

 Vue.js +  Nuxt

113489

Web Development Workshop

Julia Ebert (je073) | Lars Gerigk (lg107) | Joel Starkov (js486)

Ablauf

Vorlesung 1

11.11.2025



Vorlesung 2

18.11.2025

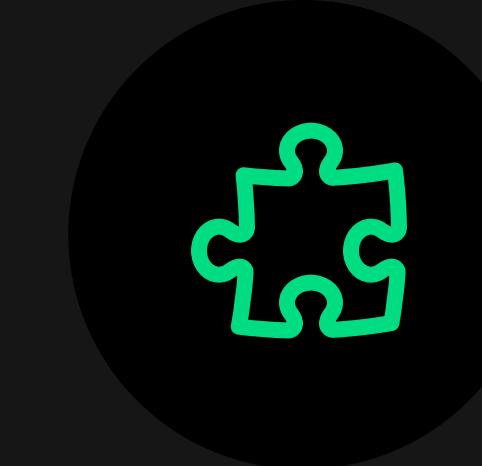


Vorlesung 3

25.11.2025



- Theorie
- Grundlagen Vue.js



- Erweiterungen
durch Nuxt



- Puffer
- Wunschthemen



Vorlesung 1

01 Theorie

- 1.1. Was ist Vue.js ?
- 1.2. Was ist Nuxt ?
- 1.3. Verwendung von Vue.js & Nuxt
- 1.4. Unterschiede & Gemeinsamkeiten

02 Grundlagen

- 2.1. Setup & Einstieg in die App
- 2.2. Zwei Arten Komponenten zu erstellen
- 2.3. Syntax (Text Interpolation, Reaktivität. Direktiven)
- 2.4. Event Handling



Was ist Vue.js ?

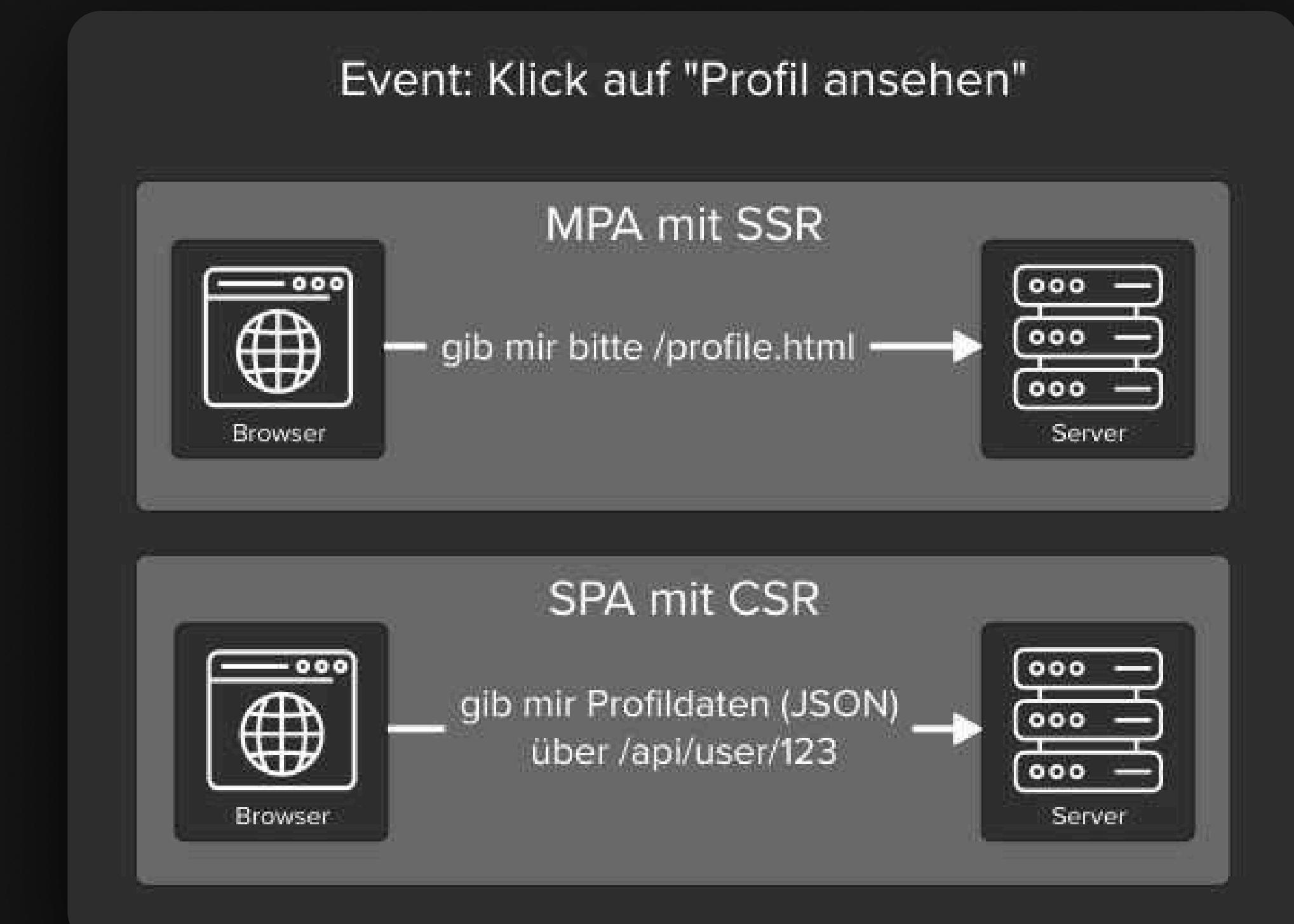
- das proggessive Framework
- Clientseitiges Javascript-Framework
- Erstellung von Single-Page-Webanwendungen nach dem MVVM-Muster
- HTML wird um proprietäre Attribute erweitert und ergänzt, die das Rendern steuern

- + Einfachheit
- + Flexibilität
- + Kann schrittweise in bestehende Projekte integriert werden



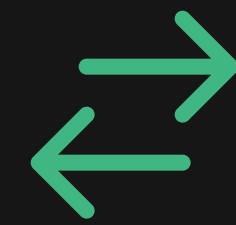
Was ist eine Single-Page-Application ?

- Entwickelten sich aus Multi-Page-Applications
 - Jede Seite hatte eigene HTML-Datei
 - Bei Klick auf Link, wurde von Server komplett neue Seite geladen
- AJAX und Start der Single-Page-Applications
 - Server liefert leere HTML-Seite und JavaScript-Bundle
 - Rendering im Browser
- Universal Rendering





Kernfunktionen von Vue.js



Reaktivität

Datenänderungen aktualisieren
automatisch die Ansicht



Deklaratives Rendering

Bindung von Daten an HTML mit
einfacher Syntax



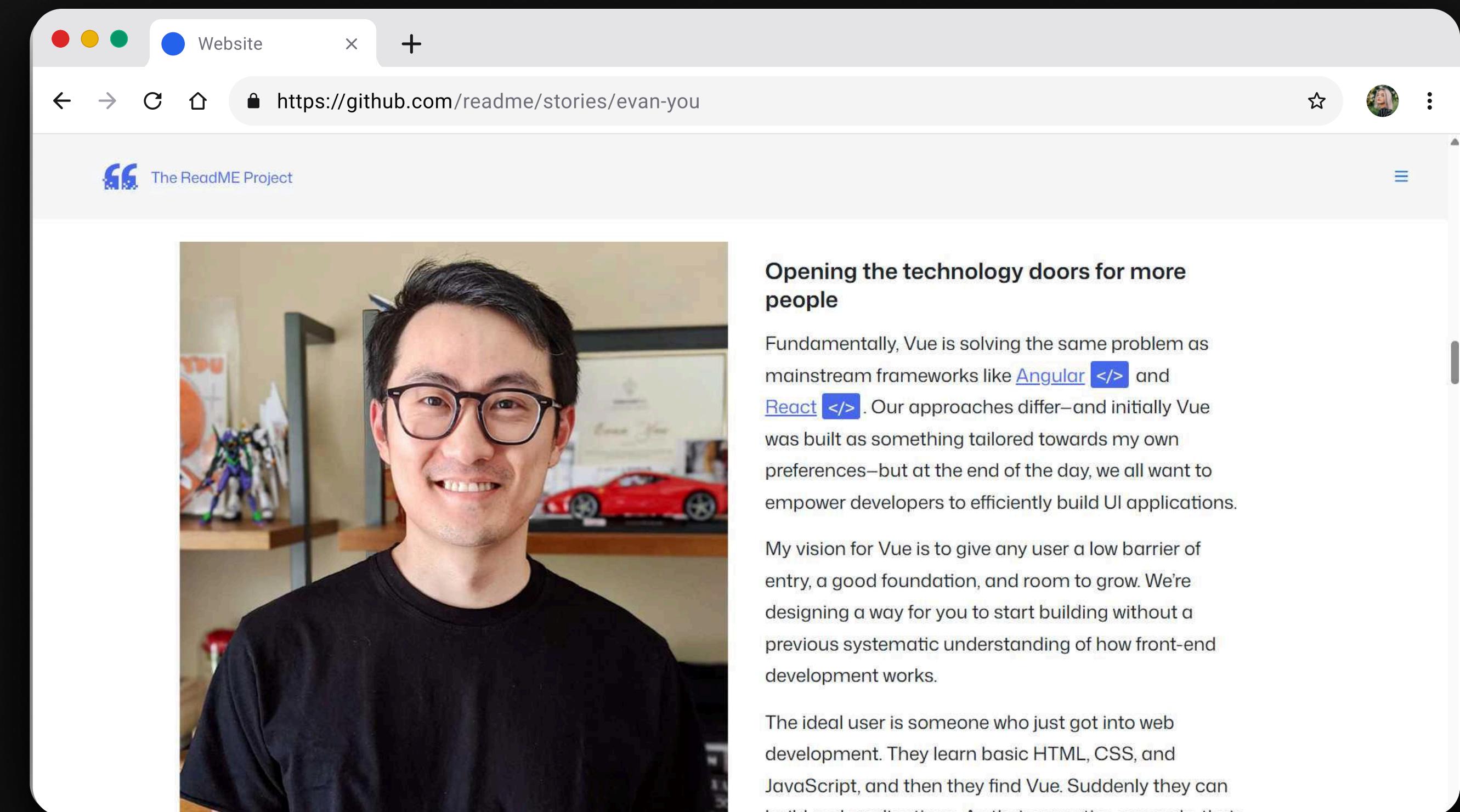
Komponenten

Wiederverwendbare, modulare
Bausteine



Fakten über Vue.js

- erste Version 2014
- Gründer: Evan You
- Vue.js → vom Hobby zum Vollzeitprojekt



Mehr dazu: <https://github.com/readme/stories/evan-you>



Fakten über Vue.js

- Funfact:
 - Versionsnamen werden aus Anime- und Mangaserien in alphabetischer Reihenfolge abgeleitet

Version	Release date	Title
3.5	September 1, 2024	Tengen Toppa Gurren Lagann ^[20]
3.4	December 28, 2023	Slam Dunk ^[21]
3.3	May 11, 2023	Rurouni Kenshin ^[22]
3.2	August 5, 2021	Quintessential Quintuplets ^[23]
3.1	June 7, 2021	Pluto ^[24]
3.0	September 18, 2020	One Piece ^[25]
2.7	July 1, 2022	Naruto ^[26]
2.6	February 4, 2019	Macross ^[27]
2.5	October 13, 2017	Level E ^[28]
2.4	July 13, 2017	Kill la Kill ^[29]
2.3	April 27, 2017	JoJo's Bizarre Adventure ^[30]
2.2	February 26, 2017	Initial D ^[31]
2.1	November 22, 2016	Hunter × Hunter ^[32]
2.0	September 30, 2016	Ghost in the Shell ^[33]
1.0	October 27, 2015	Evangelion ^[34]
0.12	June 12, 2015	Dragon Ball ^[35]
0.11	November 7, 2014	Cowboy Bebop ^[36]
0.10	March 23, 2014	Blade Runner ^[37]
0.9	February 25, 2014	Attack on Titan ^[38]



Was ist Nuxt ?

- Metaframework auf Basis von Vue.js
- Erweitert Vue um viele Funktionen, die man sonst manuell konfigurieren müsste (z.B. Routing, SSR, ...)
- Automatisiertes Rendering – wahlweise serverseitig, statisch oder clientseitig
- Integriertes Backend-Framework (Nitro)
- Klare Projektstruktur durch feste Ordner

- + Struktur
- + Performance
- + Fullstack



Vue.js vs Nuxt

	Vue.js	Nuxt.js
Routing	⚠️ (manuell)	✅ (automatisch)
SEO	🚫	✅
Rendering	⚠️ (nur Client-Side-Rendering)	✅ Server-, Static- oder Client-Side Rendering
automatische Imports	🚫	✅
Data-fetching	⚠️ (nur client-seitig)	✅ (client- und server-seitig)
Flexibilität	✅	⚠️ (viele Konventionen)



Verwendung von Vue.js & Nuxt

Vue.js

- eine einzige interaktive Seite (SPA)
- interne Web-Apps
- volle Kontrolle über Routing, Build-Prozess usw. will
- man nur Client-Side Rendering braucht

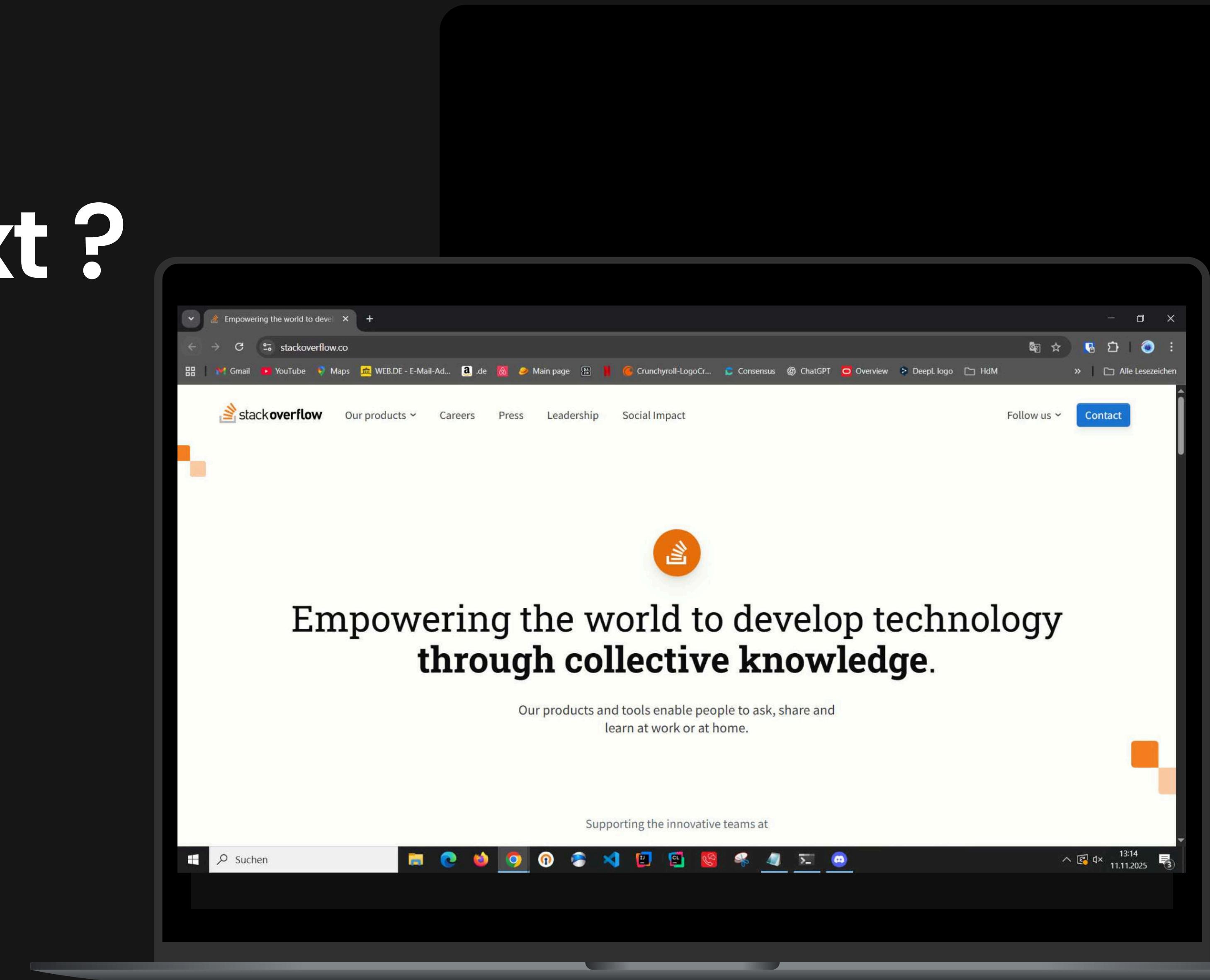
Nuxt.js

- SEO-optimierte oder öffentliche Webseite
- Seiten sollen schnell laden
- klare Projektstruktur
- man hat viele Routen



Wer nutzt Vue.js & Nuxt ?

zum Beispiel: stackoverflow





Ordnerstruktur

```
my-nuxt-app/
|
|   .nuxt/          # Automatisch generiert (Build-Output, Cache, Routen, usw.)
|   node_modules/    # Abhängigkeiten
|   public/          # Statische Dateien (z.B. favicon.ico, images, robots.txt)
|   app/
|     assets/        # Unkompilierte Ressourcen (z.B. Fonts, Bilder für Komponenten)
|     components/    # Globale Vue-Komponenten
|     composables/   # Reaktive Hilfsfunktionen (z.B. useAuth(), useFetchData())
|     layouts/        # Seiten-Layouts (z.B. default.vue, admin.vue)
|     middleware/    # Middleware für Routen (z.B. Auth-Check)
|     pages/          # Definiert automatisch das Routing-System
|     plugins/        # Client-/Server-Plugins (z.B. Axios, Pinia, i18n)
|     utils/          # Allgemeine Hilfsfunktionen
|     app.vue         # Haupteinstiegspunkt
|     app.config.ts   # API-Endpunkte und Server-Routen (Nuxt Server Engine)
|
|   nuxt.config.ts   # Hauptkonfigurationsdatei (Plugins, Build, RuntimeConfig)
|   package.json     # Projektabhängigkeiten
|   server           # API-Endpunkte und Server-Routen
```



Vorlesung 1

01 Theorie

- 1.1. Was ist Vue.js ?
- 1.2. Was ist Nuxt ?
- 1.3. Verwendung von Vue.js & Nuxt
- 1.4. Unterschiede & Gemeinsamkeiten

02 Grundlagen

- 2.1. Setup & Einstieg in die App
- 2.2. Zwei Arten Komponenten zu erstellen
- 2.3. Syntax (Text Interpolation, Reaktivität. Direktiven)



Setup - Aufgabe 01

Schritt 1: git clone <https://github.com/byGamsa/wdw-nuxt-js.git>

Schritt 2: überprüfe, dass npm und node.js auf dem aktuellen Stand sind

- node.js v24.11.0
- npm 11.6.2

Schritt 3: navigiere in einen Ordner wdw-nuxt-js/exercises/01-setup-project

Schritt 4: Führe die Befehle “npm install” und “npm run dev” aus



Wie wir es von HTML kennen

```
<!DOCTYPE html>
<html lang="de">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Vue Beispiel</title>
  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
  <style>
    button {
      background: #00ff99;
      color: #0a0a0a;
    }
  </style>
</head>
<body>
  <div id="app">
    <h1>{{ message }}</h1>
    <button @click="changeMessage">Text ändern</button>
  </div>

  <script>
    const { createApp } = Vue;

    createApp({
      data() {
        return {
          message: 'Hallo Welt!'
        };
      },
      methods: {
        changeMessage() {
          this.message = 'Text wurde geändert!';
        }
      }
    }).mount('#app');
  </script>
</body>
</html>
```



Syntax von Vue.js

- Single-File-Component
- Dateiendung: *.vue
- NEU: <template></template>



```
<template>
<div>
  <h1>{{ message }}</h1>
  <button @click="changeMessage">Text ändern</button>
</div>
</template>

<script>
export default {
  data() {
    return { message: 'Hallo Welt!' }
  },
  methods: {
    changeMessage() {
      this.message = 'Text wurde geändert!'
    }
  }
}
</script>

<style>
h1 {
  color: #00ff99;
}
</style>
```



Wie starte ich eine Vue.js Applikation ?

```
import { createApp } from 'vue'  
import App from './App.vue'  
  
createApp(App).mount('#app')
```

main.js



Mögliche Ordnerstruktur





Zwei Arten Komponenten zu definieren - Options API

- Logik wird innerhalb der JS-Anweisung
"export default {}" nach Optionen gegliedert



```
<script>
  export default {
    data() {
      return {
        count: 0
      }
    },
    methods: {
      increment() {
        this.count++
      }
    },
    mounted() {
      console.log(`The initial count is ${this.count}.`)
    }
  }
</script>

<template>
  <button @click="increment">Anzahl ist: {{ count }}</button>
</template>
```



Zwei Arten Komponenten zu definieren - Compositions API

- Komponentenlogik mit importierten API-Funktionen
- Wird in Single File Components (SFCs) meist mit `<script setup>` verwendet



```
<script setup>
import { ref, onMounted } from 'vue'

// reactive state
const count = ref(0)

// functions that mutate state and trigger updates
function increment() {
  count.value++
}

// lifecycle hooks
onMounted(() => {
  console.log(`The initial count is ${count.value}.`)
})
</script>

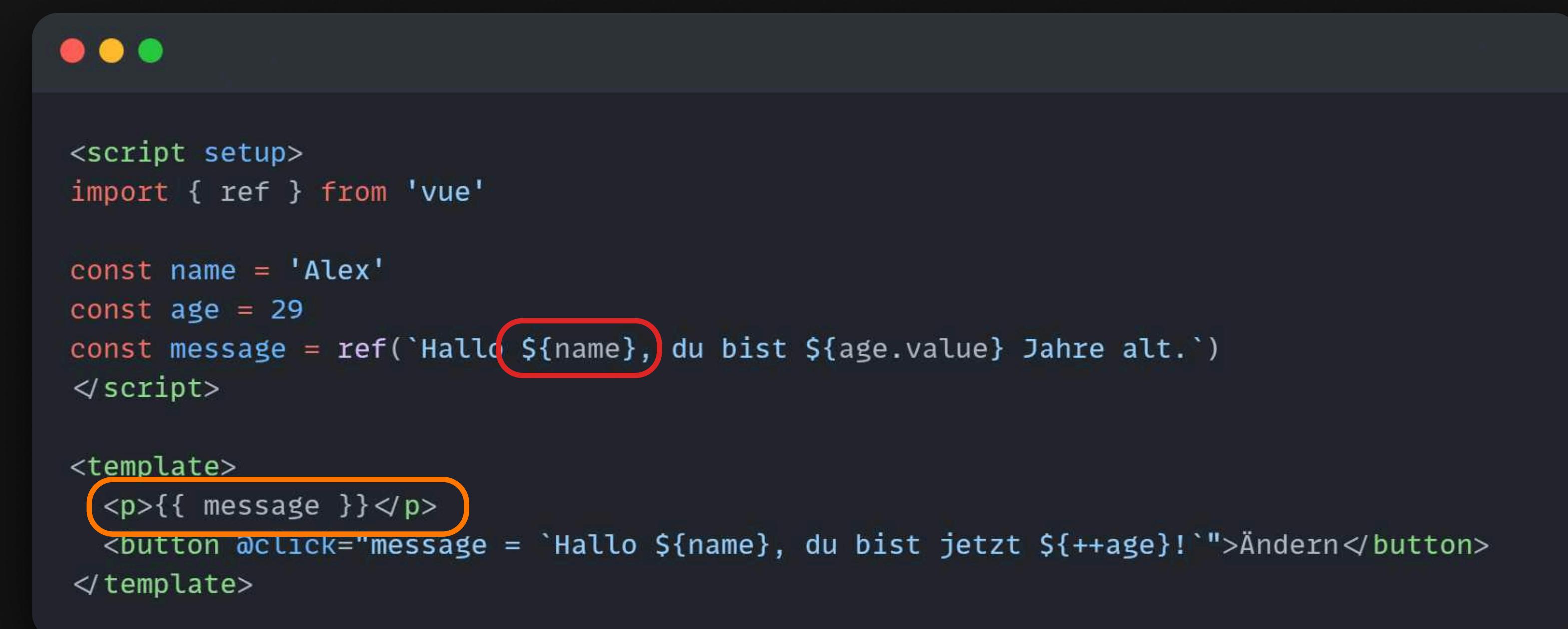
<template>
  <button @click="increment">Count is: {{ count }}</button>
</template>
```

Text Interpolation

- innerhalb der {{ ... }} kann JavaScript Code stehen
- Ähnlich wie bei JavaScript Text Interpolation

JavaScript \${ ... }

HTML {{ ... }}



```
<script setup>
import { ref } from 'vue'

const name = 'Alex'
const age = 29
const message = ref(`Hallo ${name}, du bist ${age.value} Jahre alt.`)
</script>

<template>
  <p>{{ message }}</p>
  <button @click="message = `Hallo ${name}, du bist jetzt ${++age}!`">Ändern</button>
</template>
```



Text Interpolation

Rechnungen



```
<script setup>
const price = 19.99
const quantity = 3
const isAdmin = true
const username = 'alex'
const today = new Date()
</script>

<template>
  <div>
    <p>💻 Rechnen: {{ price * quantity }} €</p>
    <p>⌚ Bedingung: {{ isAdmin ? 'Admin' : 'User' }}</p>
    <p>🔤 Funktion: {{ username.toUpperCase() }}</p>
    <p>📅 Formatierung: {{ today.toLocaleDateString() }}</p>
  </div>
</template>
```



Text Interpolation

Bedingung



```
<script setup>
const price = 19.99
const quantity = 3
const isAdmin = true
const username = 'alex'
const today = new Date()
</script>

<template>
  <div>
    <p>💻 Rechnen: {{ price * quantity }} €</p>
    <p>⌚ Bedingung: {{ isAdmin ? 'Admin' : 'User' }}</p>
    <p>🔤 Funktion: {{ username.toUpperCase() }}</p>
    <p>📅 Formatierung: {{ today.toLocaleDateString() }}</p>
  </div>
</template>
```



Text Interpolation

```
<script setup>
const price = 19.99
const quantity = 3
const isAdmin = true
const username = 'alex'
const today = new Date()
</script>

<template>
  <div>
    <p>█ Rechnen: {{ price * quantity }} €</p>
    <p>█ Bedingung: {{ isAdmin ? 'Admin' : 'User' }}</p>
    <p>█ Funktion: {{ username.toUpperCase() }}</p>
    <p>█ Formatierung: {{ today.toLocaleDateString() }}</p>
  </div>
</template>
```

Funktion





Text Interpolation

```
<script setup>
const price = 19.99
const quantity = 3
const isAdmin = true
const username = 'alex'
const today = new Date()
</script>

<template>
  <div>
    <p>寁 Rechnen: {{ price * quantity }} €</p>
    <p>⚠️ Bedingung: {{ isAdmin ? 'Admin' : 'User' }}</p>
    <p>⚠️ Funktion: {{ username.toUpperCase() }}</p>
    <p>⚠️ Formatierung: {{ today.toLocaleDateString() }}</p>
  </div>
</template>
```

Formatierung



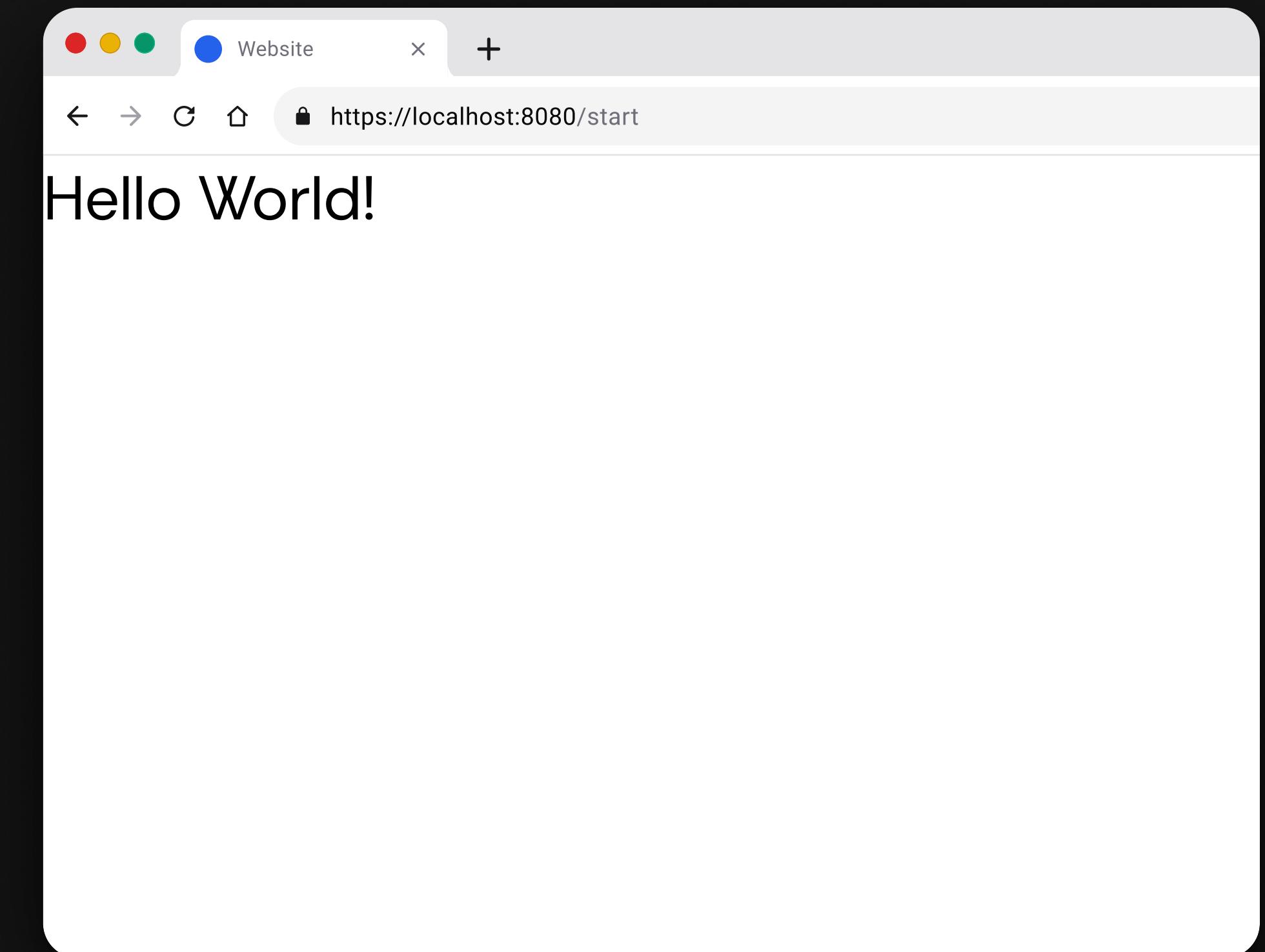


Reaktivität

```
<script setup>
import { ref } from 'vue'

const message = ref('Hello World!')
</script>

<template>
  <h1>{{ message }}</h1>
</template>
```



A screenshot of a web browser window titled "Website". The address bar shows "https://localhost:8080/start". The main content area displays the text "Hello World!".



Aufgabe 02

10 min

02 – Reactive Data

Ziel

In dieser Übung lernst du, wie du mit der Vue-Funktion `ref()` reaktive Daten erzeugst und diese dynamisch im Template anzeigenst.

Ausgangspunkt

Deine Anwendung enthält bereits:

- Ein Benutzerprofil mit `username`, `fullname` und `stats`
- Eine Sidebar und einen Profil-Bereich im Layout

Aufgabe

1. Definiere im `<script setup>`-Bereich eine neue reaktive Variable namens `bio` mit `ref()`.
2. Diese Variable soll einen kurzen Beschreibungstext (Bio) des Benutzers enthalten.
3. Binde diese `bio`-Variable im Template unterhalb der Statistik ein. Verwende dafür ein `p` Element.
4. Verwende für das neue Element die CSS-Klasse `description`.



Direktiven

- Präfix: “v-”
- DOM Manipulation mit speziellen HTML-Attributen

	Direktive	Funktion
Attribut Bindung	v-bind	dynamische Datenbindung
Conditional Rendering	v-if	erstellt HTML-Tags abhängig von einer Bedingung
List Rendering	v-for	erstellt wiederholte Elemente aus einer Liste
Event Handling	v-on	Reaktion auf Events

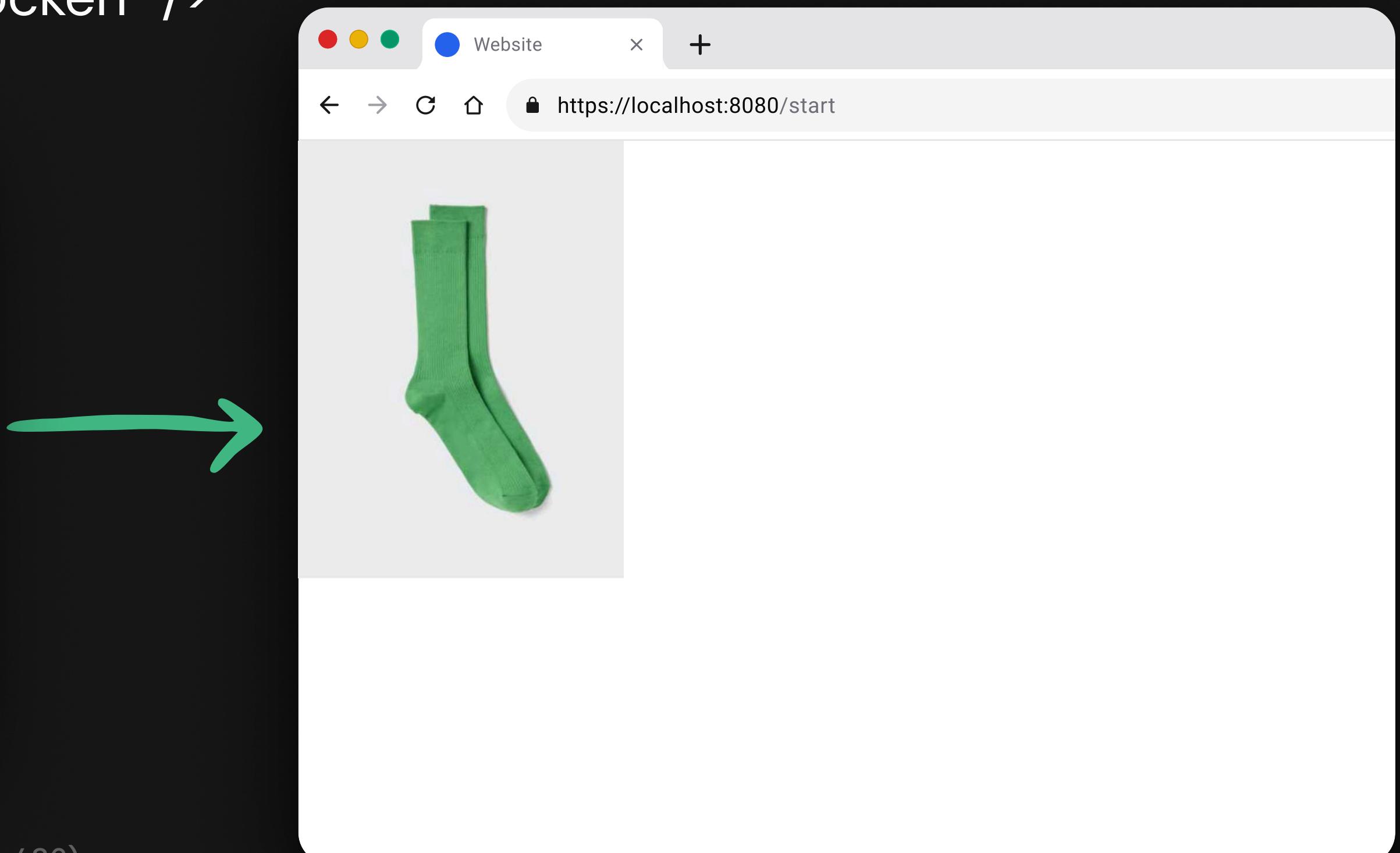
Attribut Binding

v-bind

- **Funktion:** dynamische Datenbindung an Attribute
- **Kurzschrifweise:** z.B.: ``

```
  

  <template>
    
  </template>
```



Attribut Binding

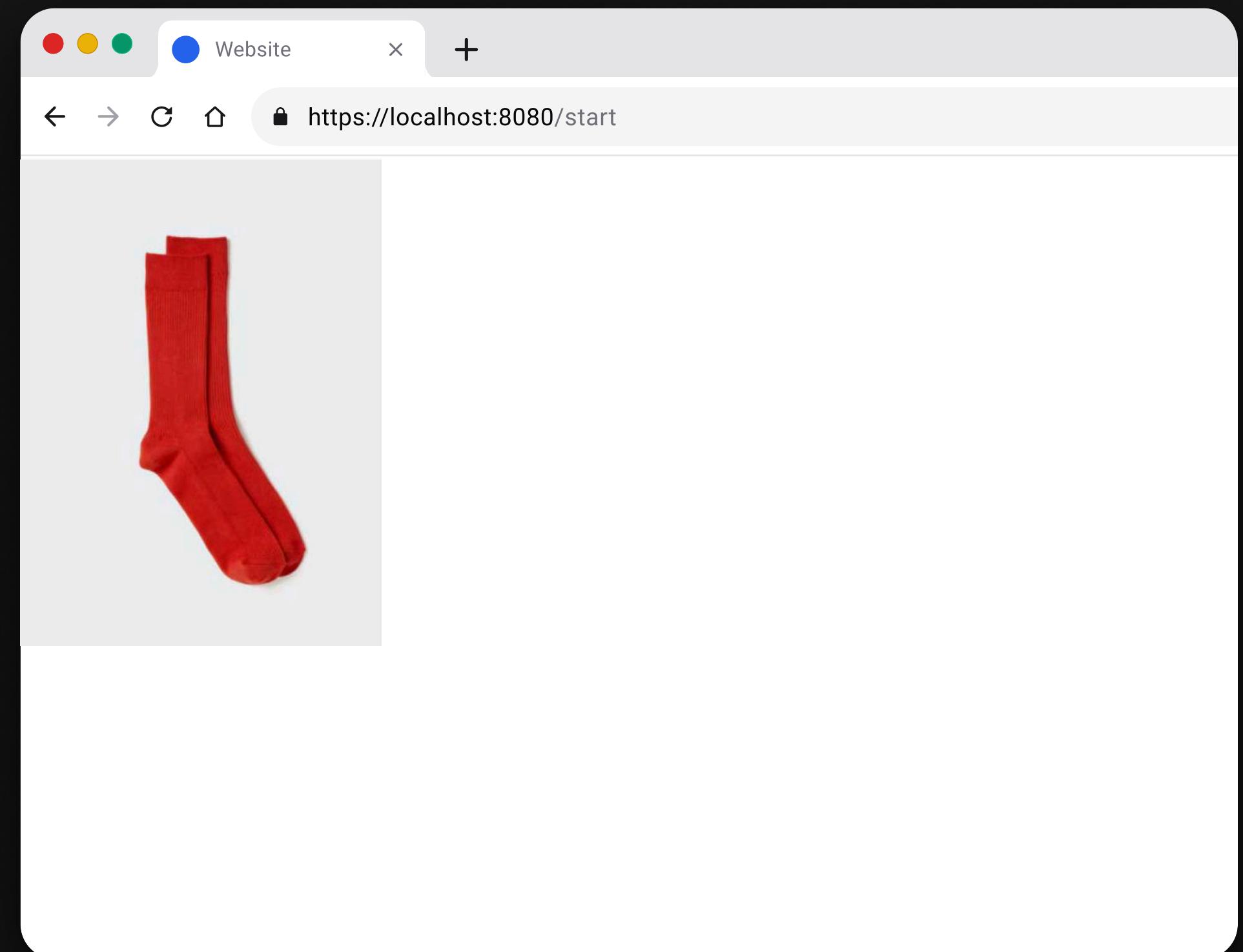
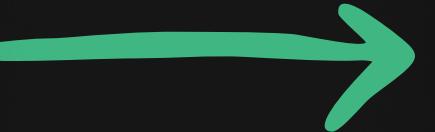
v-bind

- **Funktion:** dynamische Datenbindung an Attribute
- **Kurzschrifweise:** z.B.: ``

```
<script setup>
import { ref } from 'vue'
import socksGreenImage from './assets/images/socks_green.jpeg'
import socksRedImage from './assets/images/socks_red.jpeg'

const image = ref(socksRedImage)
</script>

<template>
  
</template>
```





Aufgabe 03

10 min

03 – Attribute-Binding

Ziel

In dieser Übung lernst du, wie man mit der Vue-Direktive `v-bind` (oder der Kurzform `:`) reaktive Daten an HTML-Attribute bindet.

Ausgangspunkt

Deine Anwendung zeigt bereits ein Profil mit Avatar, Name, Statistik und Beschreibung. Jetzt soll ein Social Link hinzugefügt werden, der auf die externe Webseite oder ein Social-Media-Profil verweist.

Aufgabe

1. Definiere im `<script setup>`-Bereich eine neue reaktive Variable `socialLink` mit `ref()`.
2. Diese Variable soll eine vollständige URL enthalten (z. B. zur offiziellen Website oder einem Instagram-Profil).
3. Binde diese `socialLink`-Variable im Template mit `v-bind` an das `href`-Attribut eines `<a>`-Tags.
4. Füge den Link **unterhalb** der Beschreibung (`.description`) ein.
5. Gib dem Link die CSS-Klasse `social-link`.

Hinweis: Verwende die Schreibweise `v-bind:href`, um das Konzept klar zu üben.



Conditional Rendering

v-if

v-else

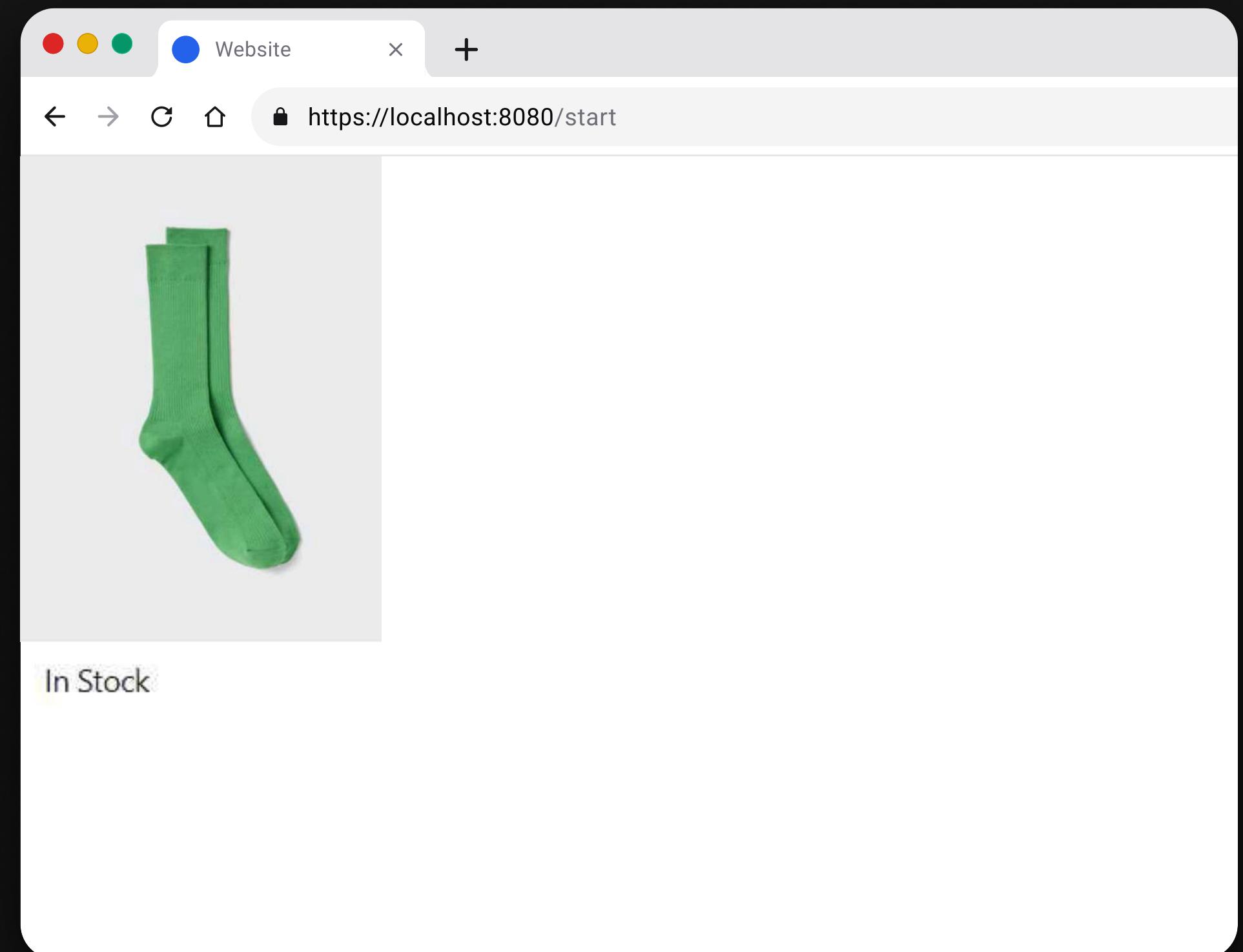
- **Funktion:** erstellt HTML-Tags abhängig von einer Bedingung
- **Weitere Direktiven:** v-if-else

```
<script setup>
import { ref } from 'vue'
import socksGreenImage from './assets/images/socks_green.jpeg'

const image = ref(socksRedImage)
const inStock = ref(true)

</script>

<template>
  
  <p v-if="inStock">In Stock</p>
  <p v-else>Out of Stock</p>
</template>
```





Conditional Rendering

v-if

v-else

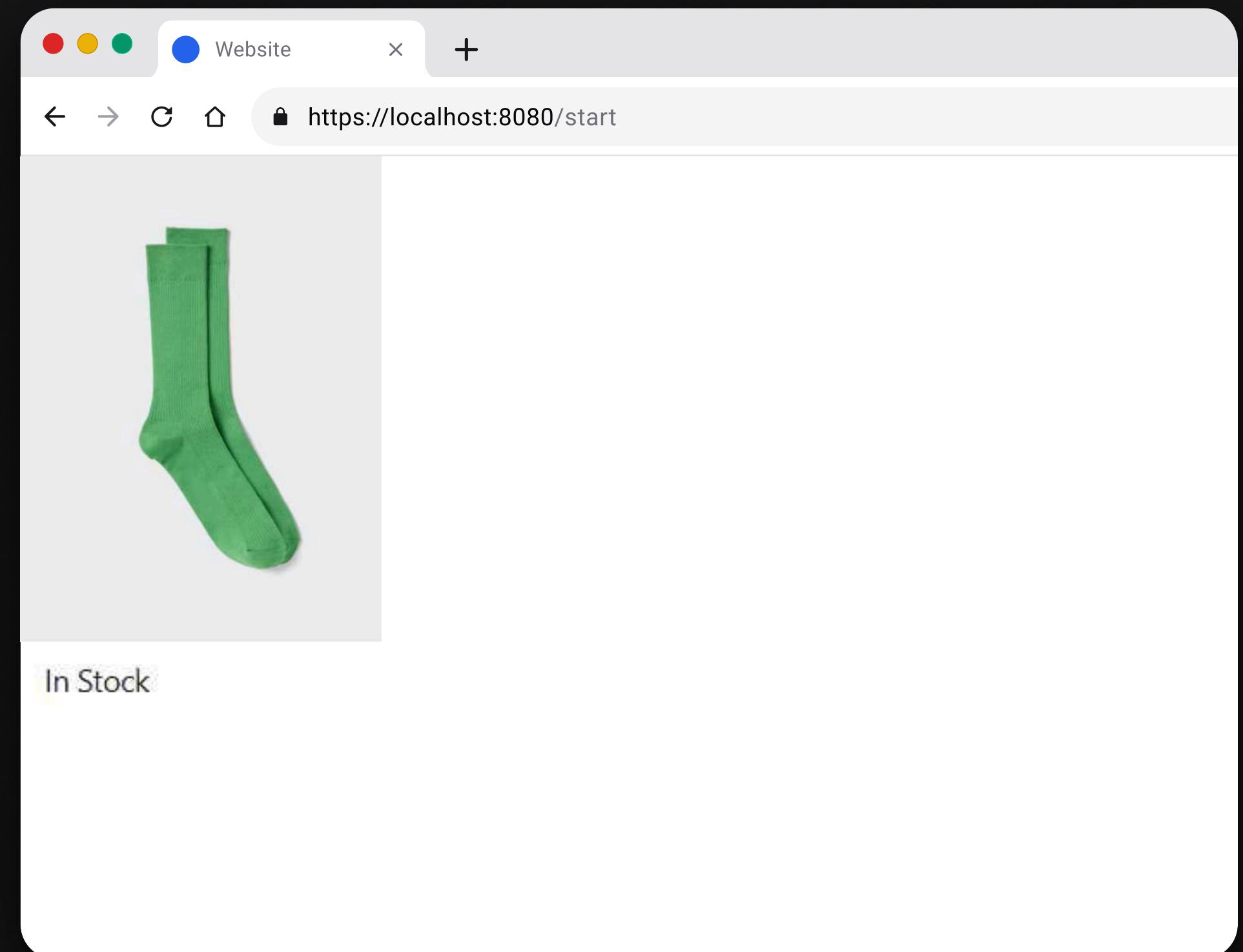
- **Funktion:** erstellt HTML-Tags abhängig von einer Bedingung
- **Weitere Direktiven:** v-if-else

```
<script setup>
import { ref } from 'vue'
import socksGreenImage from './assets/images/socks_green.jpeg'

const image = ref(socksRedImage)
const inStock = ref(true)

</script>

<template>
  
  <p v-if="inStock">In Stock</p>
  <p v-else>Out of Stock</p>
</template>
```





Conditional Rendering

v-if

v-else

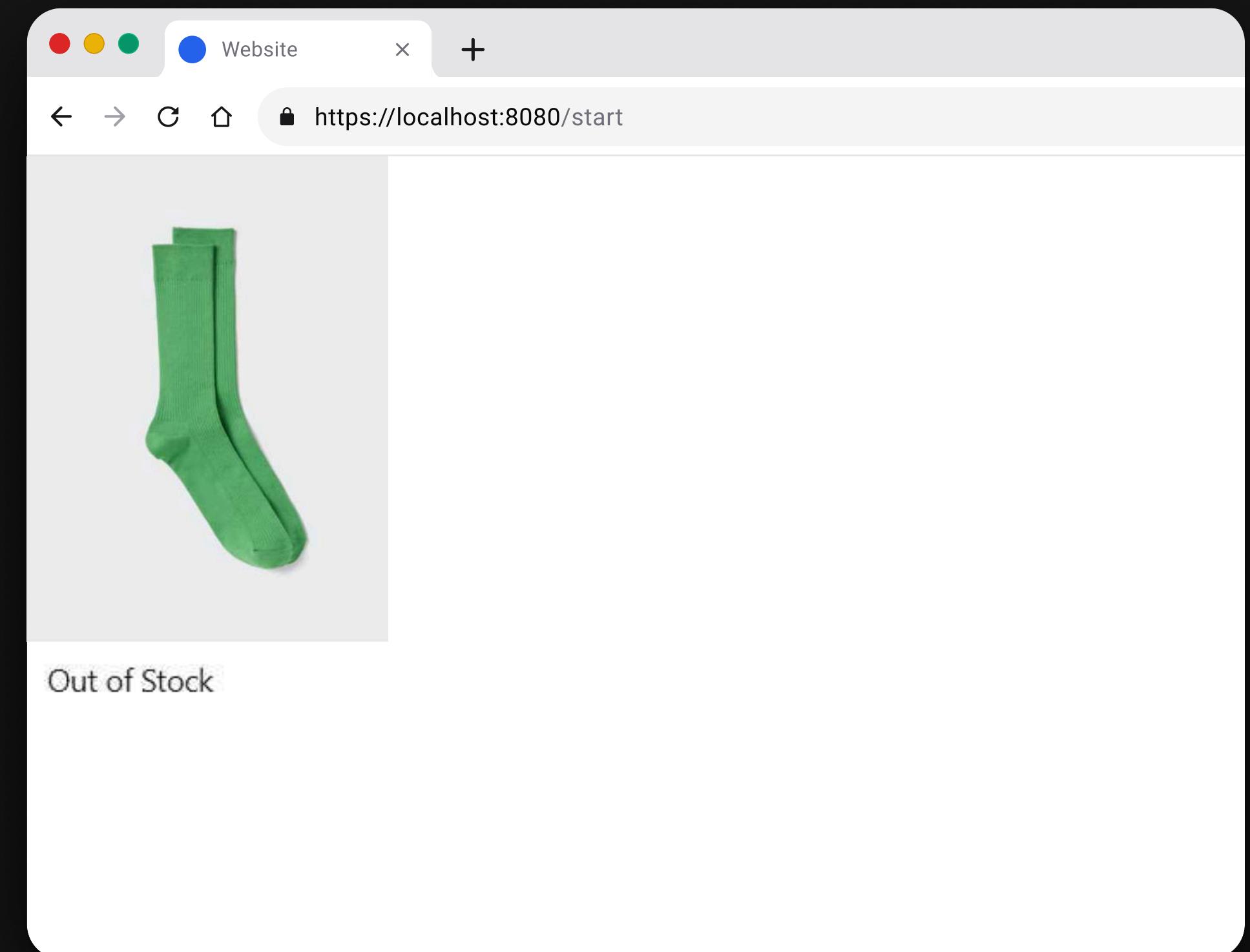
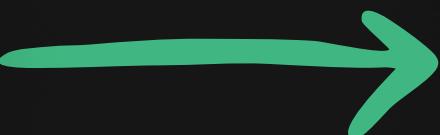
- **Funktion:** erstellt HTML-Tags abhängig von einer Bedingung
- **Weitere Direktiven:** v-if-else

```
<script setup>
import { ref } from 'vue'
import socksGreenImage from './assets/images/socks_green.jpeg'

const image = ref(socksRedImage)
const inStock = ref(false)

</script>

<template>
  
  <p v-if="inStock">In Stock</p>
  <p v-else>Out of Stock</p>
</template>
```





Conditional Rendering

v-show

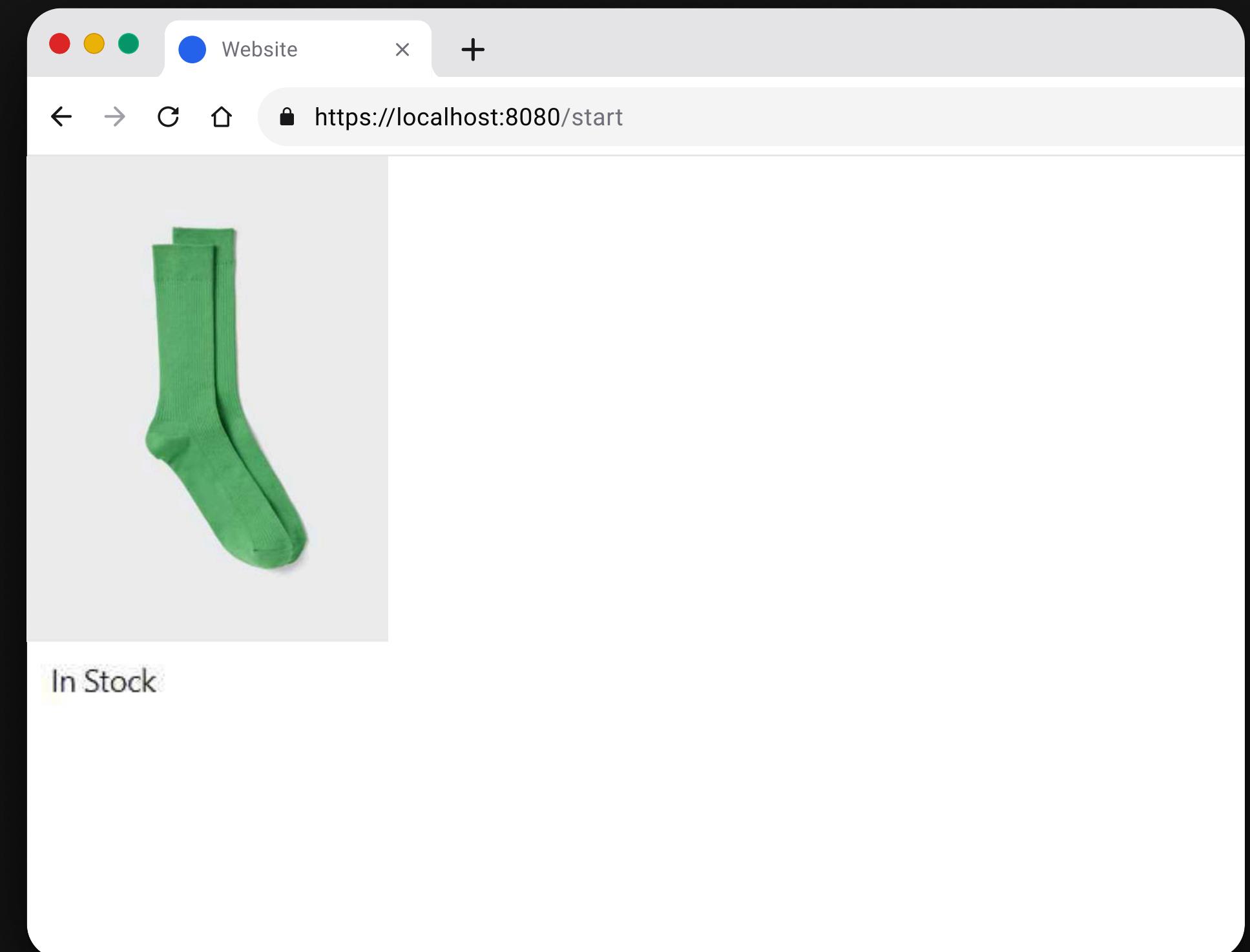
- **Funktion:** Gibt an, ob ein HTML-Element je nach Bedingung sichtbar sein soll oder nicht.

```
<script setup>
import { ref } from 'vue'
import socksGreenImage from './assets/images/socks_green.jpeg'

const image = ref(socksRedImage)
const inStock = ref(true)

</script>

<template>
  
  <p v-show="inStock">In Stock</p>
</template>
```





Conditional Rendering

v-show

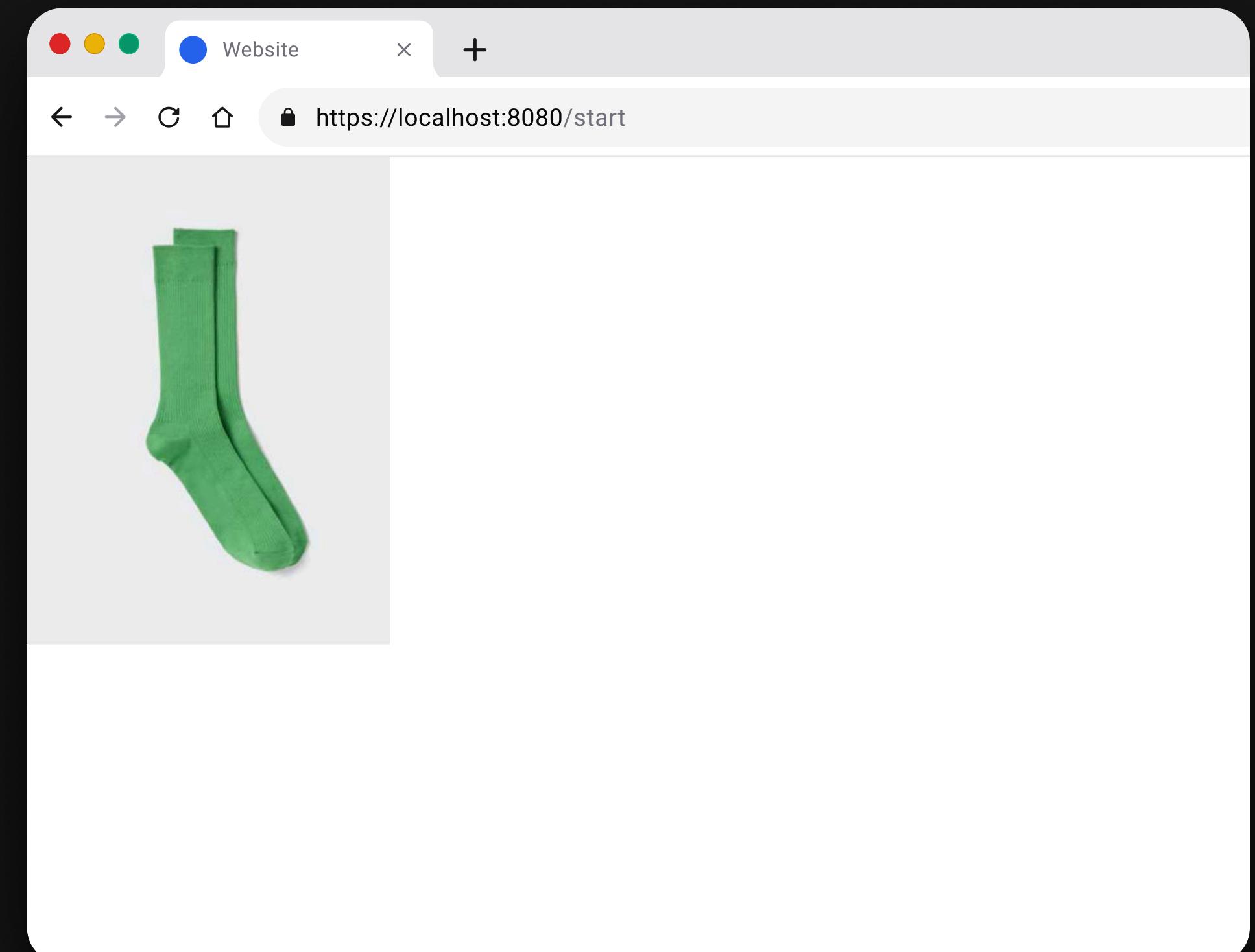
- **Funktion:** Gibt an, ob ein HTML-Element je nach Bedingung sichtbar sein soll oder nicht.

```
<script setup>
import { ref } from 'vue'
import socksGreenImage from './assets/images/socks_green.jpeg'

const image = ref(socksRedImage)
const inStock = ref(false)

</script>

<template>
  
  <p v-show="inStock">In Stock</p>
</template>
```





Aufgabe 04

10 min

04 – Conditional Rendering

Ziel

In dieser Übung lernst du, wie du Inhalte im Template abhängig von Bedingungen anzeigen kannst. Dafür verwenden wir die Vue-Direktiven `v-if`, `v-else` und `v-show`.

Ausgangspunkt

Dein Profil enthält nun bereits:

- Avatar, Benutzername und Statistik
- Social-Link
- Follow-/Unfollow-Buttons Unterhalb des Profils ist ein leerer Post-Bereich vorbereitet.

Aufgabe

1. Definiere im `<script setup>` eine neue reaktive Variable `hasPosts` mit `ref(false)`.
2. Im Template, innerhalb des `<div class="posts">`, soll Folgendes geschehen:
 - Wenn `hasPosts` `false` ist, soll der Text „Noch keine Beiträge vorhanden“ angezeigt werden.
 - Wenn `hasPosts` `true` ist, soll der Text „Beiträge werden geladen...“ erscheinen.
3. Verwende dafür die Vue-Direktive `v-if` und `v-else`.



Checkerfragen

