref(0)
9  </script>
10
11 <template>
12   <h1>{{ msg }}</h1>
13
14   <div class="card">
15     <button type="button" v-on:click="count++">Klick mich {{ count }}</button>
16   </div>
17 </template>
18
19 <style scoped>
20 .read-the-docs {
21   color: #888;
22 }
23 </style>
24
```

GRUNDLEGENDE KONZEPTE

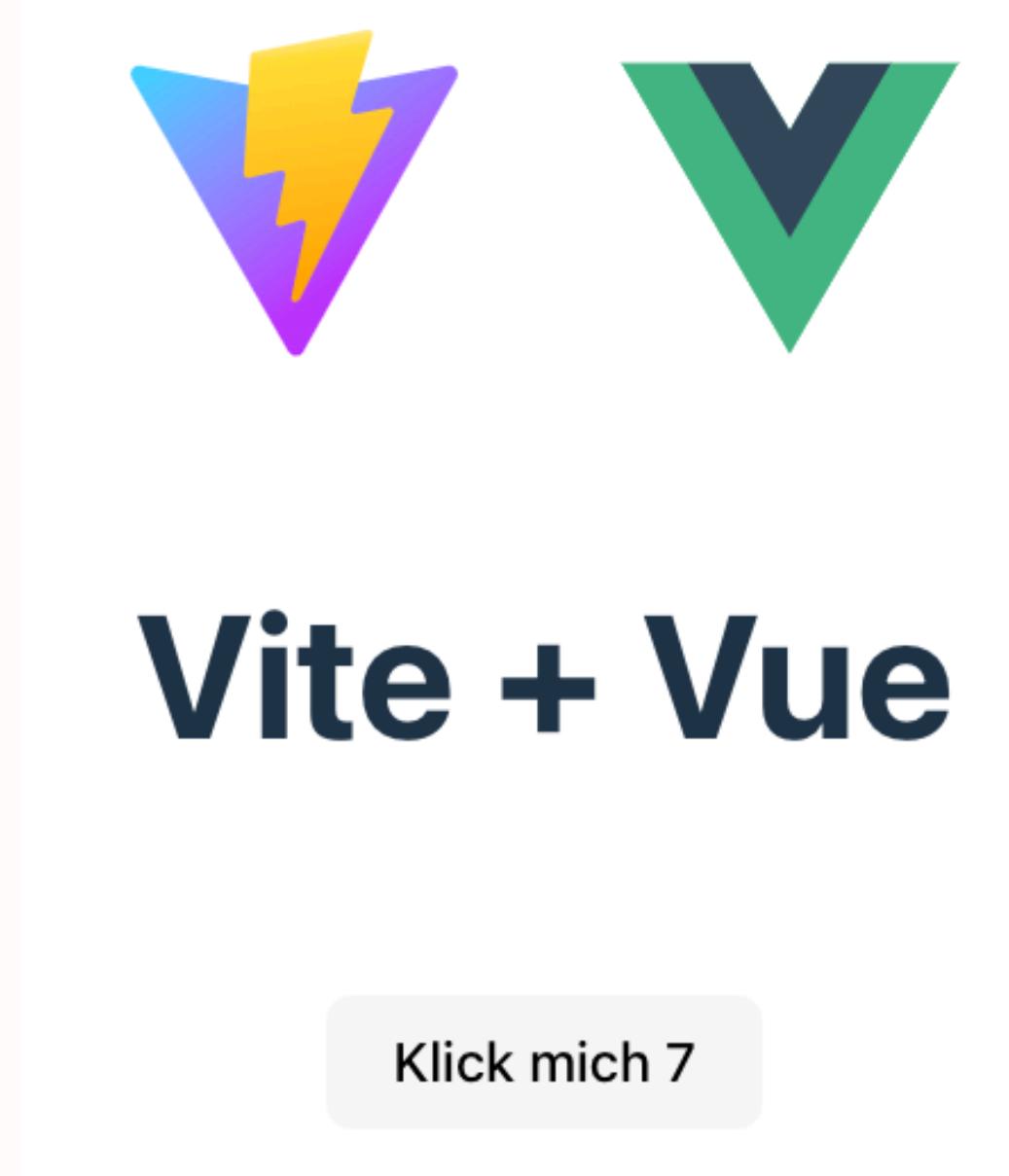
💡 Reaktivität

- Verwendung eines Reactivity-Systems - Änderung an Daten werden erkannt und UI synchron gehalten
- Reaktiver Proxy beobachtet Objekte und triggert Aktualisierungen automatisch

🌟 Nutzen:

- ✓ Keine manuellen DOM-Updates notwendig - hält Code sauber und wartbar

BEISPIEL REAKTIVITÄT



- Nutzung reaktiver Variable
- Bei v-on:click wird automatisch hochgezählt
- Wird von Vue.js erkannt und Zahl im Button automatisch aktualisiert

GRUNDLEGENDE KONZEPTE

🔨 Direktiven - Dynamisches HTML

- Direktiven sind spezielle Attribute, die Vue-Funktionalität direkt in HTML ermöglichen
- Zum Beispiel: v-if, v-for, v-bind, v-on

🌟 Nutzen:

- ✓ Direktiven reduzieren Bedarf an JavaScript-Manipulationen im DOM
- ✓ Klare Trennung zwischen HTML-Struktur und JavaScript-Logik
- ✓ Wiederverwendbarkeit durch benutzerdefinierte Direktiven

BEISPIEL DIREKTIVEN

v-bind



Zur Website

v-if



Toggle

🎭 Ich bin sichtbar!

v-for



- 🍎 Apfel
- 🍌 Banane
- 🍇 Traube

GRUNDLEGENDE KONZEPTE

🔨 Zwei-Wege-Datenbindung

- Änderungen an einer Eingabe werden automatisch im Modell gespeichert – und umgekehrt

🌟 Nutzen:

- ✓ Es wird Code gespart
- ✓ Fehler durch manuelle DOM-Manipulation werden vermieden

GRUNDLEGENDE KONZEPTE

Zwei-Wege-Datenbindung



Hallo ich zeige die Zwei

Hallo ich zeige die Zwei Wege Datenbindung

DAS VUE-ÖKOSYSTEM

TOOL	FUNKTION
Vue Router	Ermöglicht Navigation in Single Page Applications
Vuex  / Pinia 	State Management (States zentral verwalten)
Nuxt.js 	Erweiterung für SEO und Server-Side Rendering
Vue DevTools	Debugging-Tools für Vue-Anwendungen
Vite 	Moderner Build-Tool für Vue



⚡ Modernes Frontend-Build-Tool, dass Entwicklungs- und Build-Erfahrung erheblich verbessert

💡 Funktionen:

- Schneller Entwicklungsserver: Vite lädt nur Teile der Anwendung, die sich ändern und nicht gesamte Seite
- Optimierter Build-Prozess: Durch Verwendung von Entwicklungsserver-Modulen und moderner Javascript-API kann schnell gebaut werden
- Einfache Konfiguration: Es wird wenig Setup benötigt
- Vielseitig: Unterstützt viele verschiedene Frontend-Technologien und ist flexibel anpassbar

VUE ÜBUNG - ERSTE SCHRITTE MIT VUE.JS

Aufgabe:

- Einrichtung einer einfachen Vue.js-Anwendung mit Vite
 - Node.js installieren
 - **npm create vite@latest erstes-vue-projekt --template vue**
 - Vue als Framework auswählen und JavaScript
 - **npm install**
 - **npm run dev**
- Erstellung erster Komponenten
- Einrichten einer Namensliste mit einem Button zum Hinzufügen der Namen

Zusatzaufgabe:

- Füge eine Option hinzu, bereits eingegebene Namen wieder zu löschen

Es geht weiter mit der Besprechung um 10:50Uhr.

VUE ÜBUNG - ERSTE SCHRITTE MIT VUE.JS

- Erstellung der Komponente
- Einrichten einer Namensliste mit einem Button zum Hinzufügen der Namen

```
src > components > ▼ Liste.vue > {} script setup
  You, vor 28 Minuten | 1 author (You)
1  <script setup>    You, vor 28 Minuten • Vue Übung hinzugefügt ...
2  import { ref } from 'vue'
3
4  const newName = ref('')
5  const names = ref([])
6
7  function addName() {
8    if (newName.value.trim()) {
9      names.value.push(newName.value.trim())
10   newName.value = ''
11 }
12 }
13 </script>
14
15 <template>
16   <input v-model="newName" placeholder="Gib deinen Namen ein" />
17   <button v-on:click="addName" :disabled="!newName">Hinzufügen</button>
18
19   <ul>
20     <li v-for="(name, index) in names" :key="index">
21       {{ name }}
22     </li>
23   </ul>
24 </template>
```

Import der Komponente

```
src > ▼ App.vue > {} template
  1  <script setup>
  2  import Liste from './components/Liste.vue';
  3  </script>
```

Zusatzaufgabe:

```
function removeName(index) {
  names.value.splice(index, 1)
}
```

```
<button v-on:click="removeName(index)">☒</button>
```

WAS IST NUXT.JS?



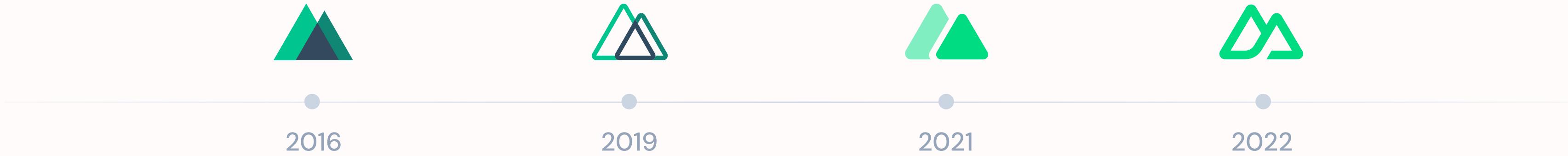
- brick Framework basierend auf Vue.js
- target Ziel: strukturierte, performante und SEO-freundliche Webanwendungen
- refresh Unterstützt SSR, SSG und SPA – je nach Projektanforderung

Wofür wird NuxtJS eingesetzt?

- document Blog, Portfolios, Dokumentationen (SSG)
- trendline SEO-optimierte Produktseiten (SSR)
- lightning-bolt Schnelle, dynamische Anwendungen (SPA)
- globe Universelle Apps (Hybrid-Ansätze möglich)

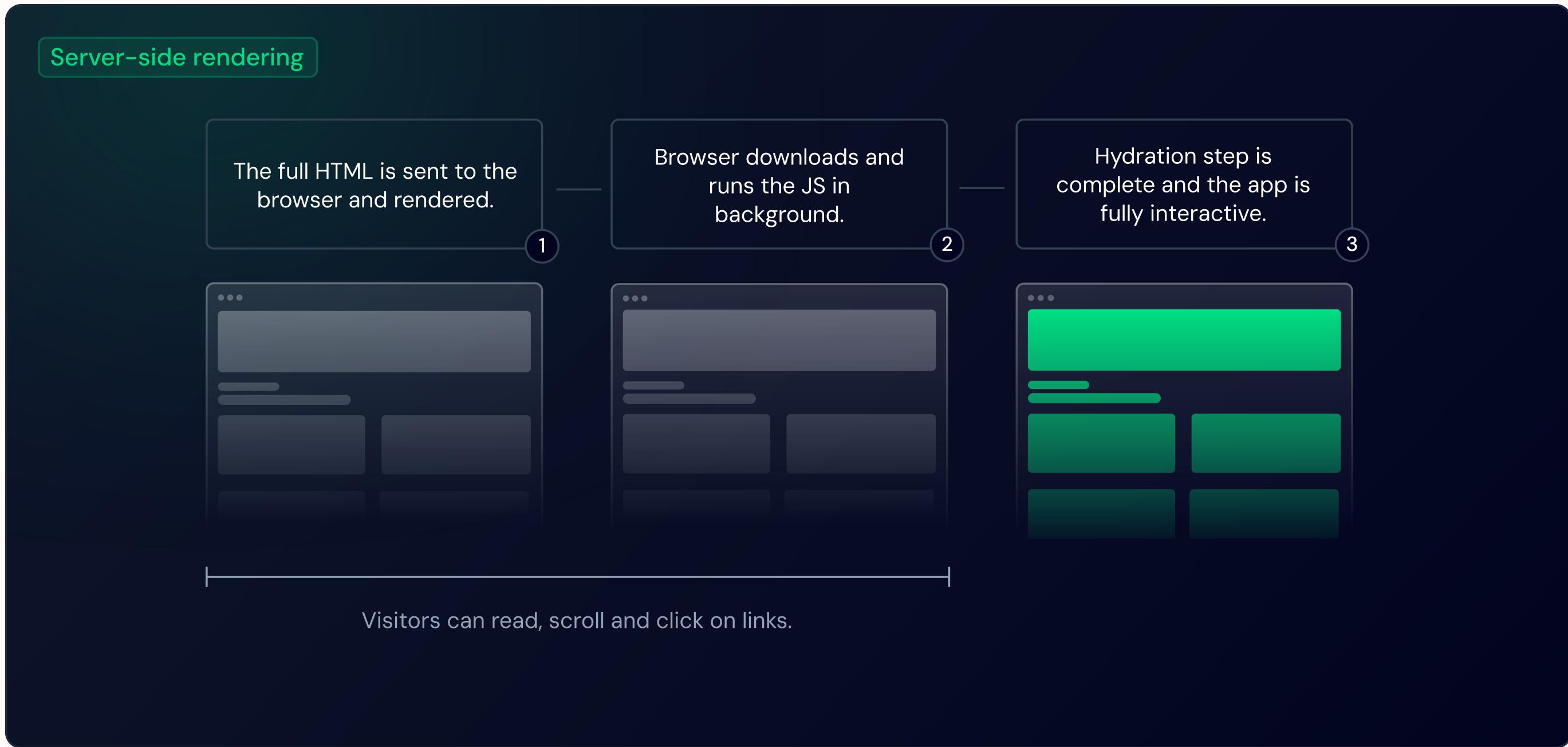
KURZER HISTORISCHER ÜBERBLICK

- Entwickelt von Alexandre und Sébastien Chopin
- Erste Version: 2016
- Nuxt 2 (2018): Einführung von Modulen
- Nuxt 3 (2022): Vue 3, Vite, Nitro Engine → bessere Performance & Flexibilität



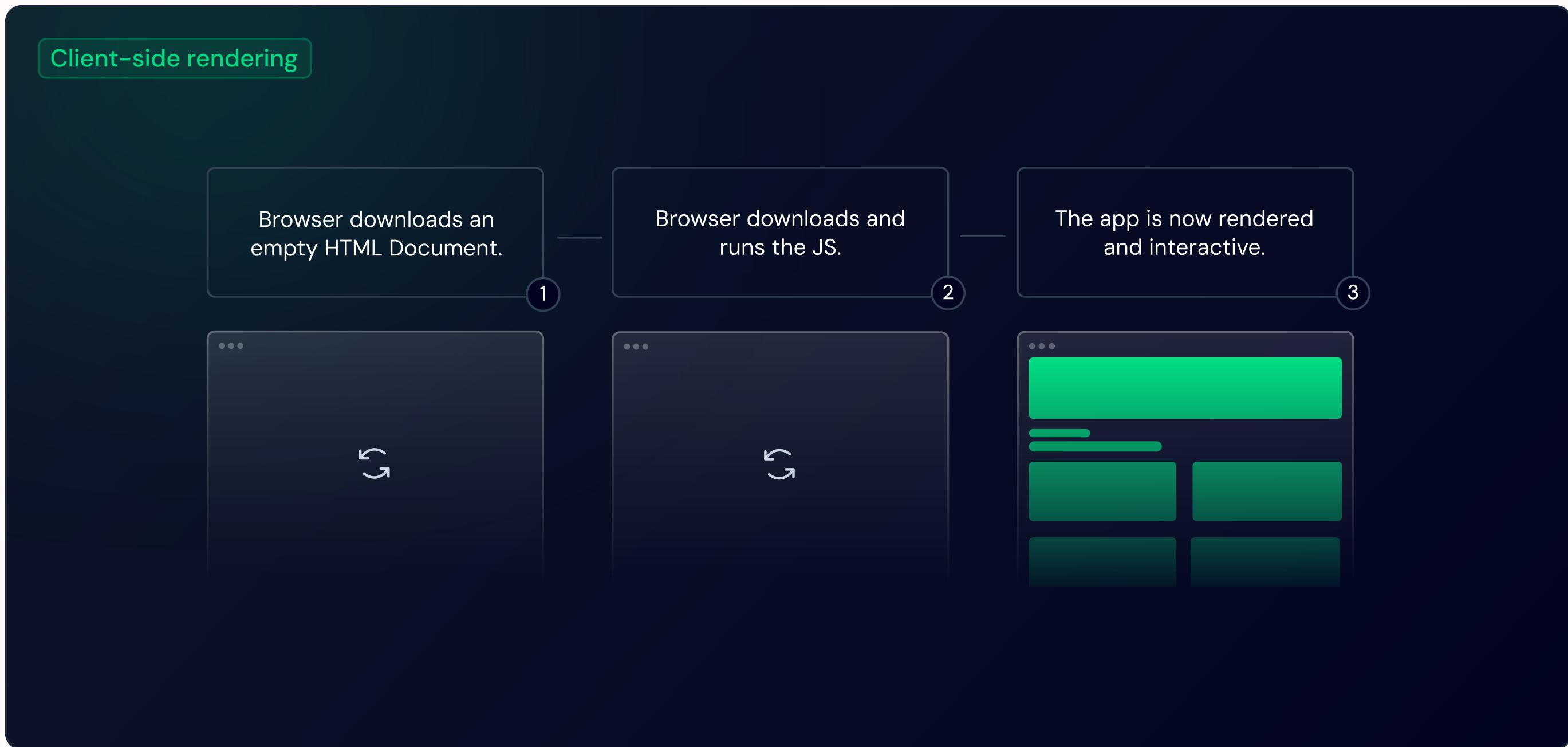
RENDERING - MODI

Server-Side Rendering (SSR)



RENDERING-MODI

Client-Side Rendering (CSR)



SSR VS. CSR – WAS MACHT WANN SINN?

Aspekt	SSR (Server)	CSR (Client)
SEO	Gut für SEO	Schwächer
Ladezeit	Schneller First-Paint	Langsamer Initial Load
Interaktivität	Verzögert (Hydration)	Sofort verfügbar
API-Calls	Server → schneller, sicherer	Browser → langsamer

Weitere Rendering-Modi:

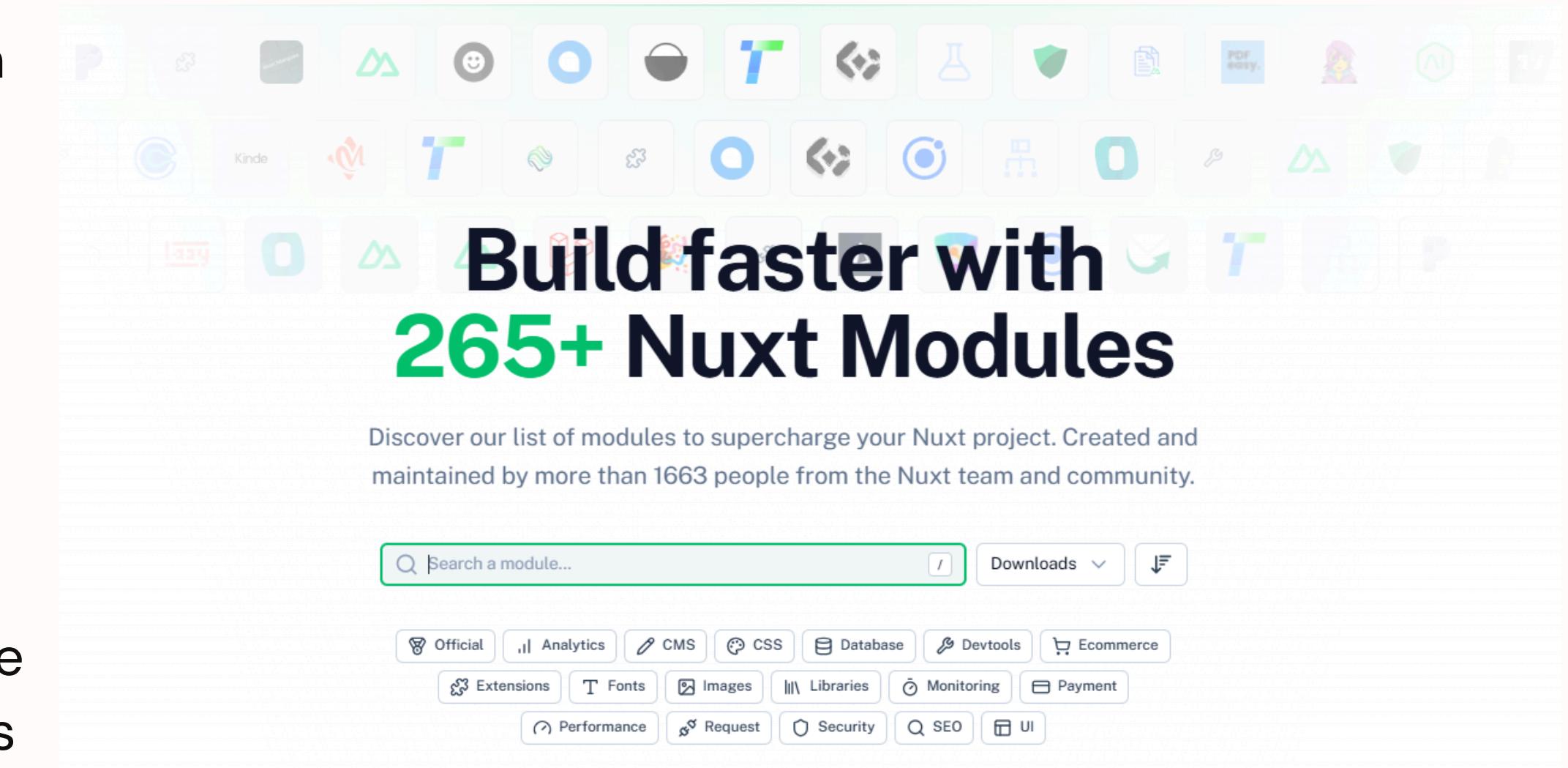
- Hybrides Rendering
- Edge-Side Rendering

MODULARE ARCHITEKTUR

- Große Auswahl an offiziellen & Community-Modulen
- Beispiele: @nuxt/content, @nuxt/auth, @nuxt/image

✓ Vorteile:

- Schnelle Erweiterbarkeit
- Weniger Boilerplate-Code
- Integrierte Best Practices



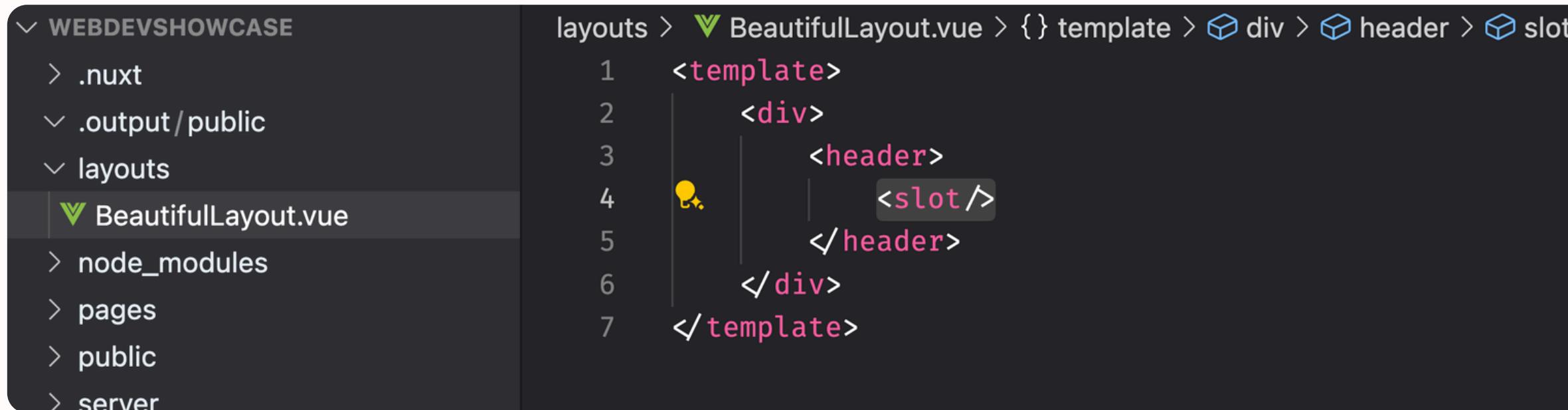
VERZEICHNISTRUKTUR UND KONVENTIONEN

```
└─ POKEDEX
    ├ .nuxt
    ├ assets
    ├ components
    ├ layouts
    ├ node_modules
    ├ pages
    ├ public
    ├ server
    ├ stores
    └ .gitignore
    └─ app.vue
        TS nuxt.config.ts
        {} package-lock.json
        {} package.json
        ⓘ README.md
        TS tsconfig.json
```

- pages/ → Automatisches Routing
- components/ → Wiederverwendbare UI-Bausteine
- layouts/ → Wiederverwendbare Layout-Komponenten
- public/ → Statische Elemente der Webseite (z. B. Bilder)
- assets/ → Assets wie Fonts, Stylesheets, Bilder
- **KONVENTION VOR KONFIGURATION → SPART ZEIT & VERMEIDET FEHLER**

LAYOUTS

- Layouts sind übergeordnete Komponenten, die alle Seiten umschließen
- Geeignet für mehrmals vorkommende Elemente wie Header / Footer
- Verschiedene Layouts lassen sich auch dynamisch mittels `definePageMeta()` definieren



The image shows a file explorer on the left and a code editor on the right. The file explorer displays a project structure with a highlighted file named 'BeautifulLayout.vue' under the 'layouts' directory. The code editor shows the template structure of this file:

```
layouts > BeautifulLayout.vue > {} template > div > header > slot
1  <template>
2    <div>
3      <header>
4        <slot/>
5      </header>
6    </div>
7  </template>
```

A yellow lightbulb icon is positioned over the opening tag of the 'slot' element.

VORTEILE GEGENÜBER REINEM VUE.JS

VORTEIL	GRUND
Strukturiertes Projekt-Setup	Nuxt gibt eine klare Verzeichnisstruktur und Konventionen vor, was Ordnung und Verständlichkeit fördert.
Vereinfachte Konfiguration & Automatisierung	Viele Funktionen wie Routing, SSR und Meta-Tags sind „out-of-the-box“ verfügbar und müssen nicht manuell eingerichtet werden.
Integriertes SSR/SSG	Serverseitiges Rendering und statische Generierung sind direkt integriert – für bessere Performance und SEO.
Optimiertes Entwicklererlebnis (DX)	Schnelles Setup, hilfreiche CLI-Tools, Hot-Reloading und zahlreiche Module machen die Arbeit angenehmer und effizienter.

NUXT ÜBUNG 1 - ERSTE SCHRITTE MIT NUXT.JS

Aufgabe:

- Node.js installieren
- Initialisierung eines neuen Nuxt-Projekts
- Tailwind installieren & Vorlagen-CSS-Sheet einfügen
- Einrichten des Projekts in GitHub
- Erstellen der Index-Seite
 - Überschrift
 - Suchfeld mit Löschbutton
- Erstellen des Layouts für den Header & Footer

Für die Schnellen:

- Anpassung des Layouts und CSS-Stylings
- Testen der Nuxt DevTools

NUXT ÜBUNG 1

```
[Zaro@MacBook-Pro-2 Entwicklung von Webanwendungen % npx nuxi init pokedex
(node:65890) ExperimentalWarning: CommonJS module /opt/homebrew/lib/node_modules
/npm/node_modules/debug/src/node.js is loading ES Module /opt/homebrew/lib/node_
modules/npm/node_modules/supports-color/index.js using require().
Support for loading ES Module in require() is an experimental feature and might
change at any time
(Use `node --trace-warnings ...` to show where the warning was created)

.d$b.
.i$$A$$L .d$b
.$$F' '$$. $$A$$.
j$$' '4$$:' '$$.
j$$' .4$: '$$.
j$$' .$$: '4$.
:$:____.d$$: ____.:$$:
'4$$$$$$$P' .i$$$$$$$P'

Welcome to Nuxt!
Creating a new project in pokedex.                                nuxi 15:58:55
nuxi 15:58:55

✓ Which package manager would you like to use?
npm
• Installing dependencies...                                     nuxi 15:58:56
(node:65917) ExperimentalWarning: CommonJS module /opt/homebrew/lib/node_modules
/npm/node_modules/debug/src/node.js is loading ES Module /opt/homebrew/lib/node_
modules/npm/node_modules/supports-color/index.js using require().
Support for loading ES Module in require() is an experimental feature and might
change at any time
(Use `node --trace-warnings ...` to show where the warning was created)

> postinstall
> nuxt prepare

[15:59:01] ERROR (node:66063) ExperimentalWarning: CommonJS module /Users/Zaro
/Documents/Studium/Master/3. Semester/Entwicklung von Webanwendungen/pokedex/nod
e_modules/debug/src/node.js is loading ES Module /Users/Zaro/Documents/Studium/M
aster/3. Semester/Entwicklung von Webanwendungen/pokedex/node_modules/supports-c
olor/index.js using require().
Support for loading ES Module in require() is an experimental feature and might
change at any time
(Use node --trace-warnings ... to show where the warning was created)

✓ Types generated in .nuxt                                         nuxi 15:59:01

added 580 packages, and audited 582 packages in 5s

125 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
✓ Installation completed.                                            nuxi 15:59:01

✓ Initialize git repository?
Yes
• Initializing git repository...                                     nuxi 15:59:05

Initialized empty Git repository in /Users/Zaro/Documents/Studium/Master/3. Seme
ster/Entwicklung von Webanwendungen/pokedex/.git/

✓ Would you like to install any of the official modules?
none

• Nuxt project has been created with the v3 template. Next steps:
  > cd pokedex                                                 nuxi 15:59:06
  > Start development server with npm run dev                nuxi 15:59:06
  > Start development server with nuxt dev                  nuxi 15:59:06
Zaro@MacBook-Pro-2 Entwicklung von Webanwendungen % ]
```

Empfohlen für Nuxt 3:

- *npx nuxi init <projektname>*

Mit beliebigem Paketmanager und Nuxt CLI:

- *npm/yarn/pnpm create nuxt <projektname>*
- Nutzung von Templates, z.B.:
 - *npm create nuxt@latest -- -t ui*
 - <https://nuxt.com/templates>

Veraltet (für Nuxt 2):

- *npx create-nuxt-app <projektname>*

NUXT ÜBUNG 1

<https://github.com/MaxTyrchan/WebEntPokemon>

NUXT ÜBUNG 1 - BASIC-SEITE

- **npx nuxi module add tailwindcss**
- **main.css** aus dem Repo kopieren
- **nuxt.config.ts** anpassen
 - **tailwindcss: { cssPath: '~/assets/css/main.css' }**

```
ts nuxt.config.ts > ...
1 // https://nuxt.com/docs/api/configuration/nuxt-config
2 export default defineNuxtConfig({
3   compatibilityDate: '2024-11-01',
4   devtools: { enabled: true },
5   modules: ['@nuxtjs/tailwindcss'],
6   tailwindcss: { cssPath: '~/assets/css/main.css' },
7 })
8
```

- **app.vue anpassen**

```
▼ app.vue > ...
1 <template>
2   <div class="main-page">
3     | <NuxtPage />
4   </div>
5 </template>
6
```

- **index.vue in pages/ erstellen**
 - Titel und Suchbox anlegen

```
pages > ▼ index.vue > ...
1  <template>
2    <div class="container">
3      <div class="search-container">
4        <div class="search-wrapper">
5          <input type="text" placeholder="Search Pokémon..." class="search-input" />
6          <button class="clear-button" aria-label="Clear search">x</button>
7        </div>
8      </div>
9    </div>
10   </template>
11
12  <script setup></script>
13
```

- **npm run dev**

NUXT ÜBUNG 1 - LAYOUT

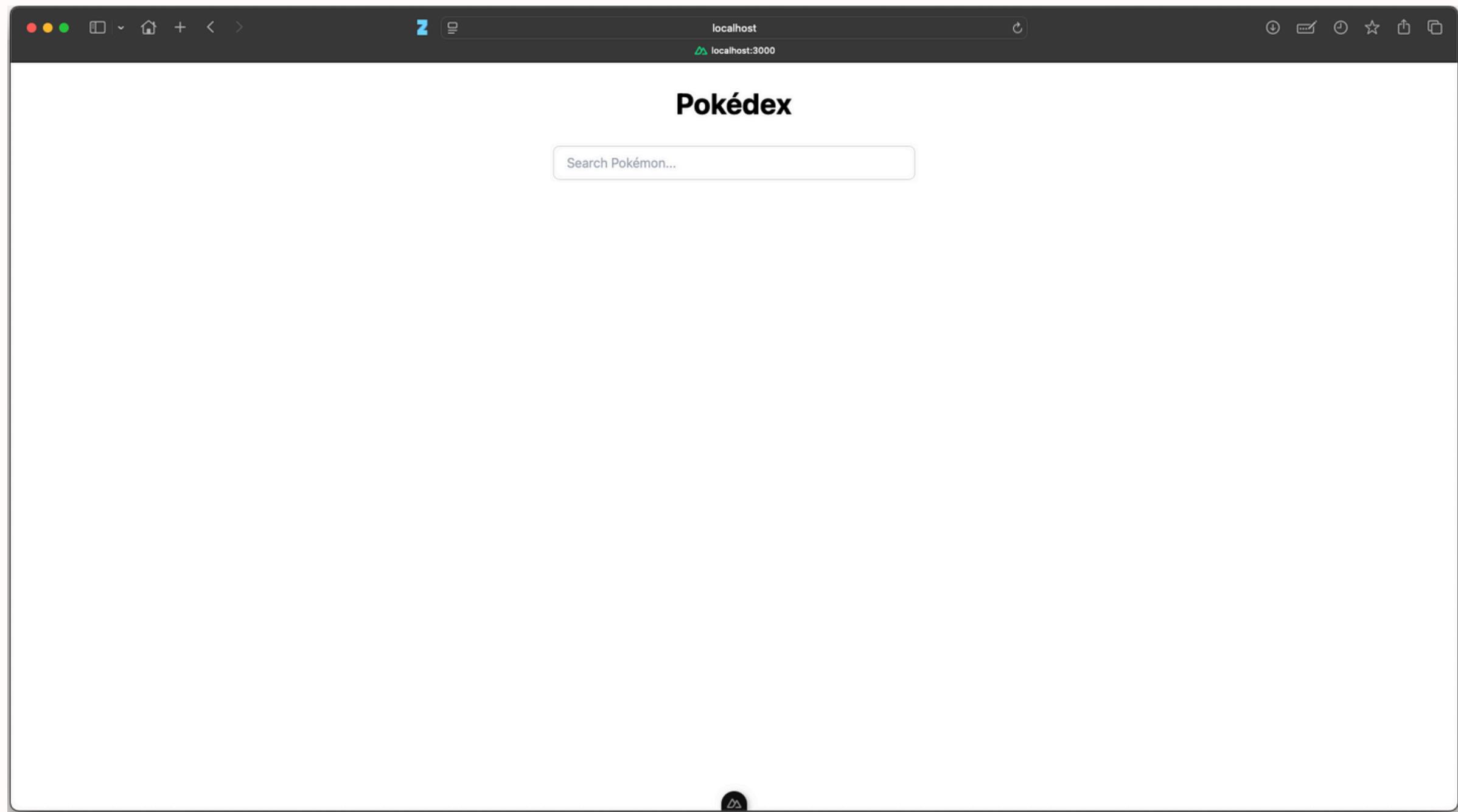
- **default.vue** in *layouts/* erstellen

```
layouts > ▼ default.vue > ...
1  <template>
2    <div class="default">
3      <header class="header">
4        <div class="container">
5          <NuxtLink to="/">
6            <h1 class="page-title">Pokédex</h1>
7          </NuxtLink>
8        </div>
9      </header>
10
11     <main class="main">
12       <slot />
13     </main>
14
15     <footer class="footer">
16       <div class="container">
17         <p>&copy; Pokédex App – Entwicklung von Web-Anwendungen SS 2025</p>
18       </div>
19     </footer>
20   </div>
21 </template>
22
23 <script setup lang="ts">
24 // You can add layout-specific logic here
25 </script>
26
27 <style scoped>
28 /* Add any scoped styles for your layout here */
29 </style>
30
```

- **app.vue** anpassen

```
▼ app.vue > ...
1  <template>
2    <NuxtLayout>
3      <NuxtPage />
4    </NuxtLayout>
5  </template>
6
```

NUXT ÜBUNG 1 - RESULT



Es geht weiter mit der Besprechung um 11:50Uhr.

03. ROUTING & SEITENVERWALTUNG

-  Routing Konfiguration
-  Dynamisches Routing
-  Verschachtelte Routen
-  Typische Fehler

ROUTE KONFIGURATION

VUE.JS

```
import { createMemoryHistory, createRouter } from 'vue-router'

import HomeView from './HomeView.vue'
import AboutView from './AboutView.vue'

const routes = [
  { path: '/', component: HomeView },
  { path: '/about', component: AboutView },
]

const router = createRouter({
  history: createMemoryHistory(),
  routes,
})
```

NUXT

```
> .nuxt
> assets
> node_modules
> pages
> .gitignore
V app.vue
TS nuxt.config.ts
npm package-lock.json
npm package.json
M README.md
~ tailwind.config.js
{...} tsconfig.json
```



NAVIGATION IN NUXT

<NUXTLINK>

- Verknüpft Seiten untereinander
- Verwendet clientseitige Navigation des Vue Routers
- Lädt Komponenten vor



```
<NuxtLink to="/about">Über uns</NuxtLink>
```

useRouter()

- Wird im <script setup> Block einer Vue-Component eingesetzt
- router.push() erwartet einen String oder Routenobjekt
- Eignet sich für Navigation per Button oder Logik



```
const router = useRouter()  
router.push('/kontakt')
```



DYNAMISCHES ROUTING

- Alles was in eckige Klammern gesetzt wird, wird in ein dynamischen Routenparamenter umgewandelt
- Gilt sowohl für Dateien als auch Ordner



```
<script setup>
const route = useRoute()
</script>

<template>
  <h1>Blogpost: {{ route.params.slug }}</h1>
</template>
```



VERSCHACHTELTE ROUTEN

Mit verschachtelten Verzeichnissen
lassen sich hierarchische Strukturen abbilden

ourapp.com/user/123/profile

```
WEBDEVSHOWCASE
> .nuxt
< .output/public
> node_modules
< pages
  < user/[id]
    < profile.vue
    < index.vue
  > public
  > server
```

```
pages > user > [id] > profile.vue > {} template
1   <script setup>
2   |   const route = useRoute()
3   </script>
4
5   <template>
6   |   <p>Profil von Benutzer ID: {{ route.params.id }}</p>
7   </template>
```



TYPISCHE FEHLER BEIM ROUTING

- Dateien außerhalb von pages/ werden nicht gerendert
- Falsche Pfadangaben
- Tippfehler bei dynamischen Parametern
- Mehrere dynamische Routen auf derselben Ordner Hierarchieebene

Daher:

1. Struktur klar und einfach halten
2. Parameter sinnvoll benennen (SEO-optimiert)
3. Konsistente Layouts und Navigation nutzen

NUXT ÜBUNG 2

ERSTELLUNG VON SEITEN UND NAVIGATION

Aufgabe:

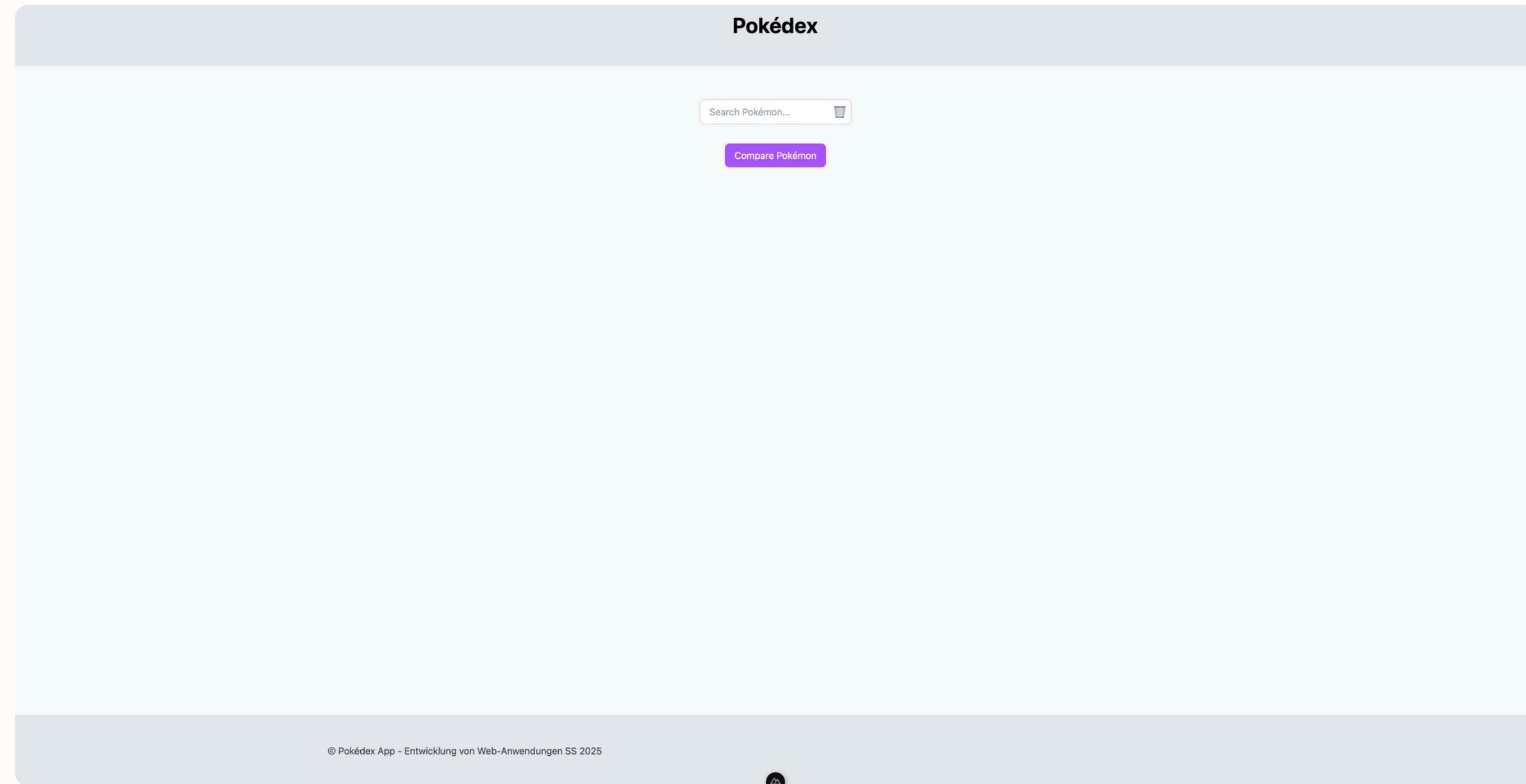
- Detailansicht Seite anlegen
- Button zurück zur Startseite auf Detailseite einfügen
 - Funktion hinter Button "Back to List"
- Error Seite anlegen
- Extra Layout für Errors einfügen
- Styling der Seite über CSS-File anpassen

Für die Schnellen:

- Anlegen neuer Seite "Compare"
 - Hinzufügen zweier Suchfelder
 - Hinzufügen eines Buttons der zurück zur Startseite führt
- Anlegen eines neuen Buttons auf Startseite, welcher zur neuen Compare Seite führt

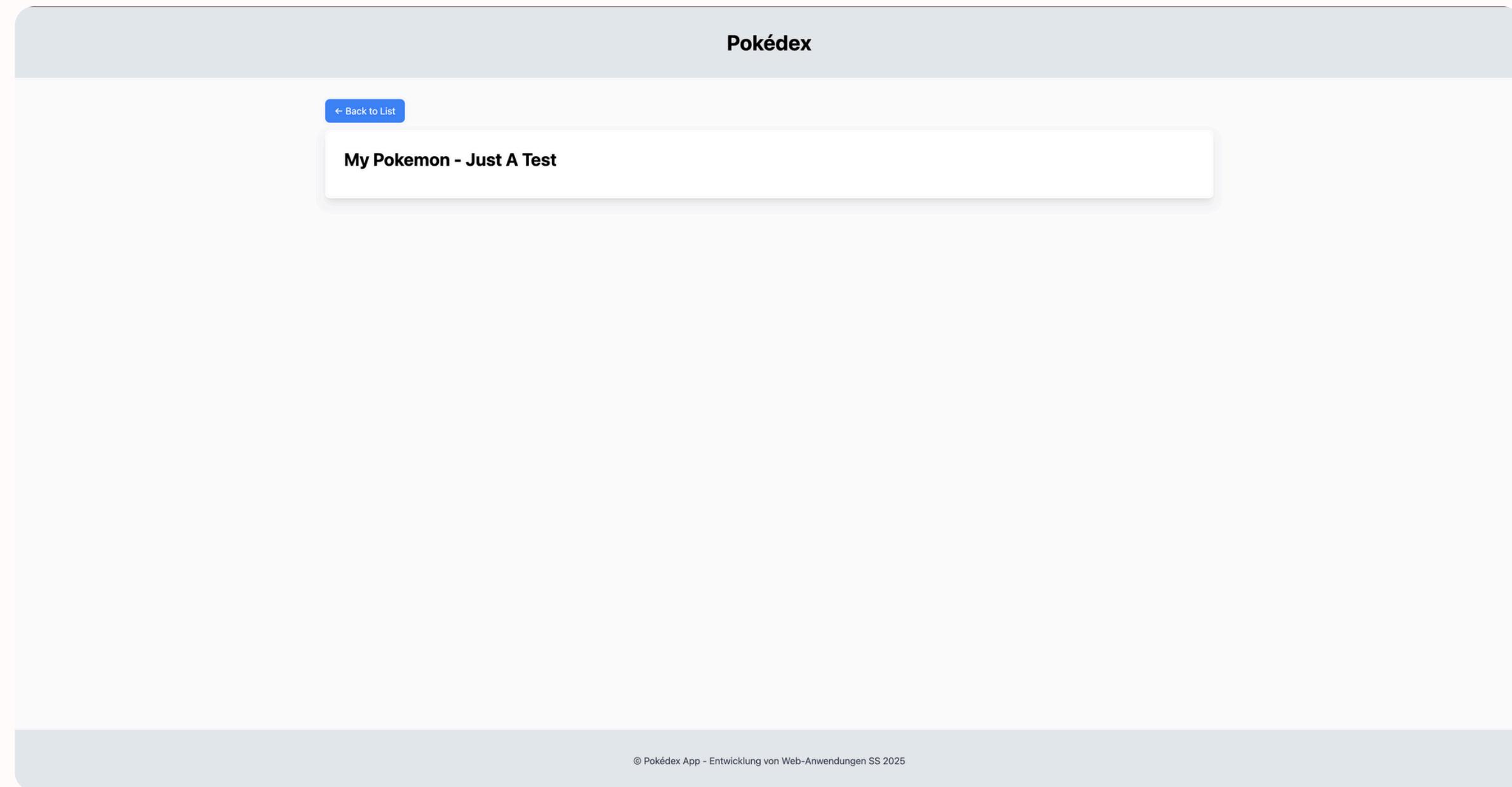
NUXT ÜBUNG 2 - ERSTELLUNG VON SEITEN UND NAVIGATION

Startseite



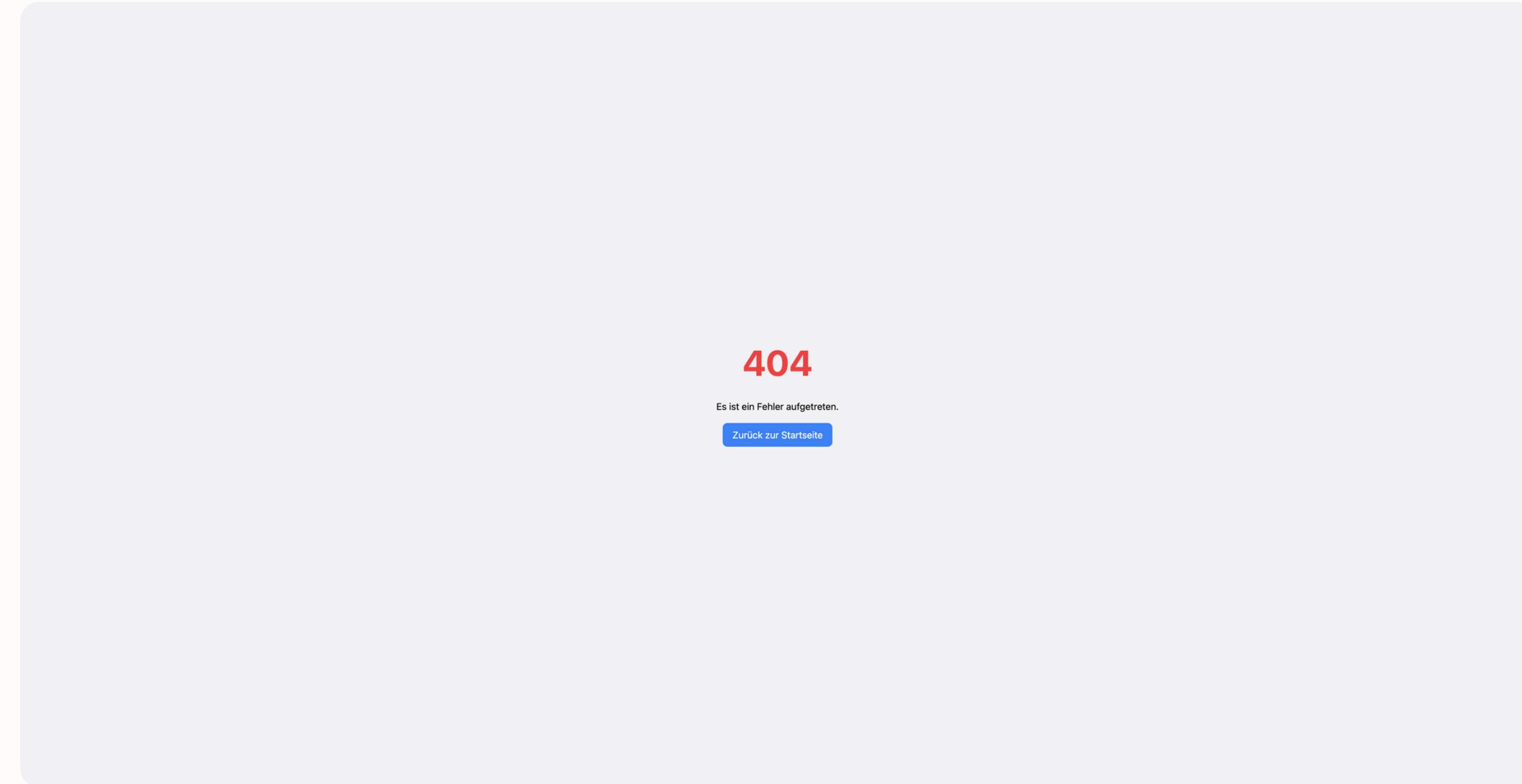
NUXT ÜBUNG 2 - ERSTELLUNG VON SEITEN UND NAVIGATION

Detail-Seite



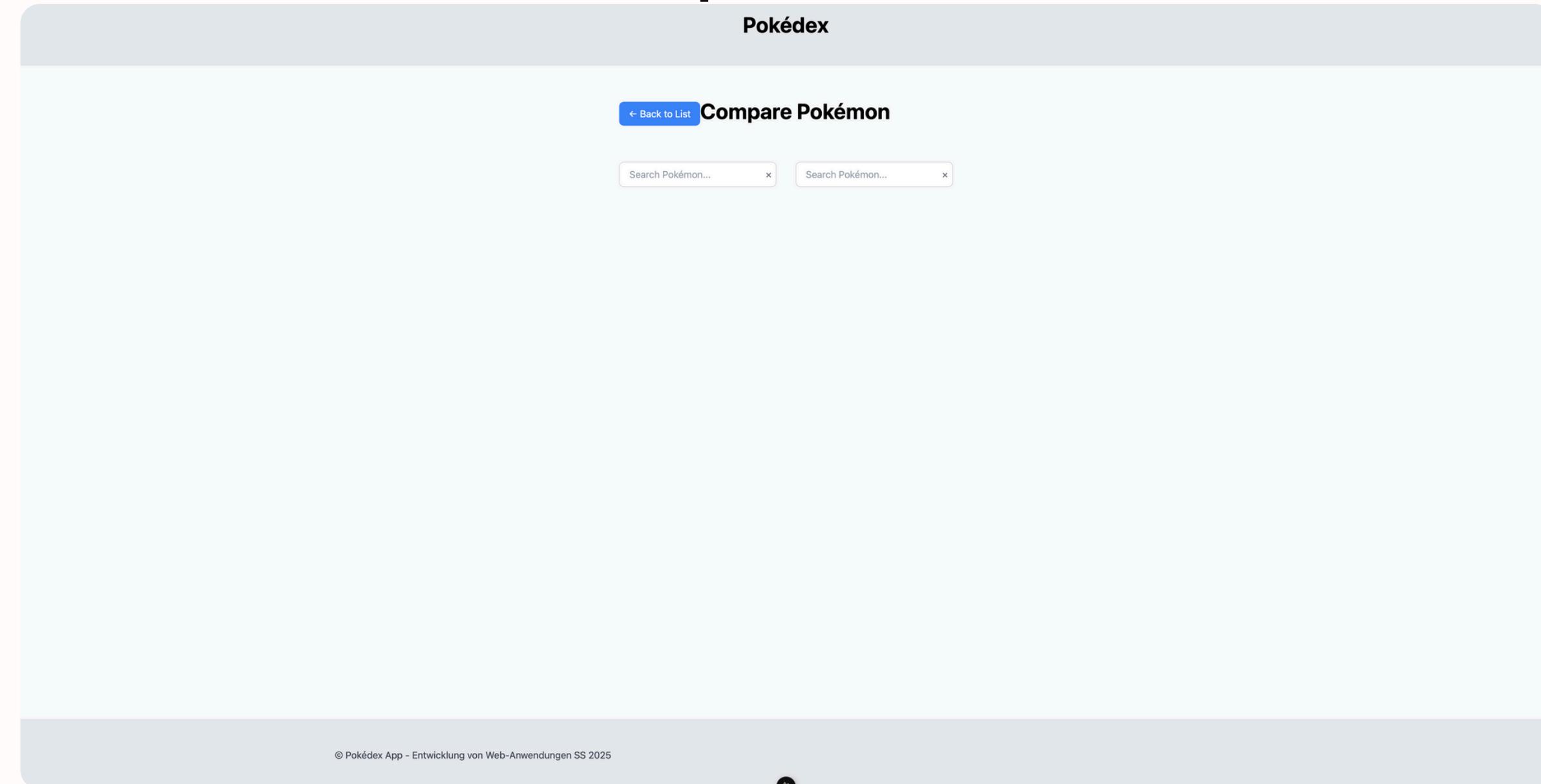
NUXT ÜBUNG 2 - ERSTELLUNG VON SEITEN UND NAVIGATION

Error-Seite



NUXT ÜBUNG 2 - ERSTELLUNG VON SEITEN UND NAVIGATION

Compare-Seite



NUXT ÜBUNG 2

Detailansicht Seite inkl. "Zurück"-Button

```
▼ [name].vue M X
pages > pokemon > ▼ [name].vue > ...
You, 1 second ago | 2 authors (You and one other)
1 <template>
2   <div class="container">
3     <div class="detail-header">
4       <button @click="navigateBack" class="back-button">< Back to List</button>
5     </div>
6     <div class="detail-container">
7       <h1 class="detail-title">My Pokemon - Just a test</h1>
8     </div>
9   </div>
10 </template>
11
12 <script setup>
13 const navigateBack = () => {
14   navigateTo("/");
15 };
16 </script>
```

NUXT ÜBUNG 2

Error Seite + Error Layout

```
▼ error.vue ×  
pages > ▼ error.vue > {} script setup  
      Simon Wimmer, 20 hours ago | 1 author (Simon Wimmer)  
1  <template>  
2  |   <div>  
3  |     <p>Es ist ein Fehler aufgetreten.</p>  
4  |   </div>  
5  </template>  
6  
7  <script setup>  
8  definePageMeta({  
9    layout: 'error'  
10 })  
11 </script>      Simon Wimmer, 20 hours ago • refactor: enhance layout structure and styles, ...
```

```
▼ error.vue ×  
layouts > ▼ error.vue > ...  
      Simon Wimmer, 20 hours ago | 1 author (Simon Wimmer)  
1  <template>  
2  |   <div class="error-container">  
3  |     <h1 class="error-title">404</h1>  
4  
5  |     <main class="py-4">  
6  |       <slot />  
7  |     </main>  
8  
9  |     <NuxtLink to="/" class="error-link">  
10 |       Zurück zur Startseite  
11 |     </NuxtLink>  
12 |   </div>  
13 </template>  
14
```

NUXT ÜBUNG 2

Compare Seite inkl. Suchfelder & "Zurück"-Button

```
▼ compare.vue ×  
pages > ▼ compare.vue > ...  
Simon Wimmer, 20 hours ago | 2 authors (You and one other)  
1 <template>  
2   <div class="container">  
3     <div class="compare-container">  
4       <button @click="navigateBack" class="back-button">< Back to List</button>  
5       <h1 class="compare-title">Compare Pokémons</h1>  
6     </div>  
7  
8     <div class="compare-search-container">  
9       <div class="compare-search">  
10      <div class="relative">  
11        <input type="text" :placeholder="`Search Pokémons ...`" class="search-input" />  
12        <button class="clear-button">x</button>  
13      </div>  
14    </div>  
15    <div class="compare-search">  
16      <div class="relative">  
17        <input type="text" :placeholder="`Search Pokémons ...`" class="search-input" />  
18        <button class="clear-button">x</button>  
19      </div>  
20    </div>  
21  </div>  
22 </div>  
23 </template>  
24  
25 <script setup>  
26 const navigateBack = () => {  
27   navigateTo("/");  
28 };  
29 </script>
```

LEARNINGS & AUSBLICK

🔑 Learnings:

- Vite unterstützt bei der Erstellung von Projekten
- NuxtJS macht Vue-Projekte strukturierter, schneller und SEO-freundlich
- Ideal für moderne Webapps dank SSR, SSG und automatischem Routing
- Vue.js mit Nuxt ist ein mega cooles Framework 

🔭 Ausblick:

- Komponenten und Statemanagement
- Datenabruft über API
- Deployment des Projekts



VUE.JS & NUXT

23. April 2025

Milena Kübler & Simon Wimmer & Max Tyrchan
Edv: 143404a Entwicklung von Web-Anwendungen

ZIEL & GLIEDERUNG



Unser Ziel ist es euch in den nächsten beiden Vorlesungen die Grundlagen von Vue.js und Nuxt zu vermitteln.

- 01** Komponenten und Datenverwaltung in Nuxt
- 02** Datenabruf und Darstellung in Nuxt
- 03** Deployment
- 04** Abschluss und Zusammenfassung

KOMPONENTEN IN NUXT

Was sind Komponenten in Nuxt?

- Basieren auf Vue-Komponenten
- Wiederverwendbare, gekapselte UI-Elemente
- Bestehen meist aus einem Template, einem Script und optional Styles

Automatisches Component-Registering in Nuxt

- Kein manueller Import notwendig
- Komponenten im components/- Ordner ablegen

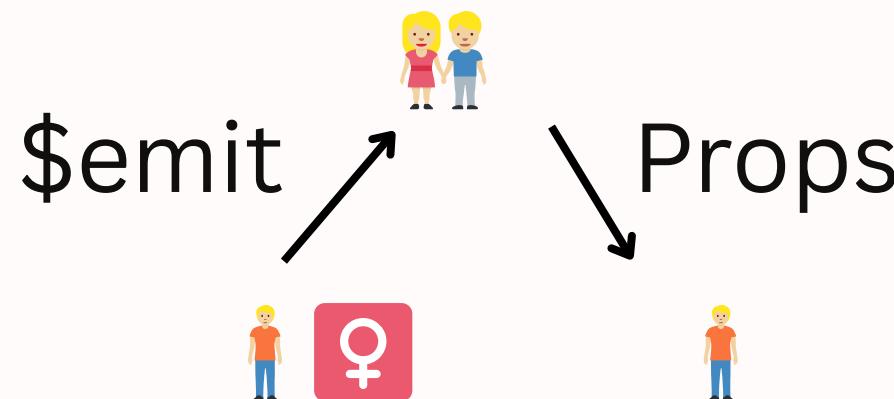
 Spart Zeit und reduziert Boilerplate-Code

KOMMUNIKATION ZWISCHEN KOMPONENTEN

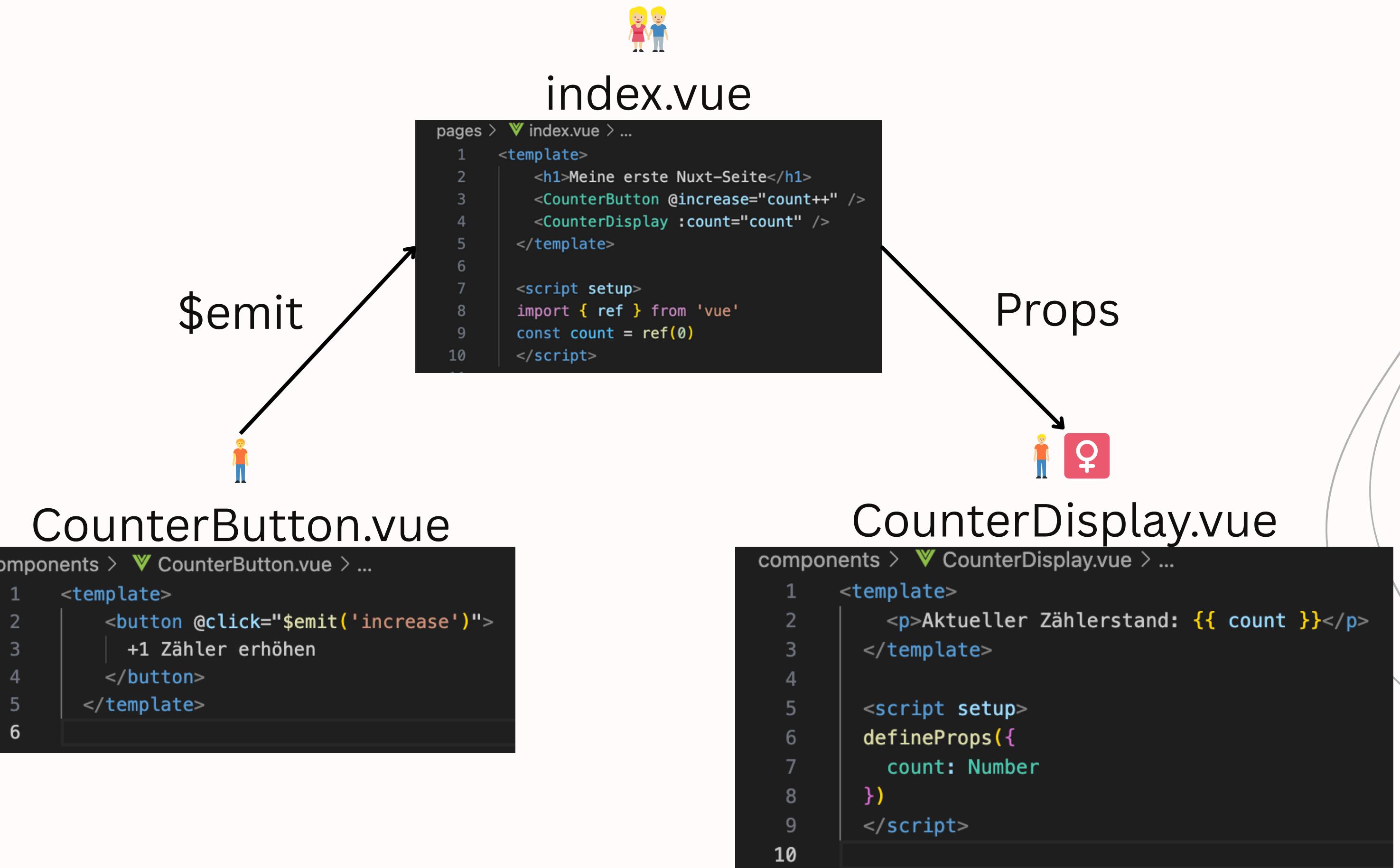


Props & Events

- Kommunikation von Child an Parent über Event-emit mit `$emit`
- Kommunikation von Parent an Child per Datenübergabe mit `Props`



BEISPIEL FÜR KOMPONENTEN & KOMMUNIKATION



EINFÜHRUNG IN STATE MANAGEMENT MIT PINIA



24

Was ist ein Store?

- Zentraler Ort an dem Daten (States) gespeichert und verwaltet werden
- Alle Komponenten einer Anwendung können darauf zugreifen und bleiben dadurch synchron

25

Bäckerei - Beispiel:

- Bäckerei mit mehreren Verkaufsständen (Komponenten)
- Alle Stände müssen wissen, wie viele Brezeln es noch gibt
- Verkauft Stand A eine Brezel müssen auch Stand B und C wissen, dass weniger da sind
- Dazu gibt es einen zentralen Zettel - den Store - der immer aktuell ist

EINFÜHRUNG IN STATE MANAGEMENT MIT PINIA



? Warum State Management?

- Wenn viele Komponenten gleichen Zustand teilen
- Stores können von jeder Komponente aus verwendet werden (kein Props-drilling)

🍍 Was ist Pinia?

- Offizielles State-Management-Tool für Vue 3 und Nuxt
- Einfacher, modularer, Typescript-freundlicher als Vuex
- Nachfolger von Vuex

→ npm install @pinia/nuxt

→ einbinden in nuxt.config.ts - modules: ['@pinia/nuxt']

VORTEILE VON PINIA



Vorteil	Beschreibung
Reaktivität	Stores sind automatisch reaktiv
Integration	Funktioniert nahtlos
Modularität	Mehrere Stores möglich
DevTools	Stores sind in Vue DevTools sichtbar

KOMMUNIKATION MIT STORES



stores/counter.ts

```
stores > TS counter.ts > [o] useCounterStore > actions
1 import { defineStore } from 'pinia'
2
3 export const useCounterStore = defineStore('counter', {
4   state: () => ({
5     count: 0
6   }),
7   actions: {
8     increase() {
9       this.count++
10    }
11  }
12})
```

components/CounterButton.vue

```
components > ▼ CounterButton.vue > {} script setup
1 <template>
2   <button @click="increase">
3     +1 Zähler erhöhen
4   </button>
5 </template>
6
7 <script setup>
8 import { useCounterStore } from '~/stores/counter'
9 const counter = useCounterStore()
10
11 function increase() {
12   counter.increase()
13 }
14 </script>
```

index.vue

```
pages > ▼ index.vue > ...
1 <template>
2   <h1>Meine erste Nuxt-Seite</h1>
3   <CounterButton />
4   <CounterDisplay />
5 </template>
```

components/CounterDisplay.vue

```
components > ▼ CounterDisplay.vue > ...
1 <template>
2   <p>Aktueller Zählerstand: {{ counter.count }}</p>
3 </template>
4
5 <script setup>
6 import { useCounterStore } from '~/stores/counter'
7 const counter = useCounterStore()
8 </script>
```

NUXT ÜBUNG 3 KOMPONENTEN & STATE MANAGEMENT



Aufgabe:

- Erstellung und Nutzung von wiederverwendbaren Komponenten - *PokemonCard.vue*
- Einbindung der erstellten Komponenten in *index.vue*
- Installation des Moduls von Pinia - *npm install @pinia/nuxt* Anpassung der *nuxt.config.ts* - *modules: ['@pinia/nuxt']*
- Einführung des State Managements mit Pinia (*usePokemonStore.ts*)
 - State Attribute: *allPokemon*, *searchQuery*, *selectedPokemon*
 - getters: *filteredPokemon* (um Pokemons zu filtern)
 - actions: *fetchAllPokemon*, *fetchPokemonByName* (zunächst mit den Dummy Daten, wird später ersetzt)

Für die Schnellen:

- Nutzung des State-Managements auf Compare-Seite (*compare.ts*)

NUXT ÜBUNG 3 KOMPONENTEN & STATE MANAGEMENT



usePokemonStore.ts

```
stores > TS usePokemonStore.ts > ...
You, vor 26 Sekunden | 2 authors (maxtyrchan and one other)
1 import { defineStore } from "pinia";
2 import type { Pokemon } from "~/types/pokemon";
3 import { mockAllPokemon } from "~/types/mock-data";
4
5 export const usePokemonStore = defineStore("pokemon", {
6   state: () => {
7     allPokemon: Pokemon[];
8     searchQuery: string;
9     selectedPokemon: Pokemon | null
10    } => ({
11      allPokemon: [],
12      searchQuery: "",
13      selectedPokemon: null
14    }),
15
16   getters: {
17     filteredPokemon: (state) => {
18       return state.allPokemon.filter((p) =>
19         p.name.toLowerCase().includes(state.searchQuery.toLowerCase())
20       );
21     },
22   },
23
24   actions: {
25     fetchAllPokemon() {
26       this.allPokemon = mockAllPokemon;
27     },
28
29     fetchPokemonByName(name: string) {
30       this.selectedPokemon = this.allPokemon.find(
31         (pokemon) => pokemon.name.toLowerCase() === name.toLowerCase()
32       ) || null;
33     },
34
35     clearSearchQuery() {
36       this.searchQuery = "";
37     },
38   },
39 });


```

index.vue

```
pages > ▼ index.vue > ...
You, vor 5 Stunden | 3 authors (maxtyrchan and others)
1 <template>
2   <div class="container">
3     <div class="search-container">
4       <div class="search-wrapper">
5         <input v-model="store.searchQuery" type="text" placeholder="Search Pokémon..." class="search-input" />
6         <button v-if="store.searchQuery" @click="store.clearSearchQuery()" class="clear-button"
7           aria-label="Clear search">x</button>
8       </div>
9     </div>
10
11   <div class="flex justify-center mb-6">
12     <NuxtLink to="/compare" class="compare-button">
13       Compare Pokémon
14     </NuxtLink>
15   </div>
16
17   <div class="pokemon-grid">
18     <PokemonCard v-for="pokemon in store.filteredPokemon" :key="pokemon.id" :pokemon="pokemon" />
19   </div>
20 </div>
21 </template>
22
23 <script setup>
24 const store = usePokemonStore();
25
26 onMounted(() => {
27   store.fetchAllPokemon();
28 });
29 </script>
```

NUXT ÜBUNG 3 KOMPONENTEN & STATE MANAGEMENT

Component: PokemonCard.vue

```
components > ▼PokemonCard.vue > ...
  Wimmer Simon (XC-CT/EMT4), vor 5 Stunden | 2 authors (maxtyrchan and one other)
1  <template>
2    <NuxtLink :to=`/pokemon/${pokemon.name}`> class="pokemon-card">
3      <h2 class="pokemon-name">
4        {{ pokemon.name }}
5      </h2>
6      <div class="type-container justify-center">
7        <div v-for="(type, index) in pokemon.types" :key="index">
8          <p>{{ type }}</p>
9        </div>
10       </div>
11     </NuxtLink>
12   </template>
13
14  <script setup>
15    defineProps({
16      pokemon: {
17        type: Object,
18        required: true,
19      },
20    });
21  </script>
```

Zusatzaufgabe (Store: compare.ts)

```
stores > TS compare.ts > ...
  maxtyrchan, gestern | 1 author (maxtyrchan)
1  import { defineStore } from "pinia";
2  import type { Pokemon } from "~/types/pokemon";
3  import { mockAllPokemon } from "~/types/mock-data";
4
5  export const useCompareStore = defineStore("compare", {
6    state: () => {
7      allPokemon: Pokemon[];
8      selectedPokemon: (Pokemon | null)[];
9      searchQueries: string[];
10     } => ({
11       allPokemon: [],
12       selectedPokemon: [null, null],
13       searchQueries: ["", ""],
14     }),
15
16    getters: {
17      filteredPokemon: (state) => (index: number) => {
18        return state.allPokemon.filter((p) =>
19          p.name.toLowerCase().includes(state.searchQueries[index].toLowerCase())
20        );
21      },
22    },
23
24    actions: {
25      fetchAllPokemon() {
26        this.allPokemon = mockAllPokemon;
27      },
28
29      selectPokemon(pokemon: Pokemon, index: number) {
30        this.selectedPokemon[index] = pokemon;
31        this.searchQueries[index] = "";
32      },
33
34      clearSearchQuery(index: number) {
35        this.searchQueries[index] = "";
36      },
37    },
38  });
39
```

WEITER GEHT ES MIT DER BESPRECHUNG UM 11:00 UHR

02. DATENABRUF

⚡ Eigenen Server erstellen mit Nitro

🧲 Einfacher Abruf mit \$fetch

⟳ Daten abrufen mit useAsyncData()

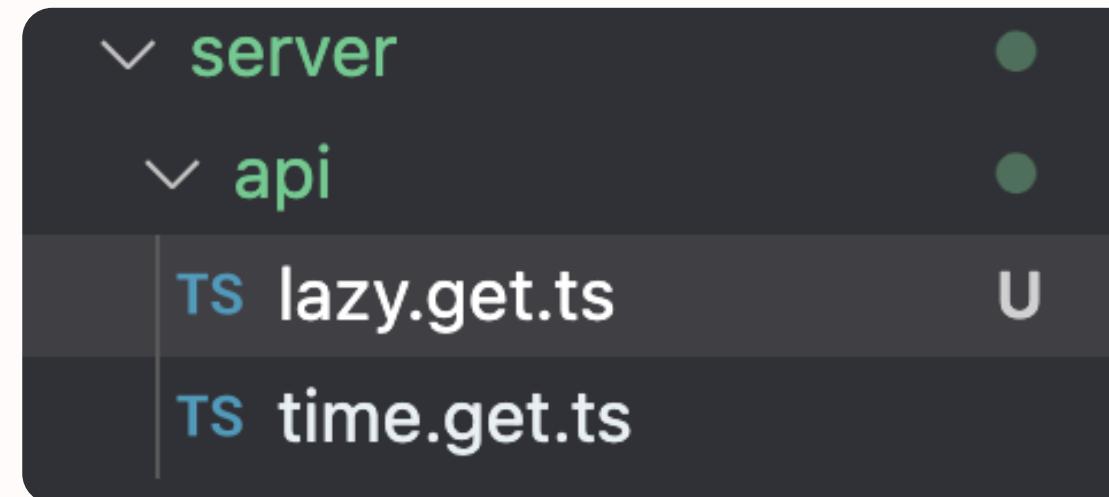
⬇️ Daten abrufen mit useFetch()

🔗💥 Vergleich useAsyncData() & useFetch()

🌐 Umgang mit async. Daten

EIGENEN SERVER ERSTELLEN MIT NITRO

- Nitro ist das serverseitige Framework von Nuxt
- Basiert ebenfalls auf Dateisystem von Nuxt
- Serverlogik direkt im Nuxt-Projekt
- Kein separates Backend notwendig





EINFACHER ABRUF MIT \$FETCH

- Einfachster Weg einen Server Request auszuführen
 - Leitet nicht automatisch Serverside geladene Daten zu Client weiter! → Doppelabruf !
 - Lediglich für reine Client-Interaktionen geeignet
- Lösungen bieten useFetch & useAsyncData



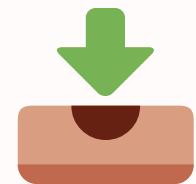
DATEN ABRUFEN MIT USEASYNCDATA()

- SSR-freundliche Composable zum asynchronen Laden von Daten
- Baut intern auf \$fetch auf, nutzt aber zusätzlich einen expliziten Cache-Schlüssel
- Hilft dabei, Daten genau einmal zu laden (Server-seitig) und via Payload automatisch an den Client weiterzugeben
- Sinnvoll bei Seiten, deren Inhalt sich selten ändern



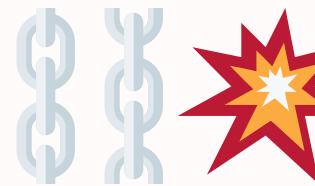
Typische Fehler:

- Cache-Key vergessen oder doppelt vergeben
- Zu viele useAsyncData Aufrufe pro Seite



DATEN ABRUFEN MIT USEFETCH()

- `useFetch()`: Komfortable Methode, um Daten mit SSR-Support von APIs zu laden
- Nutzt intern `useAsyncData` und `$fetch`
- Automatischer Transfer von serverseitig geladenen Daten zum Client (Payload)
- Vermeidet doppelte API-Abfragen (kein erneutes Laden beim Hydrieren im Client)



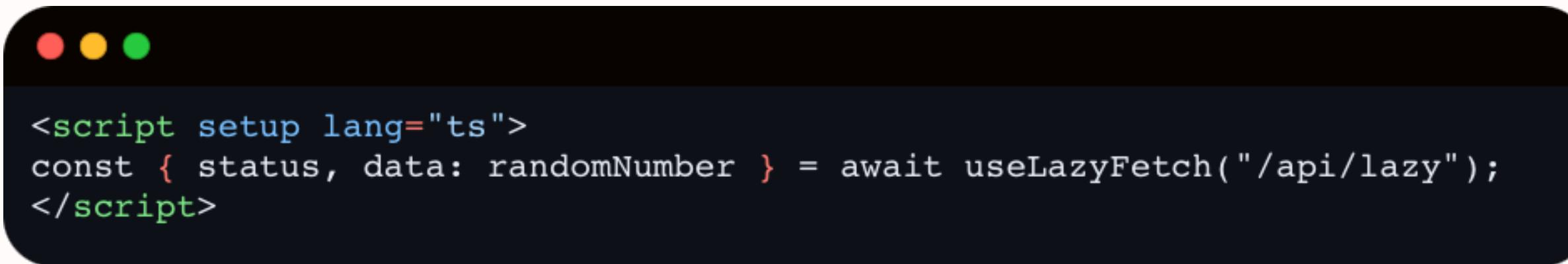
VERGLEICH USEFETCH & USEASYNCDATA

Aspekt	useAsyncData	useFetch
Basis	<code>\$fetch + Cache-Schlüssel</code>	<code>\$fetch + Automatischer Key</code>
Cache-Schlüssel	Explizit definiert	Automatisch generiert
Einsatzzweck	Explizites Caching, SSR-optimierte Daten	Universell für einfache Requests
Typische Verwendung	CMS-Inhalte, Blogposts, statische Inhalte	Module, Listen, generische API-Daten



UMGANG MIT ASYNCHRONEN DATEN

- Standardmäßig warten Fetch Komponenten, auf die Auflösung ihrer asynchronen Funktion, bevor sie zu einer neuen Seite navigieren
- Blockiert damit clientseitige Navigation
- Dieses Feature mit der Lazy-Option ignoriert werden



```
<script setup lang="ts">
const { status, data: randomNumber } = await useLazyFetch("/api/lazy");
</script>
```

NUXT ÜBUNG 4 API-MANAGEMENT

Aufgabe:

- Abruf der Daten aus der öffentlichen PokéAPI
- Darstellung der abgerufenen Daten in der Anwendung (ersetzen der Dummy-Daten durch die API-Daten) in usePokemonState.ts
- Bilder der Pokemons an die entsprechenden Stellen einfügen

Hinweis:

- Bei einem Mehrfachaufruf einer API wie wir sie benötigen, lässt sich kein useFetch nutzen!
- Stattdessen benötigen wir \$fetch mit await Promise.all()

Für die Schnellen:

- Ersetzen der Dummy-Daten auf Compare-Seite

NUXT ÜBUNG 4 API-MANAGEMENT

usePokemonStore.ts

```
actions: {
  async fetchAllPokemon() {
    this.loading = true;
    this.error = false;
    try {
      const responses = await Promise.all(
        Array.from({ length: 151 }, (_, i) =>
          $fetch(`https://pokeapi.co/api/v2/pokemon/${i + 1}`).catch(
            (error) => {
              console.error(`Error fetching Pokemon ${i + 1}:`, error);
              return null;
            }
          )
        );
      this.allPokemon = responses.filter((p): p is Pokemon => p !== null);
    } catch (error) {
      console.error("Error fetching Pokemon:", error);
      this.error = true;
      this.allPokemon = [];
    } finally {
      this.loading = false;
    }
  },
},
```

```
async fetchPokemonByName(name: string) {
  this.loading = true;
  this.error = false;
  try {
    this.selectedPokemon = await $fetch(
      `https://pokeapi.co/api/v2/pokemon/${name}`
    );
  } catch (err) {
    console.error("Error fetching Pokemon:", err);
    this.error = true;
    this.selectedPokemon = null;
  } finally {
    this.loading = false;
  }
},
```

NUXT ÜBUNG 4 API-MANAGEMENT

compare.ts

```
actions: {
  async fetchAllPokemon() {
    try {
      const responses = await Promise.all(
        Array.from({ length: 151 }, (_, i) =>
          $fetch(`https://pokeapi.co/api/v2/pokemon/${i + 1}`).catch(
            (error) => {
              console.error(`Error fetching Pokemon ${i + 1}:`, error);
              return null;
            }
          )
        );
      // Type assertion might be needed here if the API response format isn't certain.
      this.allPokemon = responses.filter((p): p is Pokemon => p !== null);
    } catch (error) {
      console.error("Error fetching Pokemon:", error);
      this.allPokemon = [];
    }
  },
},
```

NUXT ÜBUNG 4 API-MANAGEMENT

PokemonCard.Vue

```
<template>
  <NuxtLink :to=`/pokemon/${pokemon.name}` class="pokemon-card">
    
    <h2 class="pokemon-name">
      {{ pokemon.name }}
    </h2>
    <div class="type-container">
      <TypeBadge
        v-for="type in pokemon.types"
        :key="type.type.name"
        :type="type.type.name"
      />
    </div>
  </NuxtLink>
</template>

<script setup>
defineProps({
  pokemon: {
    type: Object,
    required: true,
  },
});
</script>
```

WEITER GEHT ES MIT DER BESPRECHUNG UM 12:20 UHR

03 DEPLOYMENT

 Deployment bezeichnet den Prozess, bei dem eine entwickelte Softwareanwendung von der Entwicklungsumgebung auf eine Zielumgebung übertragen wird, sodass sie für Endnutzer zugänglich und nutzbar ist.

 Ziel: Die Anwendung soll stabil, sicher und performant in der Produktionsumgebung laufen und den Nutzern den gewünschten Funktionsumfang bieten.

WARUM IST DEPLOYMENT WICHTIG?

- 👉 Zugänglich machen der Anwendung für Nutzer
- 🔄 Aktualisierung der Anwendung
- 🛡️ Qualitätssicherung durch regelmäßige Updates

DER KLASSISCHE DEPLOYMENT-PROZESS

1. Lokale Entwicklung
2. Build (z. B. mit Vite)
3. Testen (automatisiert/manuell)
4. Staging (Vorschau)
5. Produktion (Live-Schaltung)

📌 Ziel: So wenig Reibung wie möglich zwischen den Phasen!

DEPLOYMENT-STRATEGIEN

Strategie	Beschreibung	Beispiel
Manuelles Deployment	Upload via FTP, Kommandozeile o. ä.	Alt: Apache, Netlify ZIP
Automatisiertes Deployment	Toolgesteuert (CI/CD)	z.B. Vercel, Netlify, GitHub Actions
Continuous Deployment	z.B. Jeder Push wird direkt deployed (nach Tests)	Vercel, Cloudflare Pages
Staged Deployment	Tests in „Staging“ vor Live-Schaltung	z.B. mit Preview URLs

HERAUSFORDERUNGEN BEIM DEPLOYMENT

✗ Fehlkonfigurationen

- z. B. falscher Build-Befehl oder Routing

🔒 Sicherheitslücken

- z. B. vergessene Umgebungsvariablen oder offene APIs

🧪 Fehlende Tests

- Code läuft lokal, aber nicht im Web

📈 Performance-Probleme

- Zu große Bundles, fehlendes Caching

🕵️♂️ Keine Logs oder Monitoring

- Fehler schwer nachzuvollziehen

🎯 Ziel: Automatisierung, Vorschau, Fehlerfrüherkennung & Monitoring nutzen!

BEST PRACTICES

✓ Automatisierung

✓ Rollback-Strategien

✓ Monitoring

WAS IST ▲Vercel ?

- 🌐 Plattform zum Hosting von Frontend-Apps
- ❤️ Ideal für moderne JS-Frameworks (NuxtJS, Next.js etc.)
- 🚀 Automatischer Build + Deployment über Git
- 🌐 Globales CDN für hohe Ladegeschwindigkeit

WARUM VERCEL FÜR NUXTJS?

Feature	Vorteil
Automatische Erkennung von Nuxt	Kein Setup notwendig
SSR & SSG-Unterstützung	Optimal für SEO & Performance
GitHub Integration	Deployment bei jedem Push
Vorschau-URLs	Testing vor dem Go-Live

NUXT ÜBUNG 5 - DEPLOYMENT

Aufgabe:

- Stelle sicher, dass dein NuxtJS-Projekt auf GitHub verfügbar ist
- Erstelle ein Konto auf vercel.com und verknüpfe es mit deinem GitHub-Konto
- Importiere dein Projekt in Vercel und überprüfe die automatisch erkannten Einstellungen
- Starte das Deployment und warte auf den Abschluss des Build-Prozesses
- Öffne die bereitgestellte Live-URL und überprüfe die Funktionalität deiner Anwendung

Für die Schnellen:

- Teste die Monitoring-Möglichkeiten von Vercel
- Mache Änderungen am Code auf einem neuen Branch und schau was mit dem Deployment passiert

HINWEISE

- Preview Deployments:
 - Jeder Push zu einem Branch erzeugt eine Vorschau-Deployment-URL
 - Hilfreich für Tests und Reviews vor dem Merge in den Hauptbranch
- Production Deployments:
 - Pushes zum Hauptbranch (z. B. main oder master) führen zu einem Produktions-Deployment
- Monitoring und Logs:
 - Vercel bietet ein Dashboard zur Überwachung von Deployments und Einsicht in Logs.

FAZIT

- ✓ Komponenten bieten Struktur und Wiederverwendbarkeit
- ✓ Pinia ermöglicht einfache, zentrale Zustandsverwaltung
- ✓ Nitro ist das serverseitige Framework von Nuxt
- ✓ useFetch() ist die einfachste Methode, um ext. Daten zu laden
- ✓ Deployment ist mehr als nur Hochladen
- ✓ Vercel + NuxtJS = ideales Duo

