

Vue Workshop



Oliver Kulas · 29.11.2021 bis 03.12.2021 · DB Vertrieb GmbH

Version 1.0 (16.12.2021)

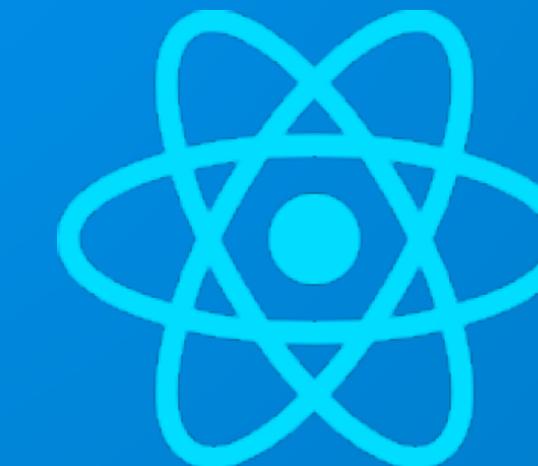
Agenda Teil 1 / 5

- **Was ist Vue?**
- Was ist eine Single Page Application?
- Vue Core Concepts
- Installation & Setup
- Vorstellung Beispielprojekt

Was ist Vue?

„The Progressive JavaScript Framework“

- Client-seitiges JavaScript Framework zur Erstellung von Single Page Applications (SPA)
- Vue wurde seit 2014 von Evan You entwickelt und veröffentlicht
- Seitdem ist die Vue-Community stark gewachsen
- Vue steht in Konkurrenz zu Angular (Google) und React (Facebook)



Warum Vue?

Die wesentlichen Vorteile

- ✓ **Einfach zu erlernen**

→ relativ einfacher Einstieg; Start nur mit HTML-, CSS- u. JS-Kenntnissen möglich

- ✓ **Vielseitig einsetzbar**

→ inkrementell anpassbares Ökosystem, das zwischen einer Bibliothek und einem Framework mit vollem Funktionsumfang skaliert

- ✓ **Performant & Leichtgewichtig**

→ 20kB min+gzip Runtime, Schneller Virtual DOM, minimaler Optimierungsaufwand

Agenda Teil 1 / 5

- Was ist Vue?
- **Was ist eine Single Page Application?**
- Vue Core Concepts
- Installation & Setup
- Vorstellung Beispielprojekt

Was ist eine Single Page Application?

Funktionsweise einer traditionellen Multipage Application (MPA)



Was ist eine Single Page Application?

Funktionsweise einer traditionellen Multipage Application (MPA)

Vorteile

Einfaches SEO Management

Native Unterstützung von Browser Navigation

Für traditionelle Webseiten geeignet
(Bereitstellung von Content-Pages vorwiegend Verlinkungen)

Sehr einfach zu verwenden durch Standard-Technologien
(HTML, CSS und gezielter Einsatz von JavaScript)

Website-Analytics per default unterstützt

Nachteile

Unschönes Nachladen bei Seitenwechsel

Nicht geeignet für umfassende Web-Applikationen bei denen es um Interaktion geht

Enge Kopplung von Frontend und Backend

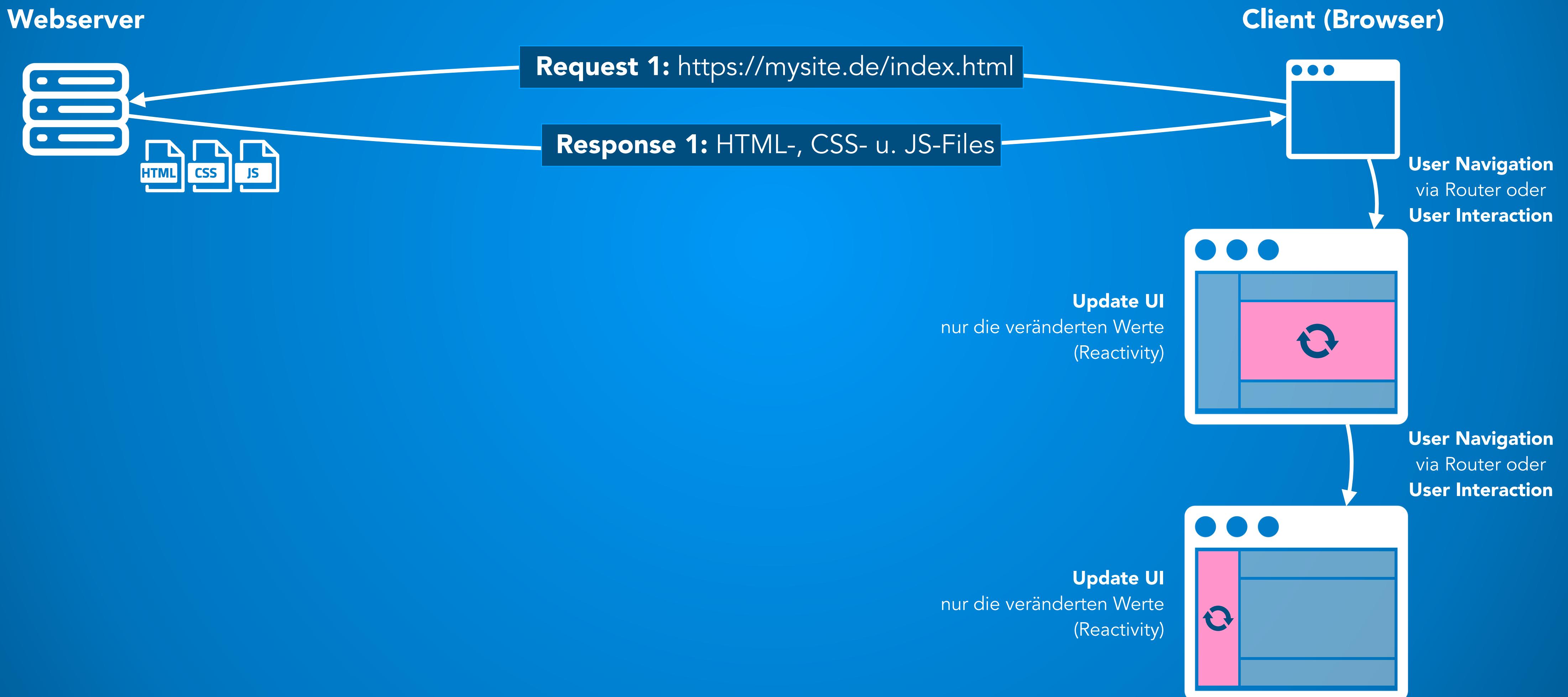
Höherer Traffic durch ständige Backend-Kommunikation

Hoher Aufwand für Offline-Fähigkeit

Langsame Webseite / Webseitenaufbau

Was ist eine Single Page Application (SPA)?

Funktionsweise einer SPA



Was ist eine Single Page Application?

Funktionsweise einer SPA

Vorteile

Sehr schnelle Webseiten
durch einmaliges Laden von Ressourcen

Keine Unschönes Nachladen Content oder Pages

Einfaches State-Management & Persistenz

Wenig Traffic

Frontend und Backend weitestgehend unabhängig

Offline-fähig

Server Side Rendering (SSR) möglich

Vereinfachte und beschleunigte Entwicklung
durch Einsatz von JS-Frameworks

Nachteile

JavaScript ist erforderlich im Browser (auf Client)

Möglicherweise aufwändiger bei Verarbeitung von komplexem
JavaScript

Schwieriges SEO-Management

Website-Analytics Unterstützung muss individuell implementiert
werden

Kein native Browser-Navigation (Routing)
→ gute Unterstützung durch Plugins

Agenda Teil 1 / 5

- Was ist Vue?
- Was ist eine Single Page Application?
- **Vue Core Concepts**
- Installation & Setup
- Vorstellung Beispielprojekt



• **Atomic Design**

- Aufbau einer Komponente
- Component Lifecycle
- Reactivity

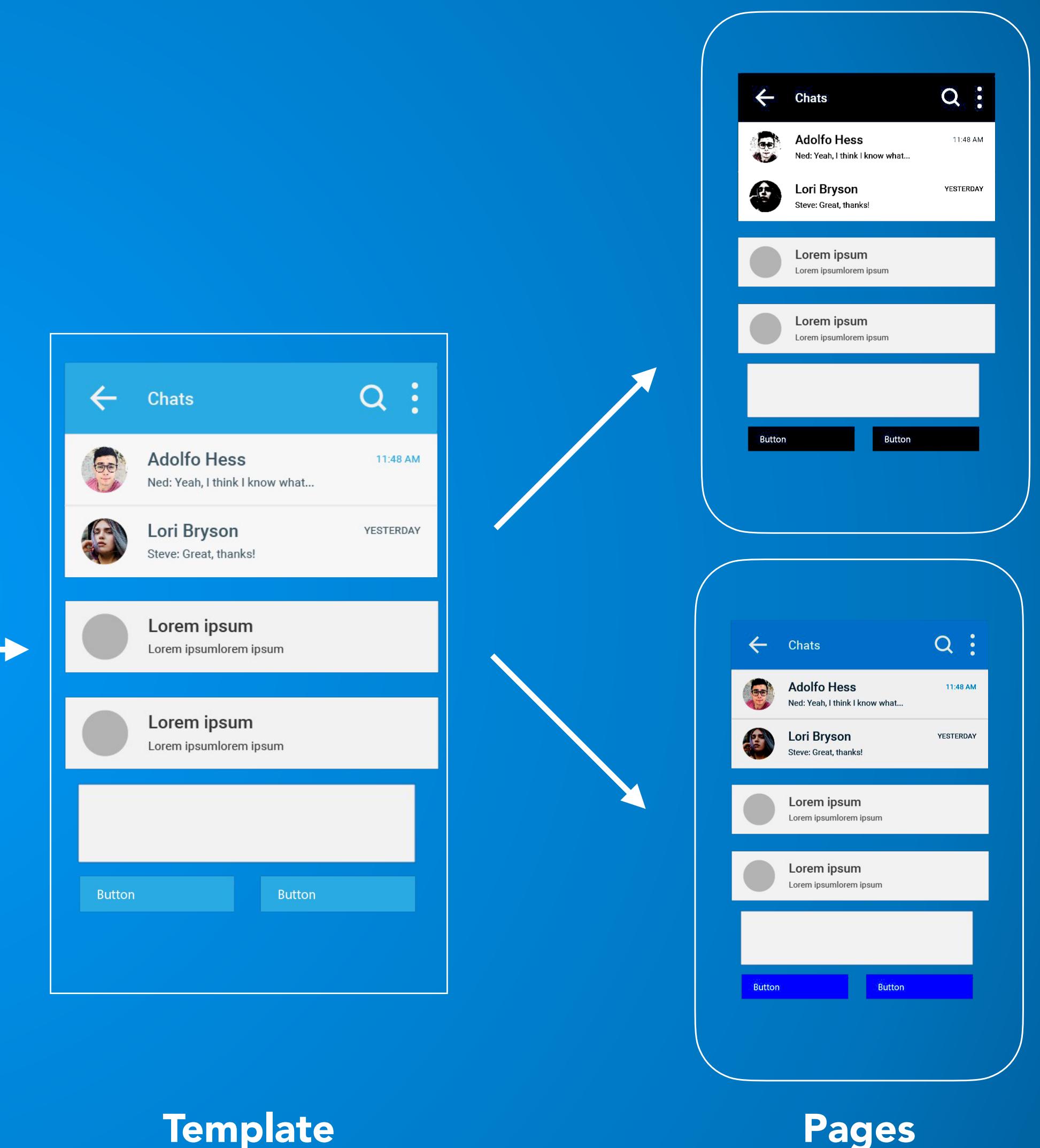
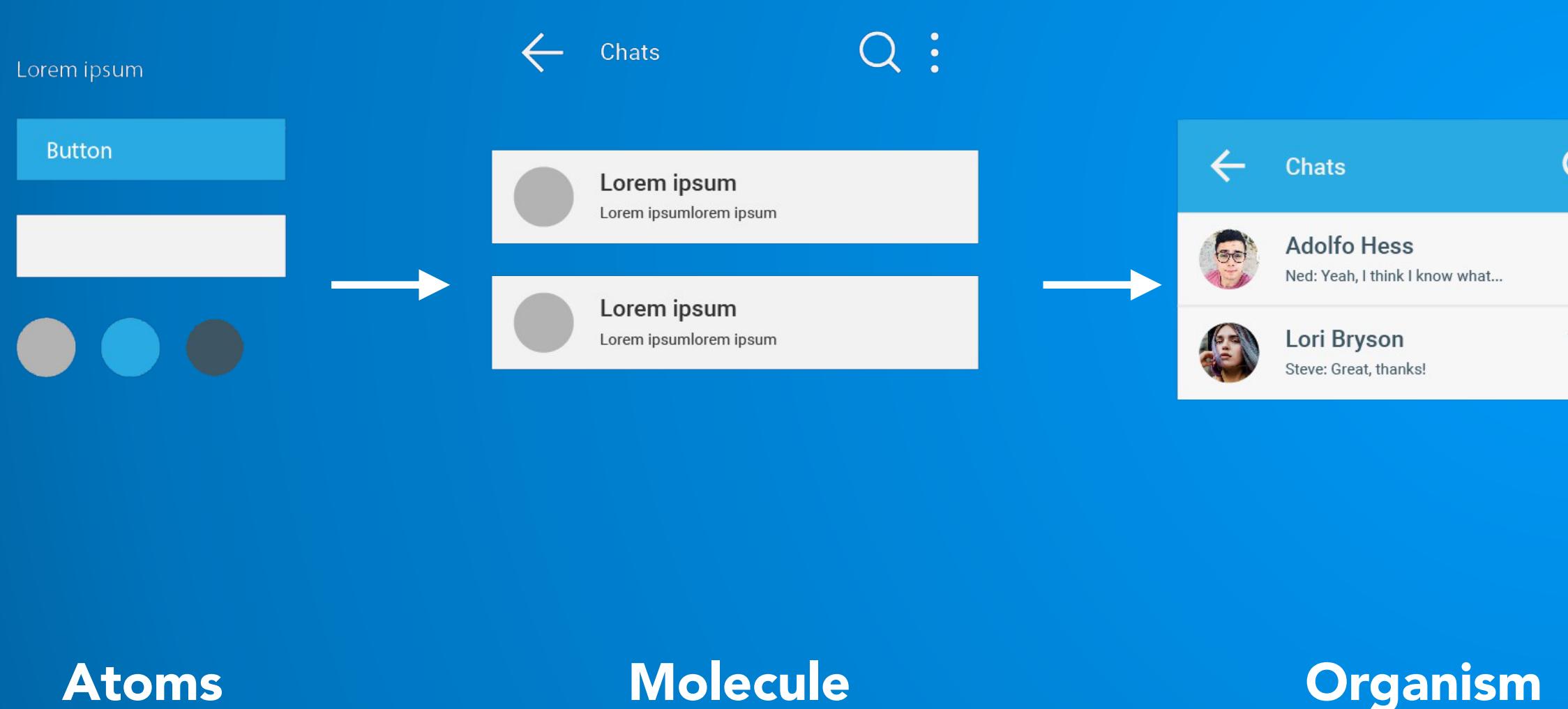
Komponenten Konzept

Atomic Design

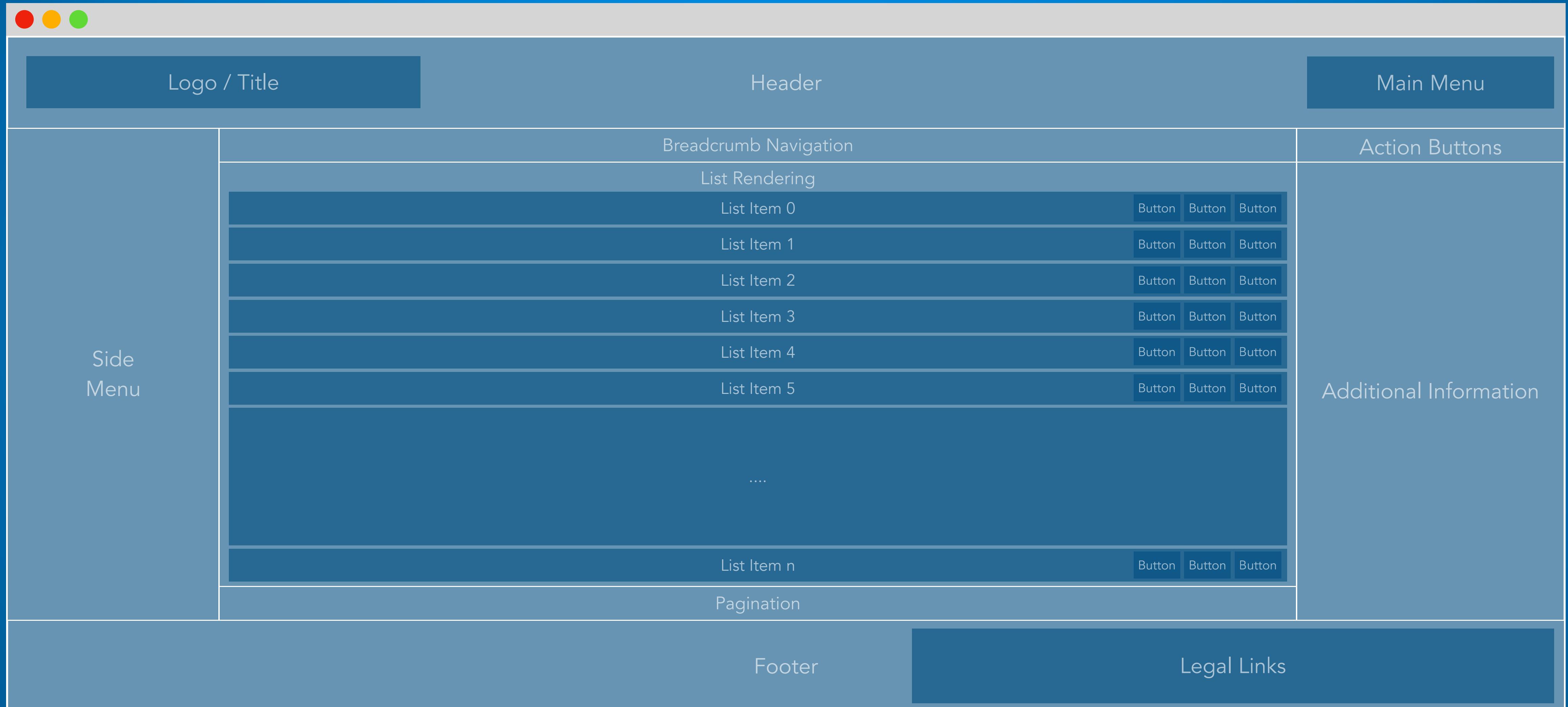
- Kein Vue spezifisches Konzept!
- Gehört nicht direkt zu Vue Core Concepts!
- **Aber:** Verwendung von Komponenten und Aufteilung einer Anwendung in kleinere Teile ist analog zum Gedanken, wie man in Vue Apps Komponenten verwendet

Komponenten Konzept

Atomic Design



Komponenten Konzept



Jedes Kästchen stellt in diesem Beispiel eine separate Komponente dar

Agenda Teil 1 / 5

- Was ist Vue?
- Was ist eine Single Page Application?
- **Vue Core Concepts**
- Installation & Setup
- Vorstellung Beispielprojekt



- Atomic Design
- **Aufbau einer Komponente**
- Component Lifecycle
- Reactivity

Aufbau einer Vue-Komponente

Vue 2.x Single File Component mit TypeScript & vue-class-component Syntax

```
<template>
  <div class="my-component">
    <!-- add html content or components here -->
  </div>
</template>

<script lang="ts">
@Component({ components: {} })
export default class MyComponent extends Vue {
  // do stuff here ...
}
</script>

<style scoped lang="scss">
.my-component {
  /* add styles here */
}
</style>
```

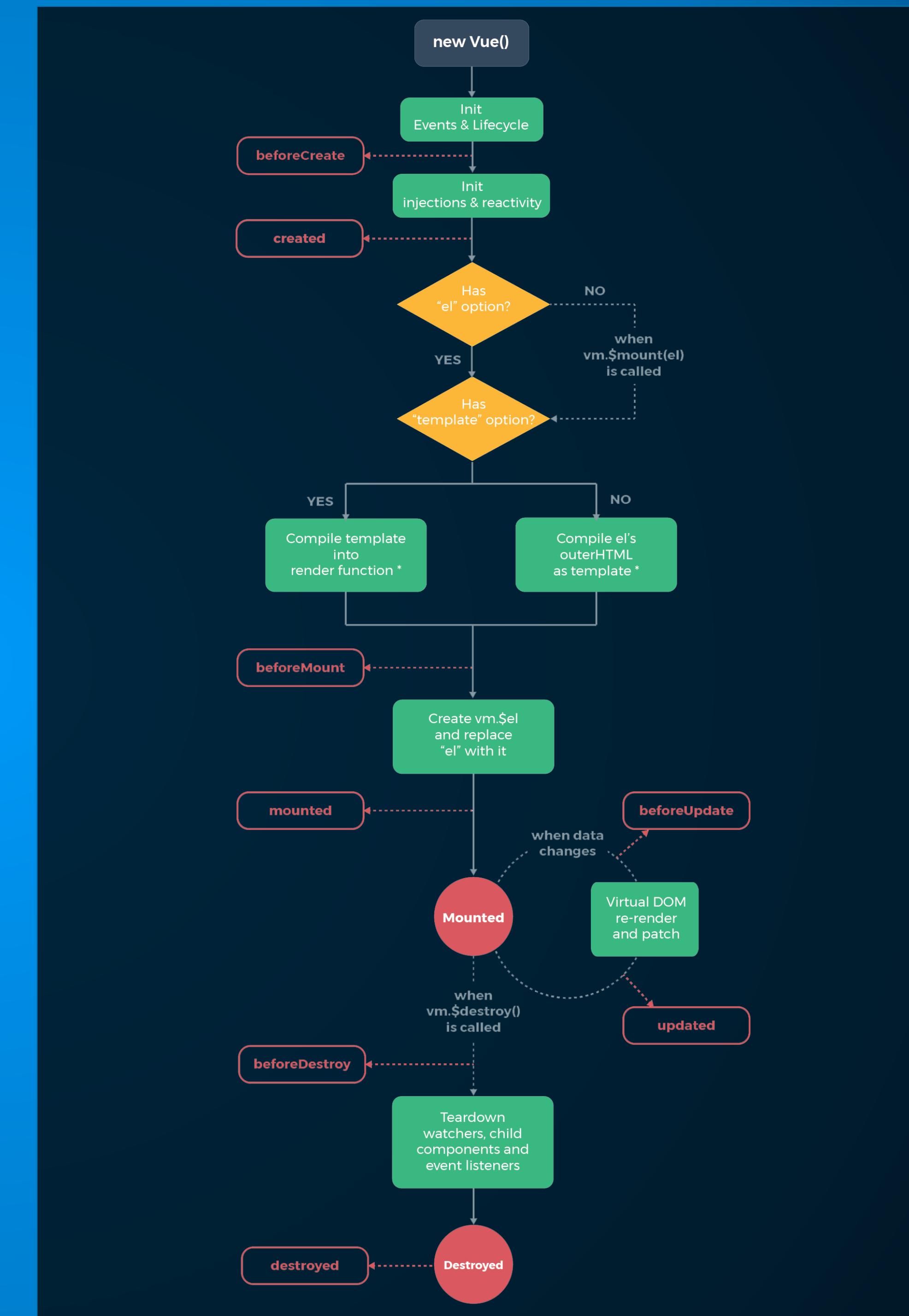
Agenda Teil 1 / 5

- Was ist Vue?
- Was ist eine Single Page Application?
- **Vue Core Concepts** →
 - Atomic Design
 - Aufbau einer Komponente
 - **Component Lifecycle**
 - Reactivity
- Installation & Setup
- Vorstellung Beispielprojekt

Component Lifecycle

Vue 2.x

- Lifecycle Diagram (rechts) dient als nützliche Referenz, die helfen kann auch wenn man die Funktionsweise nicht vollständig versteht.
- Meist genutzte Lifecycle Methoden sind:
 - `created()`
 - `mounted()`



Lifecycle Hooks

Grundsätzlich

Ein Hook entspricht einem Zeitpunkt während der Laufzeit der Anwendung.
Zu diesem Zeitpunkt „hängt man sich zwischen“ die Ausführungsschritte.

Zusammengefasst

- **Creation** → zum Start einer Komponente
- **Mounting** → wenn DOM verfügbar wird
- **Updates** → wenn reaktive Daten verändert werden
- **Destruction** → bevor Komponente zerstört wird bzw. nicht mehr existiert

Lifecycle Hooks

Vue 2.x Code-Beispiel

```
// Vue2, TypeScript and vue-class-component syntax
@Component({})
export default class MyComponent extends Vue {

    public beforeCreate(): void { ... }

    public created(): void { ... }

    public mounted(): void { ... }

    public beforeUpdate(): void { ... }

    public updated(): void { ... }

    public beforeDestroy(): void { ... }

    public destroyed(): void { ... }

}
```

Lifecycle Hooks

Weitere Lifecycle Methoden in Vue 2: Component Keep Alive

```
<!-- Template-Block -->
<keep-alive>
  <MyComponent :is="view"></MyComponent>
</keep-alive>

// Script-Block
@Component({})
export default class MyComponent extends Vue {

  // Called when a kept-alive component is activated.
  public activated(): void {
    console.log("MyComponent activated");
  }

  // Called when a kept-alive component is deactivated.
  public deactivated(): void {
    console.log("MyComponent deactivated");
  }
}
```

Mehr Information zu Component Keep Alive:

<https://v3.vuejs.org/api/built-in-components.html#keep-alive>

<https://v3.vuejs.org/guide/component-dynamic-async.html#dynamic-components-with-keep-alive>

Lifecycle Hooks

Weitere Lifecycle Methoden in Vue 2: ErrorCaptured

- Wird aufgerufen, sobald eine Child-Component einen Error wirft.
- Der Hook empfängt 3 Argumente:
`err: Error` → den Error selbst
`instance: Component` → die Instanz der Komponente
`info: string` → und Zusatzinformationen zum Error
- Hook kann `false` zurückgeben, um Weiterreichen des Fehlers an weitere Parents zu stoppen

```
// Vue2, TypeScript and vue-class-component syntax
@Component({})
export default class MyComponent extends Vue {

  public errorCaptures(): boolean {
    // handle Error from descendent component
  }
}
```

Lifecycle Hooks

Vue 3.x

| Option API | Composition API (<code>setup</code> <code>Hooks</code>) |
|------------------------------|---|
| <code>beforeCreate</code> | <i>nicht nötig*</i> |
| <code>created</code> | <i>nicht nötig*</i> |
| <code>beforeMount</code> | <code>onBeforeMount</code> |
| <code>mounted</code> | <code>onMounted</code> |
| <code>beforeUpdate</code> | <code>onBeforeUpdate</code> |
| <code>updated</code> | <code>onUpdated</code> |
| <code>beforeUnmount</code> | <code>onBeforeUnmounted</code> |
| <code>unmounted</code> | <code>onUnmounted</code> |
| <code>errorCaptured</code> | <code>onErrorCaptured</code> |
| <code>renderTracked</code> | <code>onRenderTracked</code> |
| <code>renderTriggered</code> | <code>onRenderTriggered</code> |
| <code>activated</code> | <code>onActivated</code> |
| <code>deactivated</code> | <code>onDeactivated</code> |

* `setup` -Prozess erfolgt zeitgleich zu `beforeCreate` und `created`, sodass dieser Code direkt innerhalb der `setup` -Methode geschrieben werden sollte.

Agenda Teil 1 / 5

- Was ist Vue?
- Was ist eine Single Page Application?
- **Vue Core Concepts**
- Installation & Setup
- Vorstellung Beispielprojekt

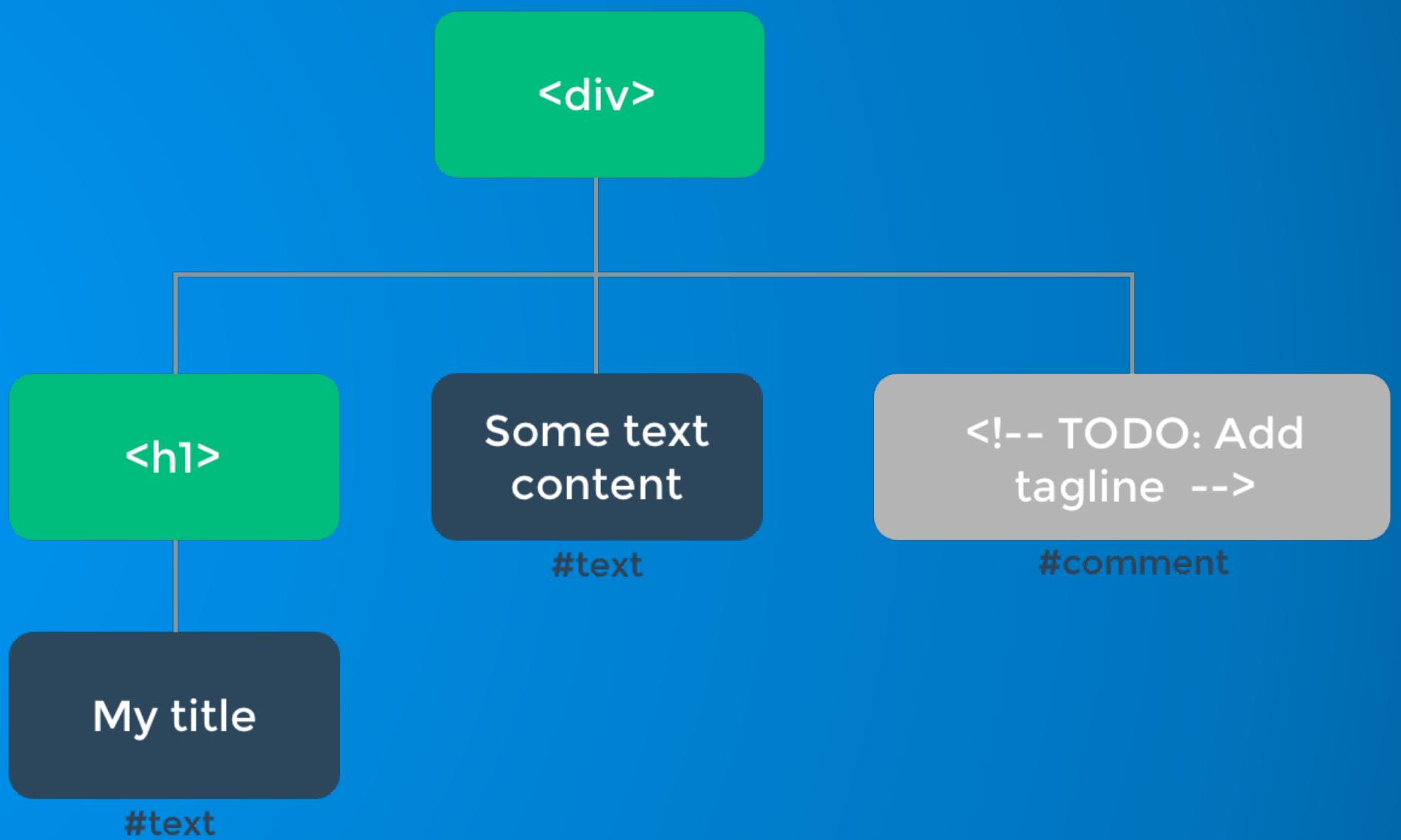


- Atomic Design
- Aufbau einer Komponente
- Component Lifecycle
- **Reactivity**

Reactivity

Nodes, Trees and the Virtual DOM

- Virtual DOM ist eine Abstraktion zwischen Vue-Instance und dem "echten" HTML-DOM
- Virtual DOM ist eine leichtgewichtige Kopie des DOM in JSON
- mit der Repräsentation des DOM als JS-Objekt, ist es möglich diverse Algorithmen anzuwenden, um Updates wesentlich effizienter als im traditionellen DOM durchzuführen
- Vue versucht den gesamten DOM so wenig wie möglich neu zu rendern bzw. zu updaten
→ aus Effizienz- und Performancen-Gründen



Reactivity

Funktionsweise Virtual DOM

- Vue kompiliert die HTML-Component-Templates in JS-Objekte
 - einfacher zu managen
 - deutlicher Performance-Gewinn

```
<template>
  <h1 class="blue">
    <span>
      Hallo DB!
    </span>
  </h1>
</template>
```

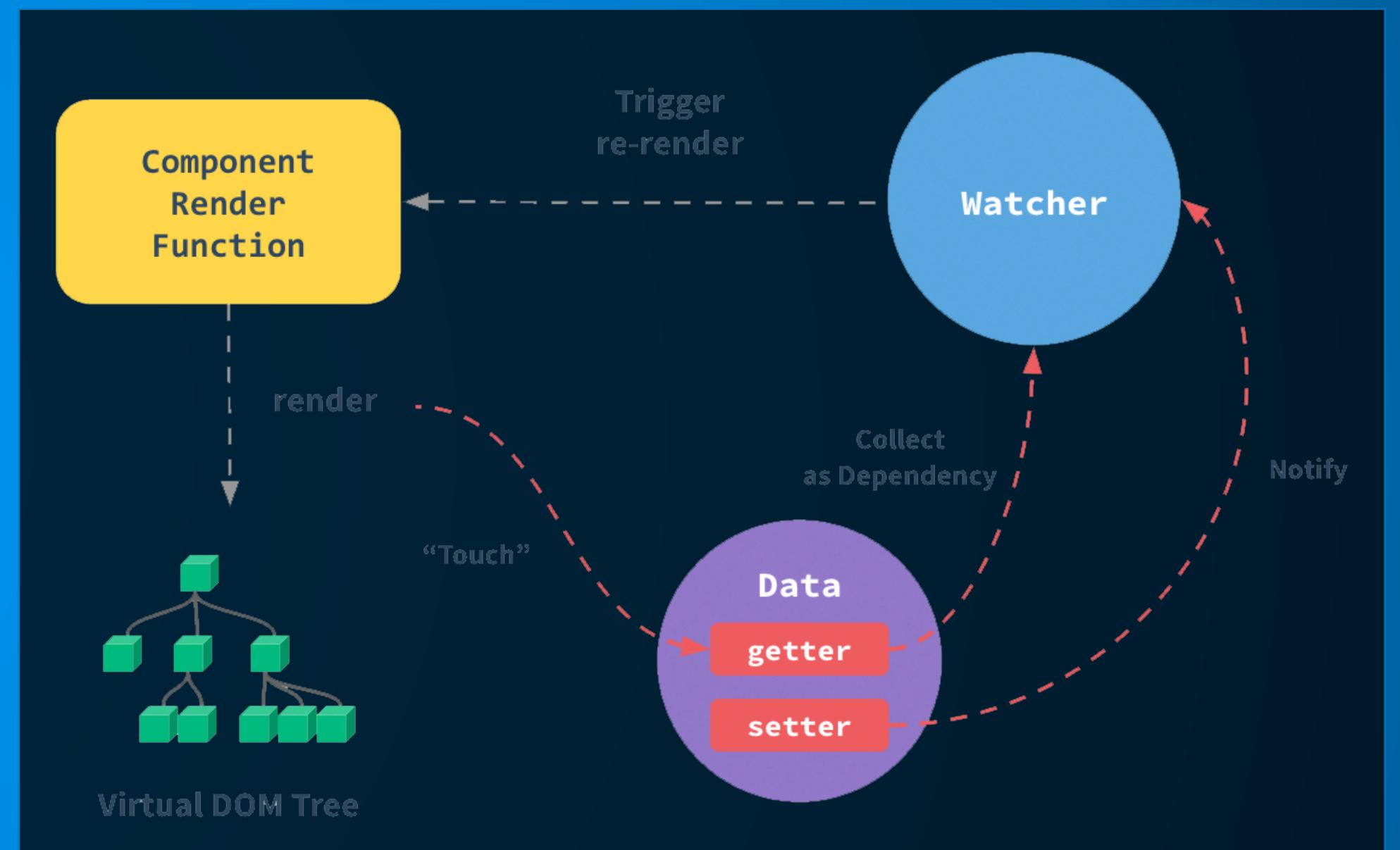


```
{
  tag: 'h1',
  attributes: {
    class: 'blue'
  },
  content: [
    {
      tag: 'span',
      attributes: {},
      content: 'Hallo DB!'
    }
  ]
}
```

Reactivity

Funktionsweise Virtual DOM

- Beispiel: HTML-Element wird hinzugefügt
- Bevor Change erfolgt bilden DOM und Virtual DOM die selben Daten ab
- Mit einem Change entsteht ein Unterschied (Diff) zwischen beiden States
- Durch den Diff des Virtual DOM wird ein **Patch** zurückgegeben, um die Änderung auf den HTML-DOM anzuwenden - **ohne ihn komplett neu zu rendern**



Reactivity

Funktionsweise Virtual DOM

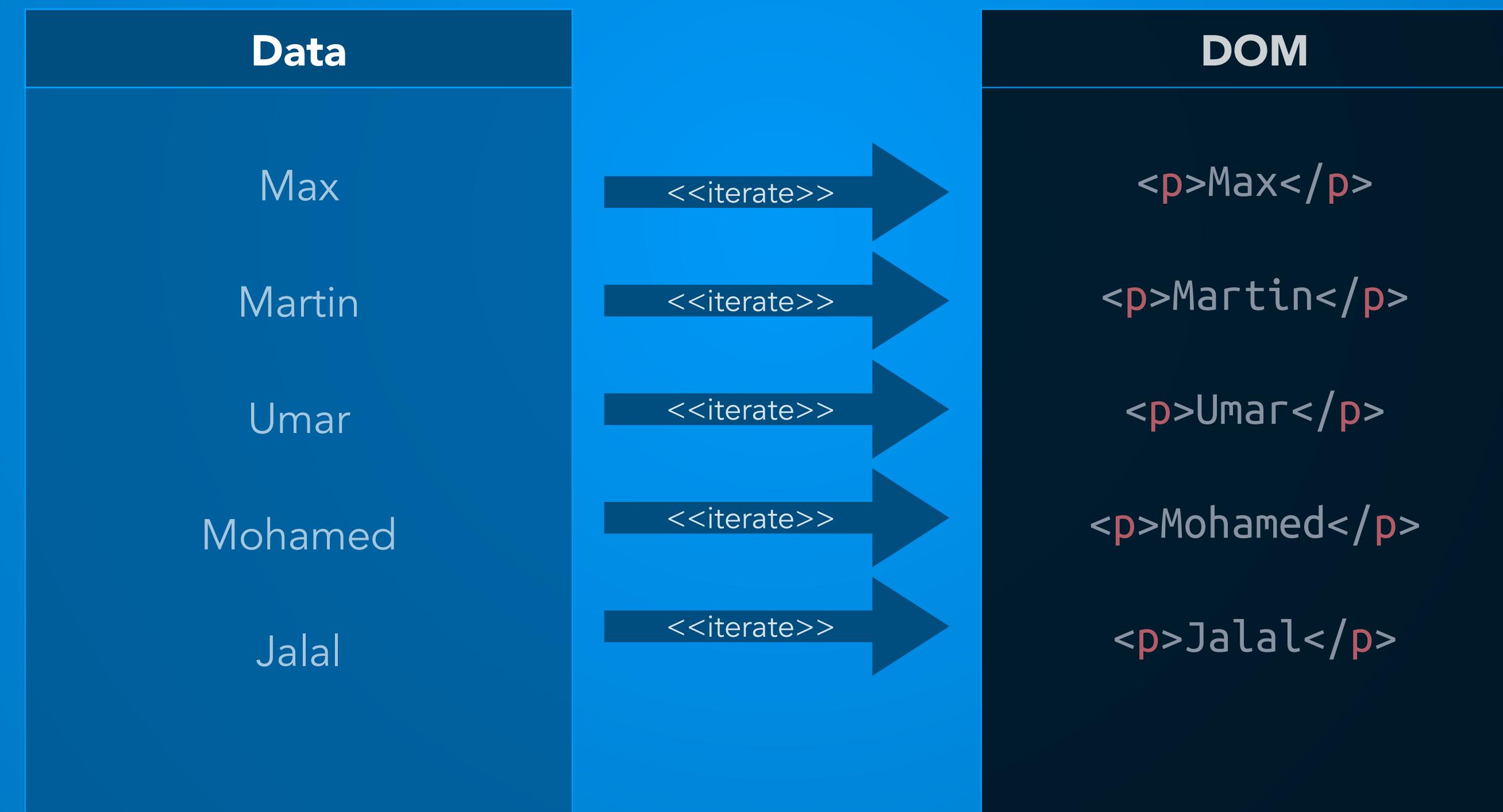
- Üblicherweise wird ein DOM-Element nach einem Change der Daten wie folgt aktualisiert:

```
document.getElementById("foo").innerHTML = "Changed Content";
```

Reactivity

Funktionsweise Virtual DOM

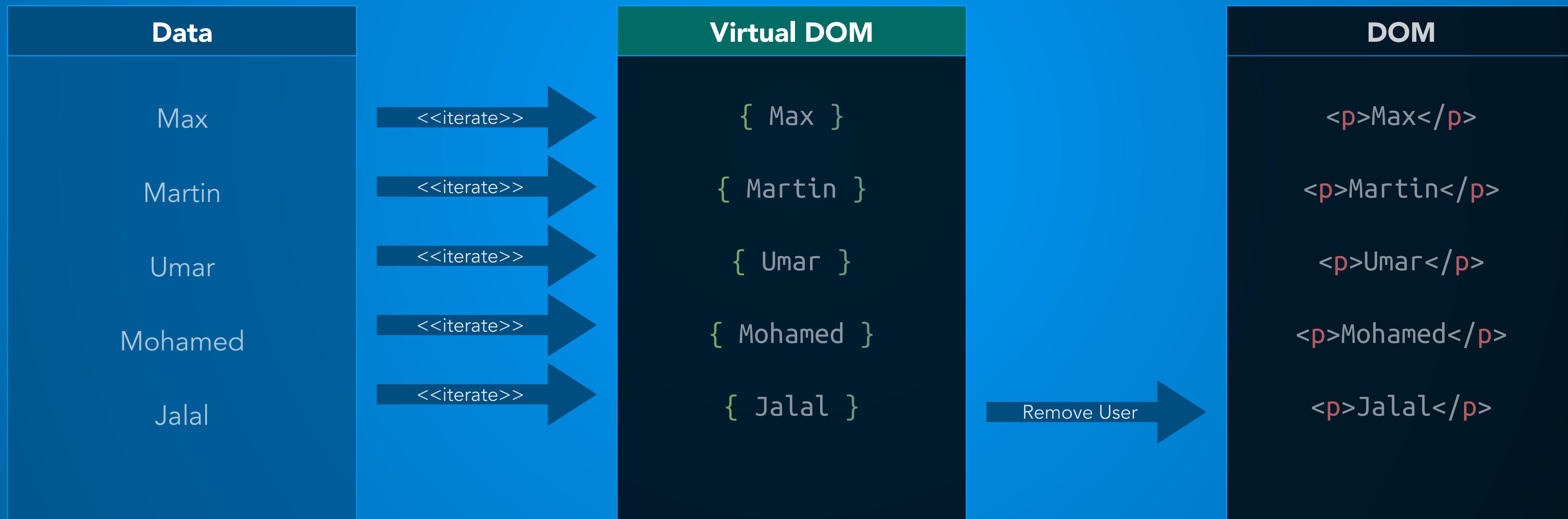
- Beispiel: Ein User soll aus einer Liste von Usern entfernt werden



Reactivity

Funktionsweise Virtual DOM

- Beispiel: Ein User soll aus einer Liste von Usern entfernt werden



Reactivity

Funktionsweise Virtual DOM - Code Deep Dive



Agenda Teil 1 / 5

- Was ist Vue?
- Was ist eine Single Page Application?
- Vue Core Concepts
- **Installation & Setup**
- Vorstellung Beispielprojekt

Vue Installation & Setup

Installation Vue CLI

- CLI → Command Line Interface
- CLI bietet Standard Tooling für Vue-Entwicklung
- z.B. automatische Installation & Setup von Vue und diversen Features & Packages

```
npm install -g @vue/cli  
# OR  
yarn global add @vue/cli
```

→ Ausführliche Anleitung unter <https://cli.vuejs.org/>

Agenda Teil 1 / 5

- Was ist Vue?
- Was ist eine Single Page Application?
- Vue Core Concepts
- Installation & Setup
- **Vorstellung Beispielprojekt**

Beispielprojekt

Einfache Notiz-App

- Notiz-App nach Vorbild von Apple Notizen (MacOS- & Web-App)
- Projekt GitLab-Repository unter:
<https://git.tech.rz.db.de/vendo/web/ausbildung/vue-workshop-2021>
- Vue2 Cheat Sheet ebenfalls im Projekt abgelegt unter:
<https://git.tech.rz.db.de/vendo/web/ausbildung/vue-workshop-2021/-/blob/develop/docs/vue2-cheatsheet.md>

Beispielprojekt

Einfache Notiz-App, ListView (Startseite)

The screenshot shows a mobile application interface for a notes application. On the left, there is a sidebar titled "Notes" containing a list of 15 notes. Each note item includes a small icon, the note title, and a timestamp. The notes are:

- asymmetric (2021-11-04T01:51:06Z)
- Virtual (2021-07-09T02:33:27Z)
- functionalities (2021-06-04T15:38:32Z)
- Ergonomic (2020-10-26T17:25:00Z)
- monitoring (2020-11-05T19:01:56Z)
- Organized (2021-01-27T02:07:30Z)
- Expanded (2021-07-31T06:53:24Z)
- Seamless (2021-02-20T07:23:06Z)
- zero tolerance (2021-02-05T17:09:24Z)
- infrastructure (2020-09-20T04:25:40Z)
- ability (2021-05-07T10:43:08Z)
- Customizable (2021-03-14T08:03:24Z)

The main content area on the right is titled "NOTEDETAIL" and displays the details of the note with the ID "5eeb3af0-1a55-438f-bd97-3bccf5753d5d". The title of this note is "functionalities". Below the title, there is a large section titled "FUNCTIONALITIES" containing placeholder text (Lorem ipsum) describing various note features.

Notes App · Vue Workshop 2021 · Oliver Kulas · DB-Vertrieb P.DLV22(1)

Beispielprojekt

Einfache Notiz-App, GridView

The screenshot shows a user interface for a 'Notes' application. At the top, a blue header bar contains the word 'Notes' on the left and three icons on the right: a menu icon, a gear icon, and a search icon. Below the header is a section titled 'NOTEGRIDLAYOUT'. The main content area displays eight notes, each in its own card:

- asymmetric** (2021-11-04T01:51:06Z)
Tags: HTML, JavaScript, TypeScript, Vue, MarkDown, Docker
Content: Integer ac neque. Duis bibendum. Morbi non quam nec dui luctus rutrum. [...]
- Virtual** (2021-07-09T02:33:27Z)
Tags: CSS, TypeScript, Vue, Nuxt, Sass, Docker
Content: Morbi a ipsum. Integer a nibh. In quis justo. Maecenas rhoncus aliquam lacus. [...]
- functionalities** (2021-06-04T15:38:32Z)
Tags: CSS, JavaScript, Nuxt, Docker
Content: Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula [...]
- Ergonomic** (2020-10-26T17:25:00Z)
Tags: HTML, JavaScript, TypeScript, Vue
Content: Nulla mollis molestie lorem. Quisque ut erat. Curabitur gravida nisi at nibh. In [...]
- monitoring** (2020-11-05T19:01:56Z)
Tags: Docker
Content: Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia [...]
- Organized** (2021-01-27T02:07:30Z)
Tags: HTML, CSS, JavaScript, Sass, Docker
Content: Curabitur gravida nisi at nibh. In hac habitasse platea dictumst. Aliquam augue [...]
- Expanded** (2021-07-31T06:53:24Z)
Tags: HTML, CSS, Sass, Docker
Content: Mauris lacinia sapien quis libero. Nullam sit amet turpis elementum ligula vehic [...]
- Seamless** (2021-02-20T07:23:06Z)
Tags: HTML
Content: Etiam faucibus cursus urna. Ut tellus. Nulla ut erat id mauris vulputate element [...]

At the bottom of the content area, there is a footer bar with the text 'Notes App · Vue Workshop 2021 · Oliver Kulas · DB-Vertrieb P.DLV22(1)'.

Beispielprojekt

Einfache Notiz-App, ListView (DarkMode)

The screenshot shows a mobile application titled "Notes" in a dark mode theme. On the left is a vertical navigation bar containing a list of notes with their titles and creation dates. On the right is the main content area, which displays a note titled "NOTEDETAIL" with a unique identifier "5eeb3af0-1a55-438f-bd97-3bccf5753d5d". Below this is a section titled "FUNCTIONALITIES" containing a large amount of placeholder text (Lorem ipsum) describing various software features.

| Notes | Date |
|-----------------|----------------------|
| asymmetric | 2021-11-04T01:51:06Z |
| Virtual | 2021-07-09T02:33:27Z |
| functionalities | 2021-06-04T15:38:32Z |
| Ergonomic | 2020-10-26T17:25:00Z |
| monitoring | 2020-11-05T19:01:56Z |
| Organized | 2021-01-27T02:07:30Z |
| Expanded | 2021-07-31T06:53:24Z |
| Seamless | 2021-02-20T07:23:06Z |
| zero tolerance | 2021-02-05T17:09:24Z |
| infrastructure | 2020-09-20T04:25:40Z |
| ability | 2021-05-07T10:43:08Z |
| Customizable | 2021-03-14T08:03:24Z |

NOTEDETAIL
5eeb3af0-1a55-438f-bd97-3bccf5753d5d,

FUNCTIONALITIES

Lore ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus. Aenean leo ligula, porttitor eu, consequat vitae, eleifend ac, enim. Aliquam lorem ante, dapibus in, viverra quis, feugiat a, tellus. Phasellus viverra nulla ut metus varius laoreet. Quisque rutrum. Aenean imperdiet. Etiam ultricies nisi vel augue. Curabitur ullamcorper ultricies nisi. Nam eget dui. Etiam rhoncus. Maecenas tempus, tellus eget condimentum rhoncus, sem quam semper libero, sit amet adipiscing sem neque sed ipsum. Nam quam nunc, blandit vel, luctus pulvinar, hendrerit id, lorem. Maecenas nec odio et ante tincidunt tempus. Donec vitae sapien ut libero venenatis faucibus. Nullam quis ante. Etiam sit amet orci eget eros faucibus tincidunt. Duis leo. Sed fringilla mauris sit amet nibh. Donec sodales sagittis magna. Sed consequat, leo eget bibendum sodales, augue velit cursus nunc, quis gravida magna mi a libero. Fusce vulputate eleifend sapien. Vestibulum purus quam, scelerisque ut, mollis sed, nonummy id, metus. Nullam accumsan lorem in dui. Cras ultricies mi eu turpis hendrerit fringilla. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; In ac dui quis mi consectetur lacinia. Nam pretium turpis et arcu. Duis arcu tortor, suscipit eget, imperdiet nec, imperdiet iaculis, ipsum. Sed aliquam ultrices mauris. Integer ante arcu, accumsan a, consectetur eget, posuere ut, mauris. Praesent adipiscing. Phasellus ullamcorper ipsum rutrum nunc. Nunc nonummy metus. Vestibulum volutpat pretium libero. Cras id dui. Aenean ut eros et nisl sagittis vestibulum. Nullam nulla eros, ultrices sit amet, nonummy id, imperdiet feugiat, pede. Sed lectus. Donec mollis hendrerit risus. Phasellus nec sem in justo pellentesque facilisis. Etiam imperdiet imperdiet orci. Nunc nec neque. Phasellus leo dolor, tempus non, auctor et, hendrerit quis, nisi. Curabitur ligula sapien, tincidunt non, euismod vitae, posuere imperdiet, leo. Maecenas malesuada. Praesent congue erat at massa. Sed cursus turpis vitae tortor. Donec posuere vulputate arcu. Phasellus accumsan cursus velit. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Sed aliquam, nisi quis porttitor congue, elit erat euismod orci, ac placerat dolor lectus quis orci. Phasellus consectetur vestibulum elit. Aenean tellus metus, bibendum sed, posuere ac, mattis non, nunc. Vestibulum fringilla pede sit amet augue. In turpis. Pellentesque posuere. Praesent turpis. Aenean posuere, tortor sed cursus feugiat, nunc augue blandit nunc, eu sollicitudin urna dolor sagittis lacus. Donec elit libero, sodales nec, volutpat a, suscipit non, turpis. Nullam sagittis. Suspendisse pulvinar, augue ac venenatis condimentum, sem libero volutpat nibh, nec pellentesque velit pede quis nunc. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Fusce id purus. Ut varius tincidunt libero. Phasellus dolor. Maecenas vestibulum mollis diam. Pellentesque ut neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. In dui magna, posuere eget, vestibulum et, tempor auctor, justo. In ac felis quis tortor malesuada pretium. Pellentesque auctor neque nec urna. Proin sapien ipsum, porta a, auctor quis, euismod ut, mi. Aenean viverra rhoncus pede. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Ut non enim eleifend felis pretium feugiat. Vivamus quis mi. Phasellus a est. Phasellus magna. In hac habitasse platea dictumst. Curabitur at lacus ac velit ornare lobortis. Curabitur a felis in nunc fringilla tristique. Morbi mattis ullamcorper velit. Phasellus gravida semper nisi. Nullam vel sem. Pellentesque libero tortor, tincidunt et, tincidunt eget, semper nec, quam. Sed hendrerit. Morbi ac felis. Nunc egestas, augue at pellentesque laoreet, felis eros vehicula leo, at malesuada velit leo quis pede. Donec interdum, metus et hendrerit aliquet, dolor diam sagittis ligula, eget egestas

Beispielprojekt

Einfache Notiz-App, GridView (DarkMode)

The screenshot shows a dark-themed application window titled "Notes". At the top right are icons for a menu and a moon symbol. Below the title is a header "NOTEGRIDLAYOUT". The main area displays a 2x4 grid of notes, each in its own card:

- asymmetric** (2021-11-04T01:51:06Z)
Tags: HTML, JavaScript, TypeScript, Vue, MarkDown, Docker
Content: Integer ac neque. Duis bibendum. Morbi non quam nec dui luctus rutrum. [...]
- Virtual** (2021-07-09T02:33:27Z)
Tags: CSS, TypeScript, Vue, Nuxt, Sass, Docker
Content: Morbi a ipsum. Integer a nibh. In quis justo. Maecenas rhoncus aliquam lacus. [...]
- functionalities** (2021-06-04T15:38:32Z)
Tags: CSS, JavaScript, Nuxt, Docker
Content: Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula [...]
- Ergonomic** (2020-10-26T17:25:00Z)
Tags: HTML, JavaScript, TypeScript, Vue
Content: Nulla mollis molestie lorem. Quisque ut erat. Curabitur gravida nisi at nibh. In [...]
- monitoring** (2020-11-05T19:01:56Z)
Tags: Docker
Content: Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia [...]
- Organized** (2021-01-27T02:07:30Z)
Tags: HTML, CSS, JavaScript, Sass, Docker
Content: Curabitur gravida nisi at nibh. In hac habitasse platea dictumst. Aliquam augue [...]
- Expanded** (2021-07-31T06:53:24Z)
Tags: HTML, CSS, Sass, Docker
Content: Mauris lacinia sapien quis libero. Nullam sit amet turpis elementum ligula vehic [...]
- Seamless** (2021-02-20T07:23:06Z)
Tags: HTML
Content: Etiam faucibus cursus urna. Ut tellus. Nulla ut erat id mauris vulputate element [...]

At the bottom center is the footer: Notes App · Vue Workshop 2021 · Oliver Kulas · DB-Vertrieb P.DLV22(1)

Agenda Teil 2 / 5

- **Vue Basics**
- CSS-Präprozessoren
- Vue Router
- State-Management mit Vuex
- Vue Cookies



- **Template Syntax & Expressions**
- Data Binding
- Component Props
- Computed Properties
- Watchers
- Event Handling
- List Rendering
- Komponenten Kommunikation
- Class- & Style-Binding
- Debugging

Vue Basics

Template Syntax

- Interpolation → Mustache Notation

```
<span>Message: {{ msg }}</span>
```

- HTML-Attribute → Mustache hier nicht erlaubt

```
<div v-bind:id="dynamicId"></div>
```

- HTML-Direktiven, z.B. v-if oder v-show

```
<p v-if="isTextShown">Now you see me</p>
```

→ alle Vue-Directives: <https://v3.vuejs.org/api/directives.html>

Vue Basics

JavaScript Expressions - Beispiele

```
<div id="app">

<p>I have a {{ product }}</p>

<p>{{ product + "s" }}</p>

<p>{{ isWorking ? "YES" : "NO" }}</p>

<p>{{ product.getSalePrice() }}</p>

</div>
```

Weitere Informationen & Quellen:

Template Syntax: <https://vuejs.org/v2/guide/syntax.html>

JS Expressions: <https://vuejs.org/v2/guide/syntax.html#Using-JavaScript-Expressions>



Agenda Teil 2 / 5

- **Vue Basics**
- CSS-Präprozessoren
- Vue Router
- State-Management mit Vuex
- Vue Cookies



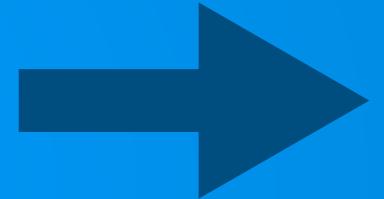
- Template Syntax & Expressions
- **Data Binding**
- Component Props
- Computed Properties
- Watchers
- Event Handling
- List Rendering
- Komponenten Kommunikation
- Class- & Style-Binding
- Debugging

Vue Basics

Two-Way-Data Binding

- HTML-Directive `v-model` ermöglicht eine sehr einfache und automatische bidirektionale Datenbindung

```
<input v-model="message" placeholder="edit me" />
<p>Message is: {{ message }}</p>
```



Meine Testeingabe 123
Message is: Meine Testeingabe 123

Ausgabe

- Die HTML-Elemente Radio, Checkbox und Select wird mittels `v-model` sind für gewöhnlich

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/forms.html> · <https://vuejs.org/v2/guide/forms.html#Value-Bindings>

Vue 3: <https://v3.vuejs.org/guide/forms.html#value-bindings>

Vue Basics

Two-Way-Data Binding

- v-bind ermöglicht den Eingabewert an beliebige, Nicht-String-Werte zu binden:

```
<MyComponent v-bind:myPropName="myValue" />
```

- Oder als Shorthand → Keyword v-bind kann dabei weg gelassen werden:

```
<MyComponent :myPropName="myValue" />
```

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/forms.html> · <https://vuejs.org/v2/guide/forms.html#Value-Bindings> · <https://vuejs.org/v2/guide/syntax.html#Shorthands>

Vue 3: <https://v3.vuejs.org/guide/forms.html#value-bindings>

Agenda Teil 2 / 5

- **Vue Basics**
- CSS-Präprozessoren
- Vue Router
- State-Management mit Vuex
- Vue Cookies



- Template Syntax & Expressions
- Data Binding
- **Component Props**
- Computed Properties
- Watchers
- Event Handling
- List Rendering
- Komponenten Kommunikation
- Class- & Style-Binding
- Debugging

Vue Basics

Component Props

- Damit werden Daten von außen (Parents) in die Komponenten übergeben
- Sind stets reaktiv
- Können primitive oder komplexe Datentypen sein
- Props dürfen nur von Parent-Component gesetzt oder verändert
 - versucht man Props innerhalb der Komponente, zu der das Prop gehört zu ändern, gibt Vue einen Fehler in der Browser-Console aus

 **[Vue warn]:** Avoid mutating a prop directly since the value will be overwritten whenever the parent component re-renders. Instead, use a data or computed property based on the prop's value. Prop being mutated: "notes"

Vue Basics

Component Props - Beispiele mit TypeScript & Prop-Decorator mit Options

```
// Props with Primitives
@Prop({required: true, type: Number})
public myNumber!: number;

@Prop({required: true, type: Boolean})
public myBoolean!: boolean;

@Prop({required: true, type: String})
public myString!: string;

// Props with complex data types
@Prop({required: true, type: Array as PropType<Note[]>})
public notes!: Note[];

@Prop({required: true, type: Object as PropType<NoteGridItem>})
public noteGridItem!: NoteGridItem;
```

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/api/#props>

Vue 3: <https://v3.vuejs.org/guide/component-props.html#props> · <https://v3.vuejs.org/guide/component-props.html#prop-types>



Vue Basics

Setzen von Props zweier Custom Components aus der Parent Component Home.vue (Beispiel innerhalb einer Single File Component)

```
<template>
  <div class="home">
    <NoteListLayout v-if="$store.getters.isListLayout"
      :notes="notes"></NoteListLayout>
    <NoteGridLayout v-else
      :notes="notes"></NoteGridLayout>
  </div>
</template>

<script lang="ts">
import ...

@Component({ components: { NoteListLayout, NoteGridLayout } })
export default class Home extends Vue {

  public notes: Note[] = [];

  ...
}

</script>
```

In diesem Beispiel wird das Notes-Array mittels Binding (`v-bind`) als Prop in die beiden Komponenten übergeben.
Die Komponenten `NoteListLayout` bzw. `NoteGridLayout` stellen dazu jeweils das Prop "notes" bereit.

Beachte: Dabei ist es egal, ob die Daten primitiv oder komplex sind.
Hier handelt es sich hier um einen komplexen Datentyp in Form eines Object-Array.

Agenda Teil 2 / 5

- **Vue Basics**
- CSS-Präprozessoren
- Vue Router
- State-Management mit Vuex
- Vue Cookies



- Template Syntax & Expressions
- Data Binding
- Component Props
- **Computed Properties**
- Watchers
- Event Handling
- List Rendering
- Komponenten Kommunikation
- Class- & Style-Binding
- Debugging

Vue Basics

Computed Properties

- Während die In-Template Syntax für einfache Operationen gedacht ist, werden Computed Properties für komplexere Logik verwendet
 - Komplexe Expressions im Template blähen den Code auf und sind schwer lesbar
 - Code ist durch Computed Properties besser wartbar
- Sind ebenfalls stets reaktiv
- sind im Kontext von TypeScript und unter Verwendung des vue-class-component Package get & set-Methoden (sogenannte Getter / Setter)
- Werden syntaktisch nicht wie Methoden, sondern wie Variablen behandelt

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/computed.html>

Vue 3: <https://v3.vuejs.org/guide/computed.html#computed-properties>

Vue Basics

Computed Properties - Beispiele

```
<!-- Template: easy to read, simple to use -->
<div v-if="isValidNote">

// Script with complex expression
public get isValidNote(): boolean {
  return !(this.note.id && this.note.content && this.note.title && this.note.createdAt);
}
```

```
<!-- Template: easy to read, simple to use -->
<NoteList :noteListItems="noteListItems"></NoteList>

// Script with complex logic
public get noteListItems(): NoteListItem[] {
  return this.notes.map((note: Note) => {
    return {
      id: note.id,
      title: note.title,
      lastModified: note.lastModified,
    };
  });
}
```

Agenda Teil 2 / 5

- **Vue Basics**
- CSS-Präprozessoren
- Vue Router
- State-Management mit Vuex
- Vue Cookies



- Template Syntax & Expressions
- Data Binding
- Component Props
- Computed Properties
- **Watchers**
- Event Handling
- List Rendering
- Komponenten Kommunikation
- Class- & Style-Binding
- Debugging

Vue Basics

Watchers

- Watcher sind eine direkte Möglichkeit, um auf veränderte Daten zu reagieren, anders als reaktive Getter/Setter, die "magisch" UI-Updates über die Vue-Reactivity auslösen
- Funktionsweise ähnlich eines Listeners
- Sobald sich Daten verändert, wird die mit dem `@Watch`-Decorator versehene Methode ausgeführt
- Vorteil: Man kann nicht reaktive Werte und Objekte überwachen, z.B. `$route`, `window`

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/computed.html#Computed-vs-Watched-Property>

Vue 3: <https://v3.vuejs.org/guide/computed.html#watchers>



Vue Basics

Watchers - Beispiele

```
// watch on route change
@Watch('$route')
public async getMyParamFromRoute(): Promise<void> {
  const myParam = this.$route.params.myParam;
  // do something ...
}
```

```
public myValue: string = '';
...

// watch for changes of a non-reactive class-member; watcher with options
@Watch('myValue', { immediate: true })
public async getNoteIdFromRoute(): Promise<void> {
  // do something on change of this.myValue
}
```

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/computed.html#Computed-vs-Watched-Property>

Vue 3: <https://v3.vuejs.org/guide/computed.html#watchers>



Agenda Teil 2 / 5

- **Vue Basics**
- CSS-Präprozessoren
- Vue Router
- State-Management mit Vuex
- Vue Cookies



- Template Syntax & Expressions
- Data Binding
- Component Props
- Computed Properties
- Watchers
- **Event Handling**
- List Rendering
- Komponenten Kommunikation
- Class- & Style-Binding
- Debugging

Vue Basics

Event Handling

- Mit Vue-Direktive `v-on` kann auf Events beliebiger DOM-Elemente gehört werden, um auf User Interaktionen (Klick, Touch, Keypress usw.) zu reagieren
- Events werden in Vue auch dazu verwendet, um Daten einer Child-Komponente an die Parent-Komponente zu übergeben
→ als Gegenstück zu Props, über die Daten in die Child-Komponente kommen
- Beispiel: Klick-Event mit `v-on:click` → Shorthand: `@click`
- Beachte: Vue-Event-Namen verzichten auf den Prefix `on`
- Innerhalb von Vue sollten alle Events über Vue-Direktive verwendet werden, pure JavaScript-Listener, wie `onclick` oder `onchange`, verhalten sich u.U. unerwartet

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/events.html>

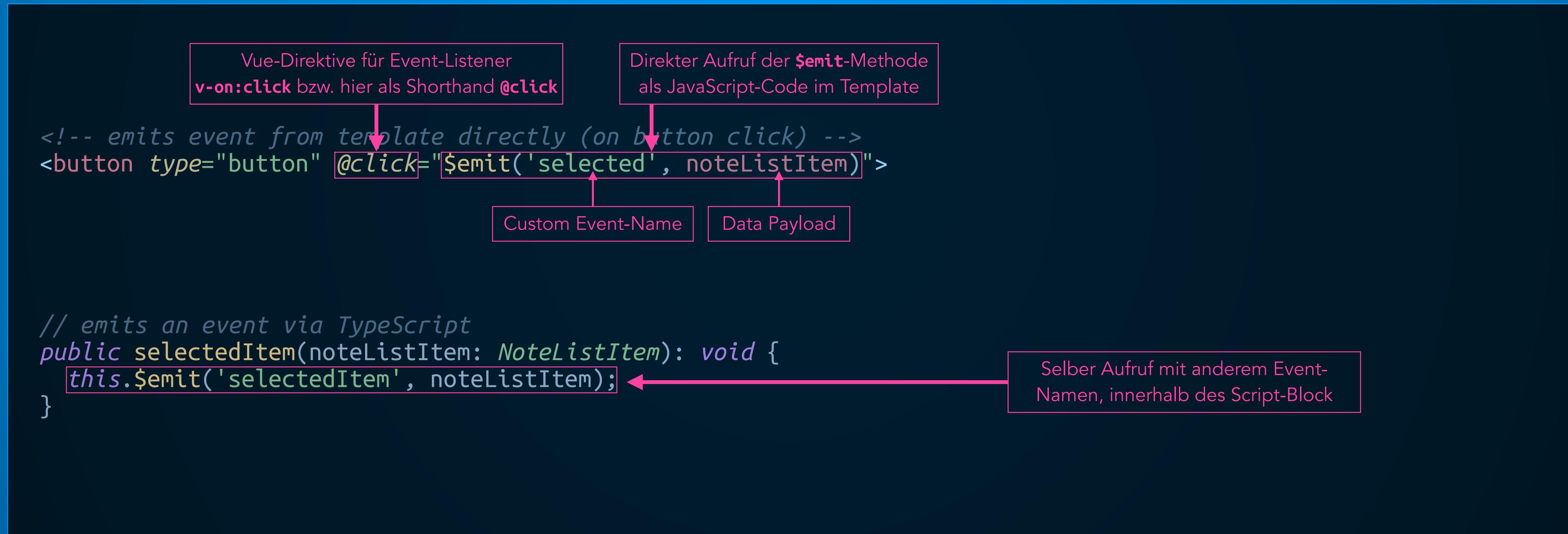
Vue 3: <https://v3.vuejs.org/guide/events.html>



Vue Basics

Event Handling - Beispiel

- Komponente sendet Event mittels \$emit an Parent (aus Template- oder Script-Block)



Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/events.html>

Vue 3: <https://v3.vuejs.org/guide/events.html>

Vue Basics

Event Handling - Event Modifier

- In JavaScript ist es oft notwendig `event.preventDefault()` oder `event.stopPropagation()` aufzurufen
 - Vue vereinfacht den Einsatz durch Event Modifier, die man einfach mit den Event-Namen verkettet:
 - `.stop`
 - `.prevent`
 - `.capture`
 - `.self`
 - `.once`
 - `.passive`
- 
- Beispiel
- ```
<form v-on:submit.prevent="onSubmit"></form>
```

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/events.html> · Vue2 Event Modifier: <https://vuejs.org/v2/guide/events.html#Event-Modifiers>

Vue 3: <https://v3.vuejs.org/guide/events.html> · Vue3 Event Modifier: <https://v3.vuejs.org/guide/events.html#event-modifiers>

# Vue Basics

## Event Handling - Key & Mouse Modifier

- Um auf Keyboard- bzw. Mouse-Events zu reagieren

```
<!-- Handle Keyboard Events -->
<input @keyup.enter="submit" />

<input @keyup.page-down="onPageDown" />

<!-- Handle Mouse Events -->
<button @click.ctrl="onClick">A</button>

<button @click.ctrl.exact="onCtrlClick">A</button>

<button @click.exact="onClick">A</button>
```

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/events.html> · Vue2 Key-Modifiers: <https://vuejs.org/v2/guide/events.html#Key-Modifiers>  
Vue 3: <https://v3.vuejs.org/guide/events.html> · Vue3 Key-Modifiers: <https://v3.vuejs.org/guide/events.html#key-modifiers>

# Agenda Teil 2 / 5

- **Vue Basics**
- CSS-Präprozessoren
- Vue Router
- State-Management mit Vuex
- Vue Cookies



- Template Syntax & Expressions
- Data Binding
- Component Props
- Computed Properties
- Watchers
- Event Handling
- **List Rendering**
- Komponenten Kommunikation
- Class- & Style-Binding
- Debugging

# Vue Basics

## List Rendering mit v-for

- Analog zu einer Schleife im Script gibt es mit der Direktive `v-for` die Möglichkeit List-Items innerhalb des Template zu rendern
- Die Syntax entspricht der Form: `item in items`
  - `items` ist dabei die Datenquelle (z.B. ein Array)
  - `item` ist ein Alias für das Array-Element, über das gerade iteriert wird
- Das Array kann primitive Werte oder komplexe Werte enthalten
  - Primitives String-Array: [„John“, „Jane“, „Jessica“, „Jamal“]
  - Komplexes Object-Array: [{ myValue: 23; }, { myValue: 42; }]
- Optional kann ein 2. Parameter für den `index` des aktuellen Elements genutzt werden

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/list.html>

Vue 3: <https://v3.vuejs.org/guide/list.html#mapping-an-array-to-elements-with-v-for>

# Vue Basics

## List Rendering mit v-for - Beispiel 1 mit primitivem String-Array (1/4)

```
<template>
 <div class="list">

 <li v-for="(listItem, index) in listItems" :key="index">
 {{ listItem }}

 </div>
</template>

<script lang="ts">
import { Component, Vue } from "vue-property-decorator";

@Component({})
export default class List extends Vue {

 public listItems = ["Alpha", "Beta", "Gamma", "Delta", "Epsilon", "Zeta", "Lambda"];

}
</script>
```

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/list.html>

Vue 3: <https://v3.vuejs.org/guide/list.html#mapping-an-array-to-elements-with-v-for>



# Vue Basics

## List Rendering mit v-for - Beispiel 1 mit primitivem String-Array (2/4)

```
<template>
 <div class="list">

 <li v-for="(listItem, index) in listItems" :key="index">
 {{ listItem }}

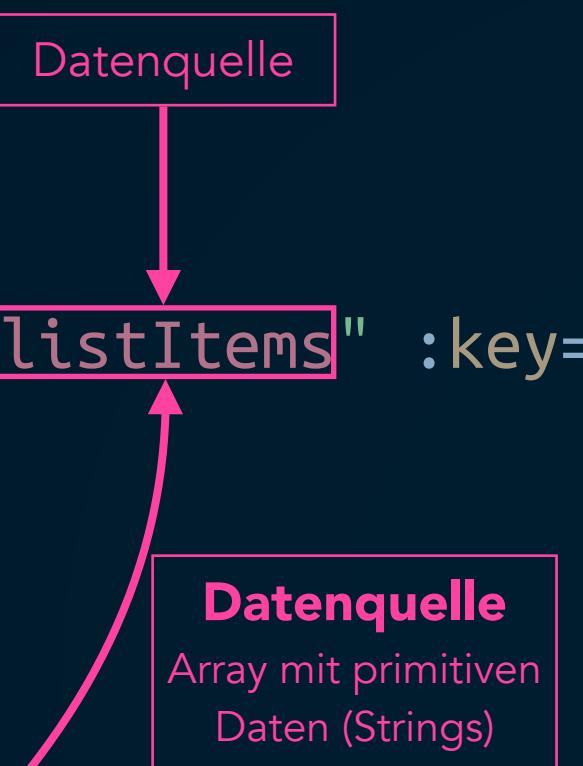
 </div>
</template>

<script lang="ts">
import { Component, Vue } from "vue-property-decorator";

@Component({})
export default class List extends Vue {

 public listItems = ["Alpha", "Beta", "Gamma", "Delta", "Epsilon", "Zeta", "Lambda"];
}

</script>
```



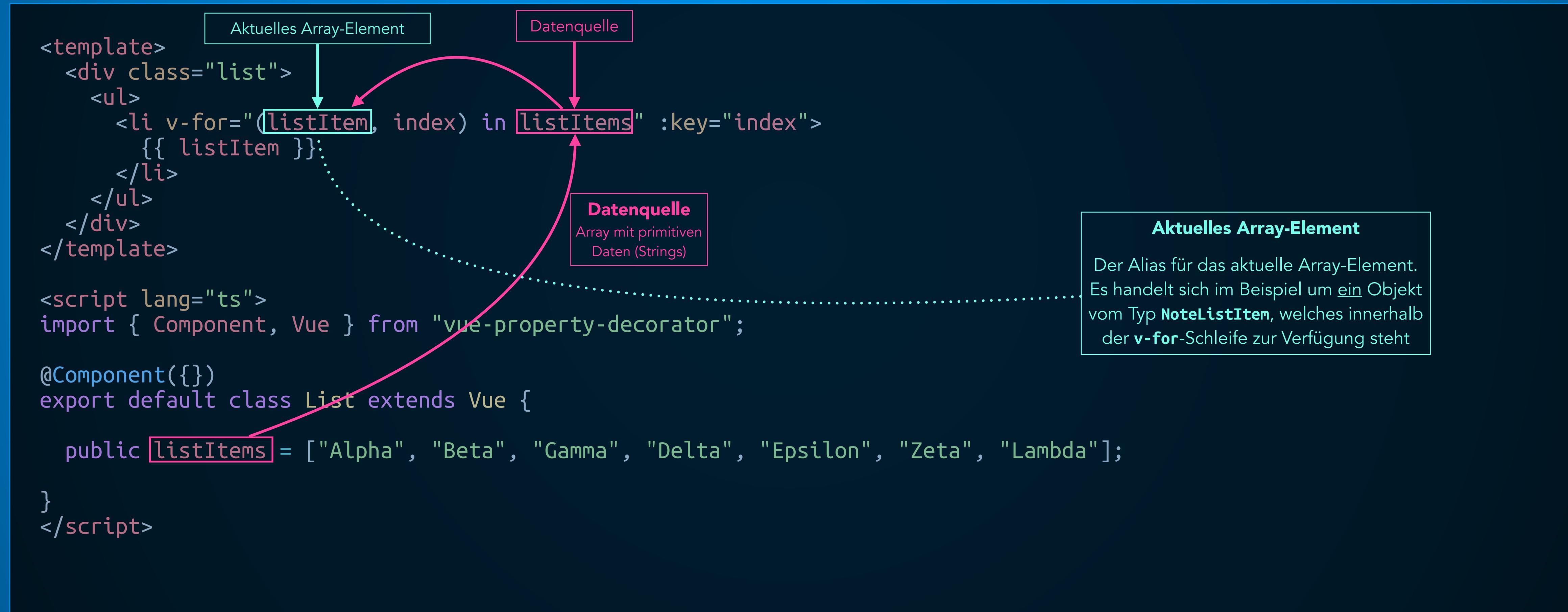
Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/list.html>

Vue 3: <https://v3.vuejs.org/guide/list.html#mapping-an-array-to-elements-with-v-for>

# Vue Basics

## List Rendering mit v-for - Beispiel 1 mit primitivem String-Array (3/4)



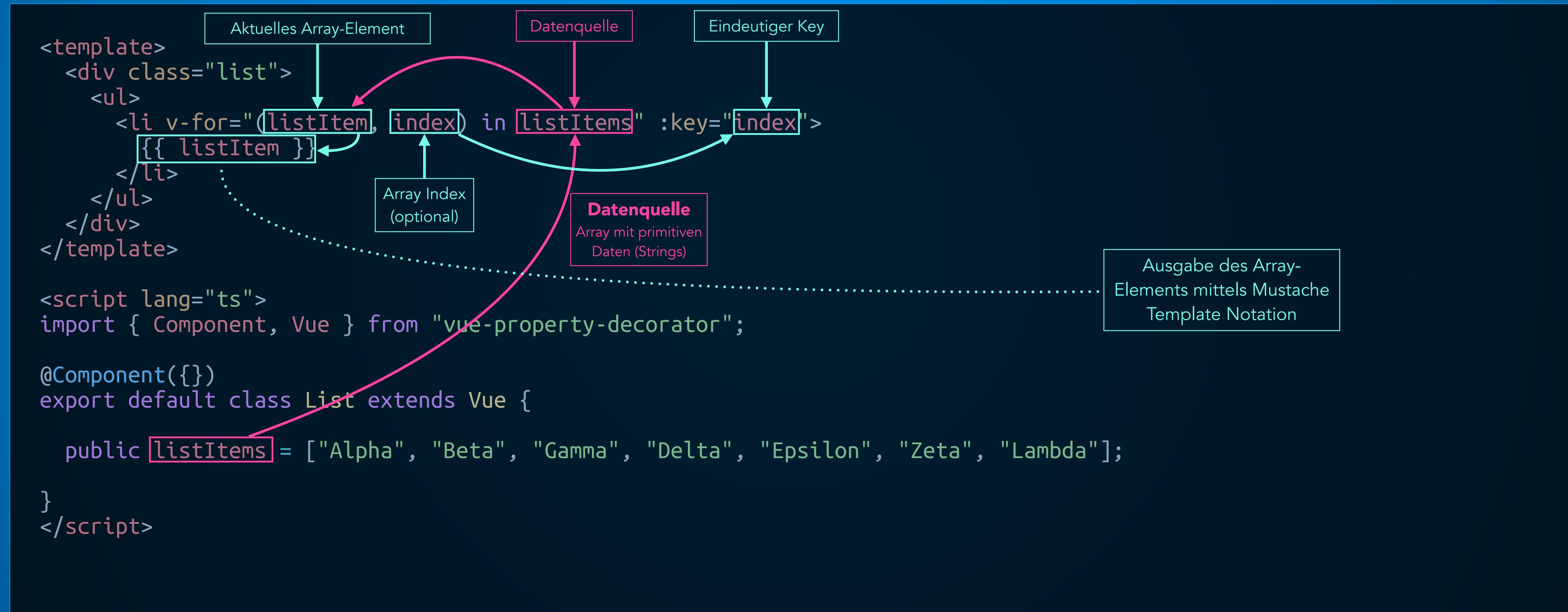
Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/list.html>

Vue 3: <https://v3.vuejs.org/guide/list.html#mapping-an-array-to-elements-with-v-for>

# Vue Basics

## List Rendering mit v-for - Beispiel 1 mit primitivem String-Array (4/4)



Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/list.html>

Vue 3: <https://v3.vuejs.org/guide/list.html#mapping-an-array-to-elements-with-v-for>

# Vue Basics

## List Rendering mit v-for - Beispiel 1 mit primitivem String-Array → Ausgabe

```
<template>
 <div class="list">

 <li v-for="(listItem, index) in listItems" :key="index">
 {{ listItem }}

 </div>
</template>

<script lang="ts">
import ...

@Component({})
export default class List extends Vue {

 public listItems = ["Alpha", "Beta", "Gamma", "Delta",
 "Epsilon", "Zeta", "Lambda"];

}
</script>
```



Rendering-  
Ergebnis im DOM

```
<div class="list">

 Alpha
 Beta
 Gamma
 Delta
 Epsilon
 Zeta
 Lambda

</div>
```

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/list.html>

Vue 3: <https://v3.vuejs.org/guide/list.html#mapping-an-array-to-elements-with-v-for>

# Vue Basics

## List Rendering mit v-for - Beispiel 2 mit komplexem Object-Array (1/5)

```
<template>
 <div class="note-list">
 <NoteListEntry v-for="noteListItem in noteListItems" :key="noteListItem.id"
 class="note-list__item"
 :noteListItem="noteListItem"
 @selected="selectedItem"/>
 </div>
</template>

<script lang="ts">
import ...

@Component({components: {NoteListEntry}})
export default class NoteList extends Vue {

 @Prop({required: true, type: Array as PropType<NoteListItem[]>})
 public noteListItems!: NoteListItem[];

 public selectedItem(noteListItem: NoteListItem): void {
 // do something here
 }
}
</script>
```

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/list.html>

Vue 3: <https://v3.vuejs.org/guide/list.html#mapping-an-array-to-elements-with-v-for>



# Vue Basics

## List Rendering mit v-for - Beispiel 2 mit komplexem Object-Array (2/5)

```
<template>
 <div class="note-list">
 <NoteListEntry v-for="noteListItem in noteListItems" :key="noteListItem.id"
 class="note-list__item"
 :noteListItem="noteListItem"
 @selected="selectedItem"/>
 </div>
</template>

<script lang="ts">
import ...

@Component({components: {NoteListEntry}})
export default class NoteList extends Vue {

 @Prop({required: true, type: Array as PropType<NoteListItem[]>})
 public noteListItems!: NoteListItem[];

 public selectedItem(noteListItem: NoteListItem): void {
 // do something here
 }
}</script>
```

Datenquelle

Datenquelle  
Ein Object-Array mit  
Objekten vom Typ  
**NoteListItem**

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/list.html>

Vue 3: <https://v3.vuejs.org/guide/list.html#mapping-an-array-to-elements-with-v-for>

# Vue Basics

## List Rendering mit v-for - Beispiel 2 mit komplexem Object-Array (3/5)

```
<template>
 <div class="note-list">
 <NoteListEntry v-for="noteListItem in noteListItems" :key="noteListItem.id"
 class="note-list__item"
 :noteListItem="noteListItem"
 @selected="selectedItem"/>
 </div>
</template>

<script lang="ts">
import ...

@Component({components: {NoteListEntry}})
export default class NoteList extends Vue {

 @Prop({required: true, type: Array as PropType<NoteListItem[]>})
 public noteListItems!: NoteListItem[];

 public selectedItem(noteListItem: NoteListItem): void {
 // do something here
 }
}</script>
```

**Aktuelles Array-Element**  
Der Alias für das aktuelle Array-Element.  
Es handelt sich im Beispiel um ein Objekt  
vom Typ **NoteListItem**, welches innerhalb  
der **v-for**-Schleife zur Verfügung steht

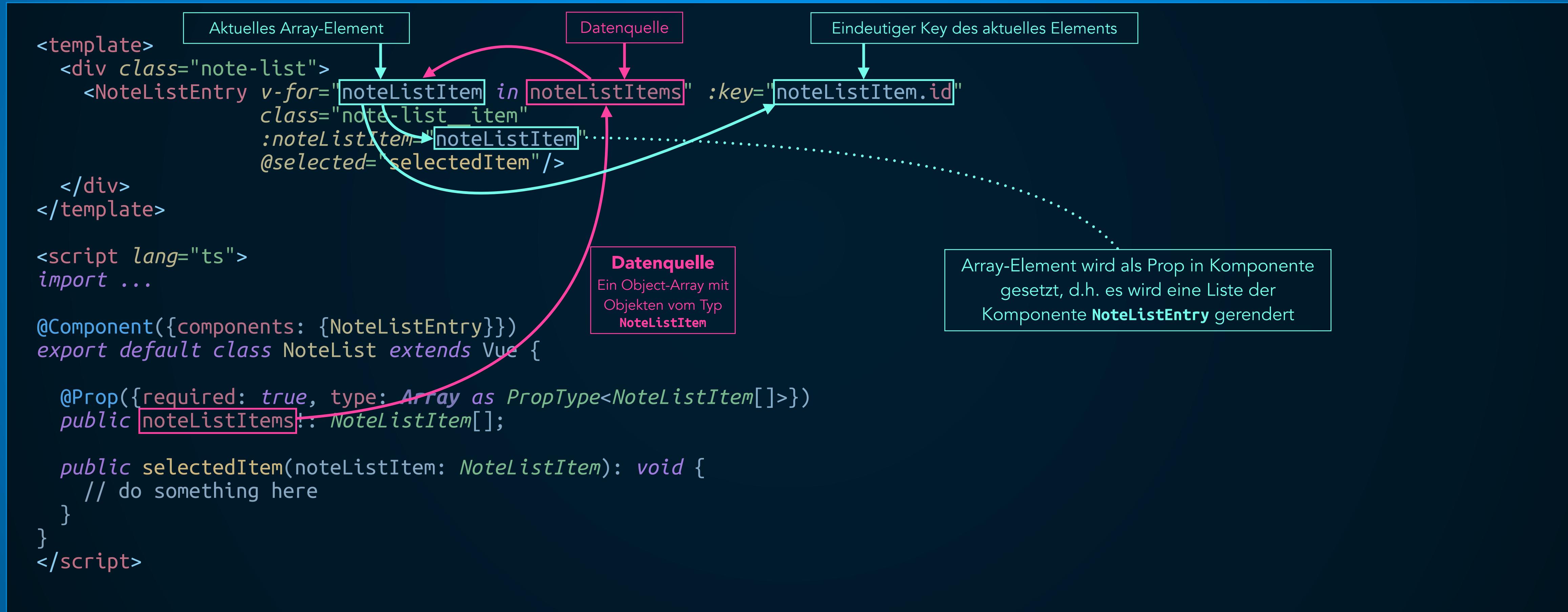
Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/list.html>

Vue 3: <https://v3.vuejs.org/guide/list.html#mapping-an-array-to-elements-with-v-for>

# Vue Basics

## List Rendering mit v-for - Beispiel 2 mit komplexem Object-Array (4/5)



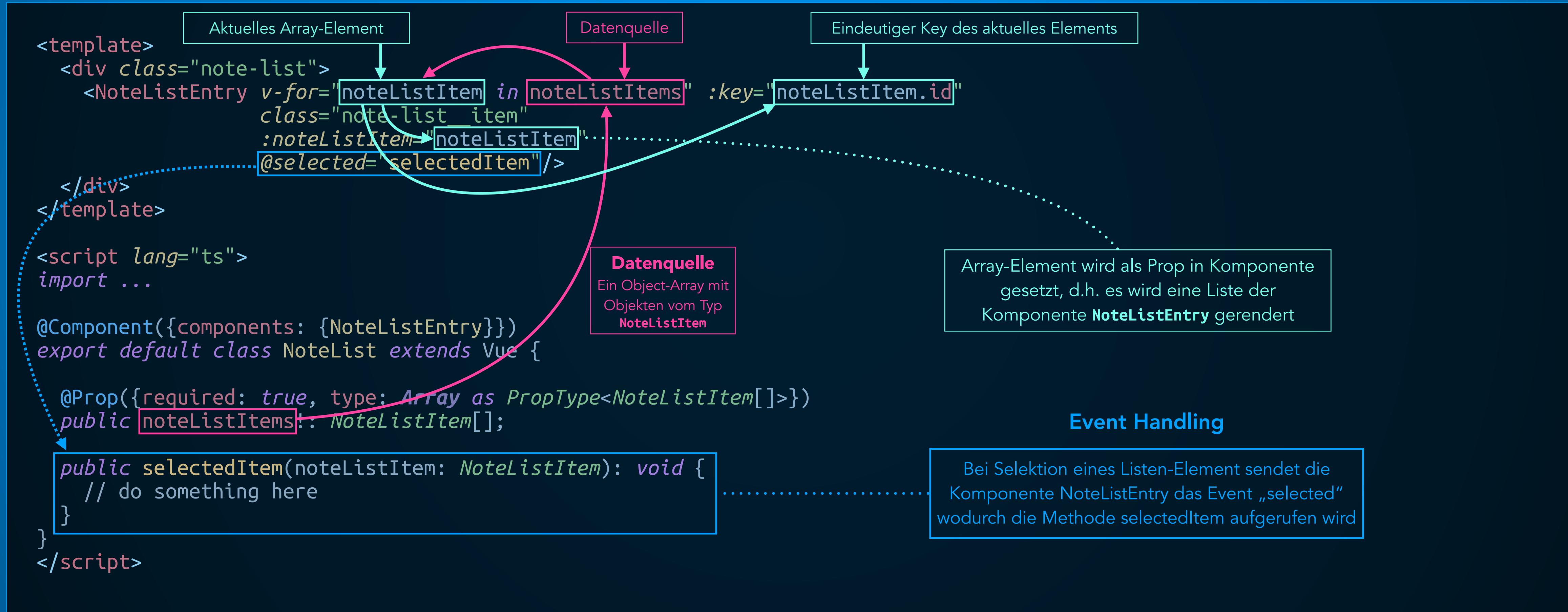
Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/list.html>

Vue 3: <https://v3.vuejs.org/guide/list.html#mapping-an-array-to-elements-with-v-for>

# Vue Basics

## List Rendering mit v-for - Beispiel 2 mit komplexem Object-Array (5/5)



Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/list.html>

Vue 3: <https://v3.vuejs.org/guide/list.html#mapping-an-array-to-elements-with-v-for>

# Vue Basics

## List Rendering mit v-for - Beispiel 2 mit Object-Array → Ausgabe

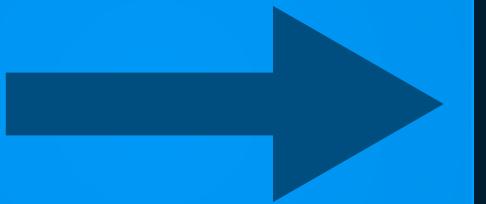
```
<template>
 <div class="note-list">
 <NoteListEntry v-for="noteListItem in noteListItems"
 :key="noteListItem.id"
 class="note-list__item"
 :noteListItem="noteListItem"
 @selected="selectedItem"/>
 </div>
</template>

<script lang="ts">
import ...

@Component({components: {NoteListEntry}})
export default class NoteList extends Vue {

 @Prop({required: true, type: Array as PropType<NoteListItem[]>})
 public noteListItems!: NoteListItem[];

 public selectedItem(noteListItem: NoteListItem): void {
 this.$emit('selectedItem', noteListItem);
 }
}
</script>
```



Rendering-  
Ergebnis im DOM  
(vereinfacht)

```
<div class="note-list">
 <div class="note-list__item note-list-entry">
 <button type="button"
 class="note-list-entry__item">
 Label 1
 </button>
 </div>
 <div class="note-list__item note-list-entry">
 <button type="button"
 class="note-list-entry__item">
 Label 2
 </button>
 </div>
 <div class="note-list__item note-list-entry">
 <button type="button"
 class="note-list-entry__item">
 Label 3
 </button>
 </div>
 [...]
 <div class="note-list__item note-list-entry">
 <button type="button"
 class="note-list-entry__item">
 Label n
 </button>
 </div>
</div>
```

Element-Hierarchie ist abhängig vom Aufbau der Komponente **NoteListEntry**

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/list.html>

Vue 3: <https://v3.vuejs.org/guide/list.html#mapping-an-array-to-elements-with-v-for>

# Agenda Teil 2 / 5

- **Vue Basics**
- CSS-Präprozessoren
- Vue Router
- State-Management mit Vuex
- Vue Cookies



- Template Syntax & Expressions
- Data Binding
- Component Props
- Computed Properties
- Watchers
- Event Handling
- List Rendering
- **Komponenten Kommunikation**
- Class- & Style-Binding
- Debugging

# Vue Basics

## Komponenten Kommunikation (Standard)

Wie kommen Daten zur Anzeige in eine Komponente?

→ Daten werden mittels **@Props** von Parents in Child-Komponenten übergeben

Wie kommen Daten & Changes wieder aus der Komponente raus?

→ Child-Komponenten senden Events an Parent-Komponenten mit oder ohne Payload

Weitere Informationen & Quellen:

Vue 2: Prop: <https://vuejs.org/v2/guide/components.html#Passing-Data-to-Child-Components-with-Props>

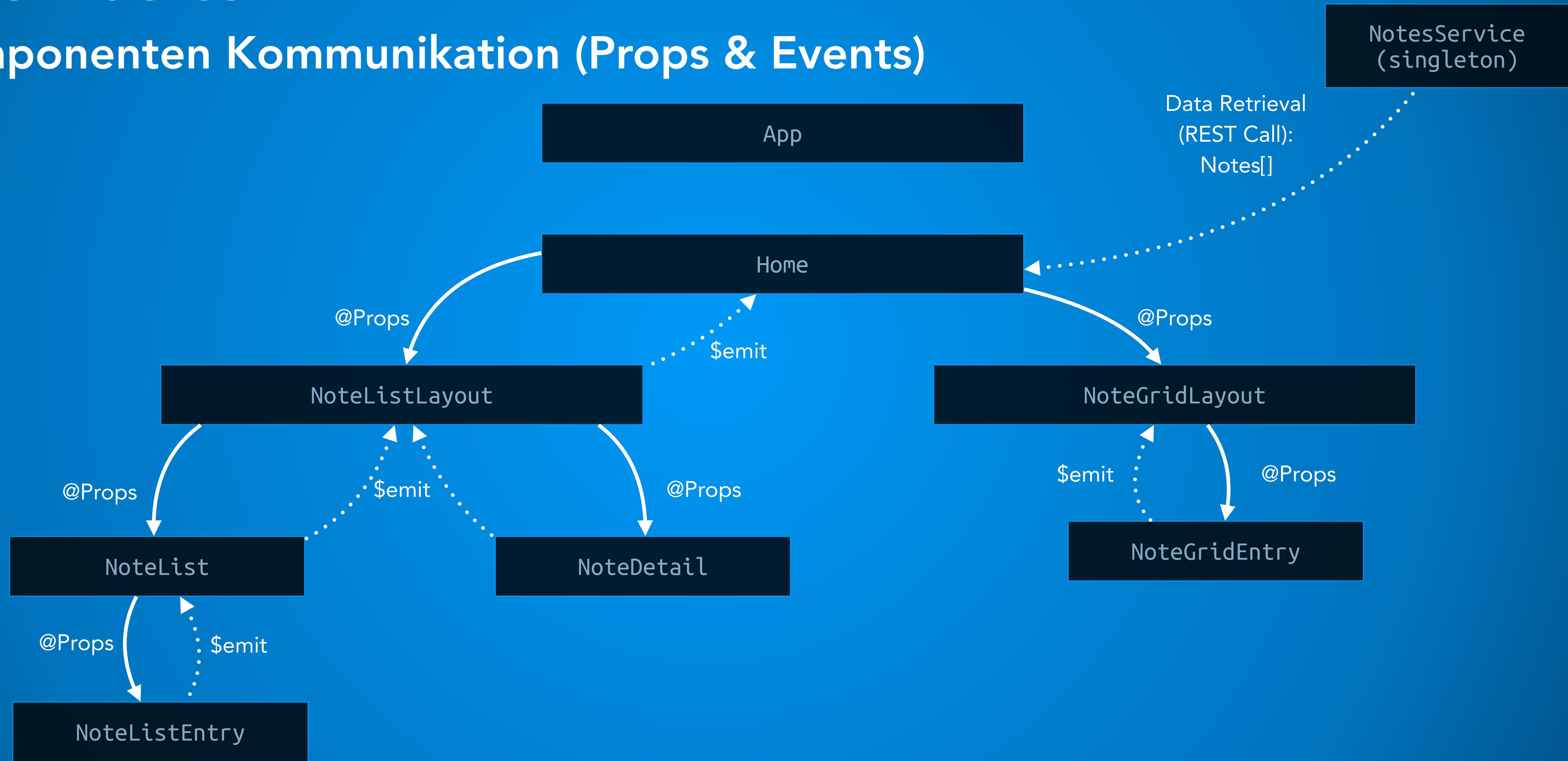
Events: <https://vuejs.org/v2/guide/components.html#Listening-to-Child-Components-Events>

Vue 3: Prop: <https://v3.vuejs.org/guide/component-basics.html#passing-data-to-child-components-with-props>

Events: <https://v3.vuejs.org/guide/events.html#methods-in-inline-handlers>

# Vue Basics

## Komponenten Kommunikation (Props & Events)



# Vue Basics

## Komponenten Kommunikation (Advanced)

- State-Management (Vuex Store)
- Provide & Inject Pattern
- Service Worker
- Services (kein Best Practice)

 Bis auf Vuex wurde nichts im Workshop behandelt. Es handelt sich um alternative Pattern, vorwiegend für komplexere Vue-Apps. Diese Folie dient zur Information.  
Der Einsatz von Service Workern und Services sind keine Vue-spezifische Pattern.

Weitere Informationen:

Vue 2 Provide & Inject: <https://vuejs.org/v2/api/#provide-inject>

Vue 3 Provide & Inject: <https://v3.vuejs.org/guide/component-provide-inject.html>

# Agenda Teil 2 / 5

- **Vue Basics**
- CSS-Präprozessoren
- Vue Router
- State-Management mit Vuex
- Vue Cookies



- Template Syntax & Expressions
- Data Binding
- Component Props
- Computed Properties
- Watchers
- Event Handling
- List Rendering
- Komponenten Kommunikation
- **Class- & Style-Binding**
- Debugging

# Vue Basics

## Class-Binding

- Für dynamisches Hinzufügen & Entfernen von CSS-Classes in HTML-Elementen
- Bei Custom Components werden die CSS-Klassen dem Root-Element hinzugefügt
- Vorhandene Klassen auf Elementen werden nicht überschrieben
- Object- & Array-Syntax verfügbar

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/class-and-style.html>

Vue 3: <https://v3.vuejs.org/guide/class-and-style.html#class-and-style-bindings>



# Vue Basics

## Class-Bindung - Beispiele

### Object Syntax

```
<MyComponent v-bind:class="{ classname: myValue }"></MyComponent>
```

<!-- Echtes Beispiel -->

```
<MyComponent v-bind:class="{ active: isActive }"></MyComponent>
```

Variable aus Script-Block liefert **true** oder **false**.

Wird **true** geliefert, wird die CSS-Class „**active**“ auf MyComponent angewendet, bei **false** nicht.

### Array Syntax

### Array Syntax

```
<div class="classname" :class="[{ 'classname--modifier1': myValue1, 'classname--modifier2': myValue2 }]">
[...]
</div>
```

Variable liefert **true** oder **false**, hier im Beispiel wird direkt auf den Vuex-Store zugegriffen.

Es können beliebig viele Klassen gebunden bzw. eingefügt werden. Meist werden hier nach BEM-Notation "Modifier" eingesetzt.

<!-- Echtes Beispiel -->

```
<div class="note-list" :class="[{ 'note-list--light': !$store.getters.isDarkMode, 'note-list--dark': $store.getters.isDarkMode }]">
[...]
</div>
```

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/class-and-style.html>

Vue 3: <https://v3.vuejs.org/guide/class-and-style.html#class-and-style-bindings>

# Vue Basics

## Style-Binding

- Für dynamisches Hinzufügen & Entfernen von Inline-Styles in HTML-Elementen
- Object- & Array-Syntax verfügbar
- **Empfehlung:** Class-Bindings sind Style-Binding vorzuziehen, weil Style-Bindings
  - Inline-Styles erzeugen, was zu Problemen beim Überschreiben von CSS-Klassen führen kann
  - die saubere Trennung des Code in Template und Style-Code verhindern

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/class-and-style.html>

Vue 3: <https://v3.vuejs.org/guide/class-and-style.html#class-and-style-bindings>



# Vue Basics

## Style-Binding - Beispiele

Dynamische CSS-Property-Values vom Typ String aus dem Script-Block können genutzt werden, um das Aussehen eines Elements zu modifizieren, direkt als Inline-Style.

### Object Syntax

```
<MyComponent v-bind:style="{ cssproperty1: myValue1_asString, cssproperty2: myValue2_asString }"></MyComponent>

<!-- Echtes Beispiel -->
<MyComponent v-bind:style="{ color: myColor, background: myBgColor, font-size: myFontSize + 'px' }"></MyComponent>
z.B. "red" z.B. "#00ff00" z.B. "14"
```

Die Array-Syntax für v-bind:style ermöglicht es, mehrere Stilobjekte auf dasselbe Element anzuwenden.

### Array Syntax

```
<MyComponent v-bind:style="[baseStyles, overridingStyles]"></MyComponent>

<MyComponent v-bind:style="[{ color: myColor }, { width: myValue + '%', height: myValue + 'px' }]"></MyComponent>
```

Weitere Informationen & Quellen:

Vue 2: <https://vuejs.org/v2/guide/class-and-style.html#Object-Syntax>

Vue 3: <https://v3.vuejs.org/guide/class-and-style.html#class-and-style-bindings>

# Agenda Teil 2 / 5

- **Vue Basics**
- CSS-Präprozessoren
- Vue Router
- State-Management mit Vuex
- Vue Cookies



- Template Syntax & Expressions
- Data Binding
- Component Props
- Computed Properties
- Watchers
- Event Handling
- List Rendering
- Komponenten Kommunikation
- Class- & Style-Binding
- **Debugging**

# Vue Basics

## Debugging

Mehrere Möglichkeiten der Fehleranalyse stehen zur Verfügung:

- Vue Dev Tools
- Browser Dev-Tools → Breakpoints setzen mit `debugger`; Keyword
- Browser Console
- NodeJS Console in Entwicklungsumgebung
- Style Linter Errors

# Vue Basics

## Debugging - Vue Dev Tools

The screenshot shows the Vue Dev Tools extension in a browser's developer tools. A red box highlights the 'Vue' tab in the top navigation bar. A red arrow points from a text box above to this tab. Another red box highlights the 'Components' tab in the main toolbar. A red arrow points from another text box below to this tab.

**Nach Installation der Extension öffnet sich ein neuer Tab in den Browser- Dev-Tools**

**Menü zur Auswahl von Vue-App Bereichen**

The interface includes:

- Component Hierarchy:** Shows the tree structure of the application components, starting from <Root> and <App>, down to <NoteGridLayout> and <NoteGridEntry>.
- Component Details:** Provides detailed information about the selected component, including its props (noteGridItem) and data (\$route).
- Console:** A bottom pane for running JavaScript in the browser's console.

The application itself, 'Notes App', is visible on the left, showing two note cards: 'asymmetric' and 'Virtual'.

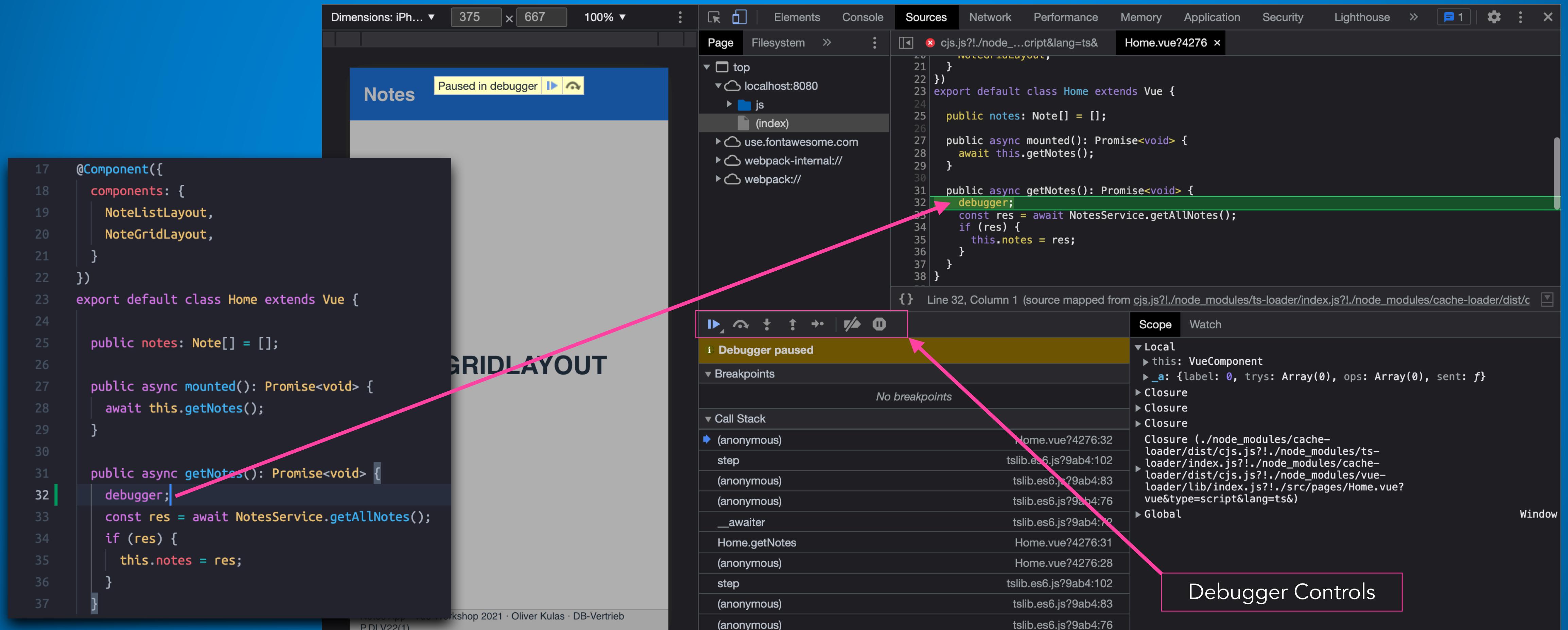
Weitere Informationen & Quellen:

Vue2 Dev Tools: <https://vuejs.org/v2/guide/installation.html#Vue-Devtools>

Vue3 Dev Tools: <https://v3.vuejs.org/guide/installation.html#vue-devtools>

# Vue Basics

## Debugging - Browser Dev Tools → Breakpoints setzen mit debugger; Keyword



Weitere Informationen:

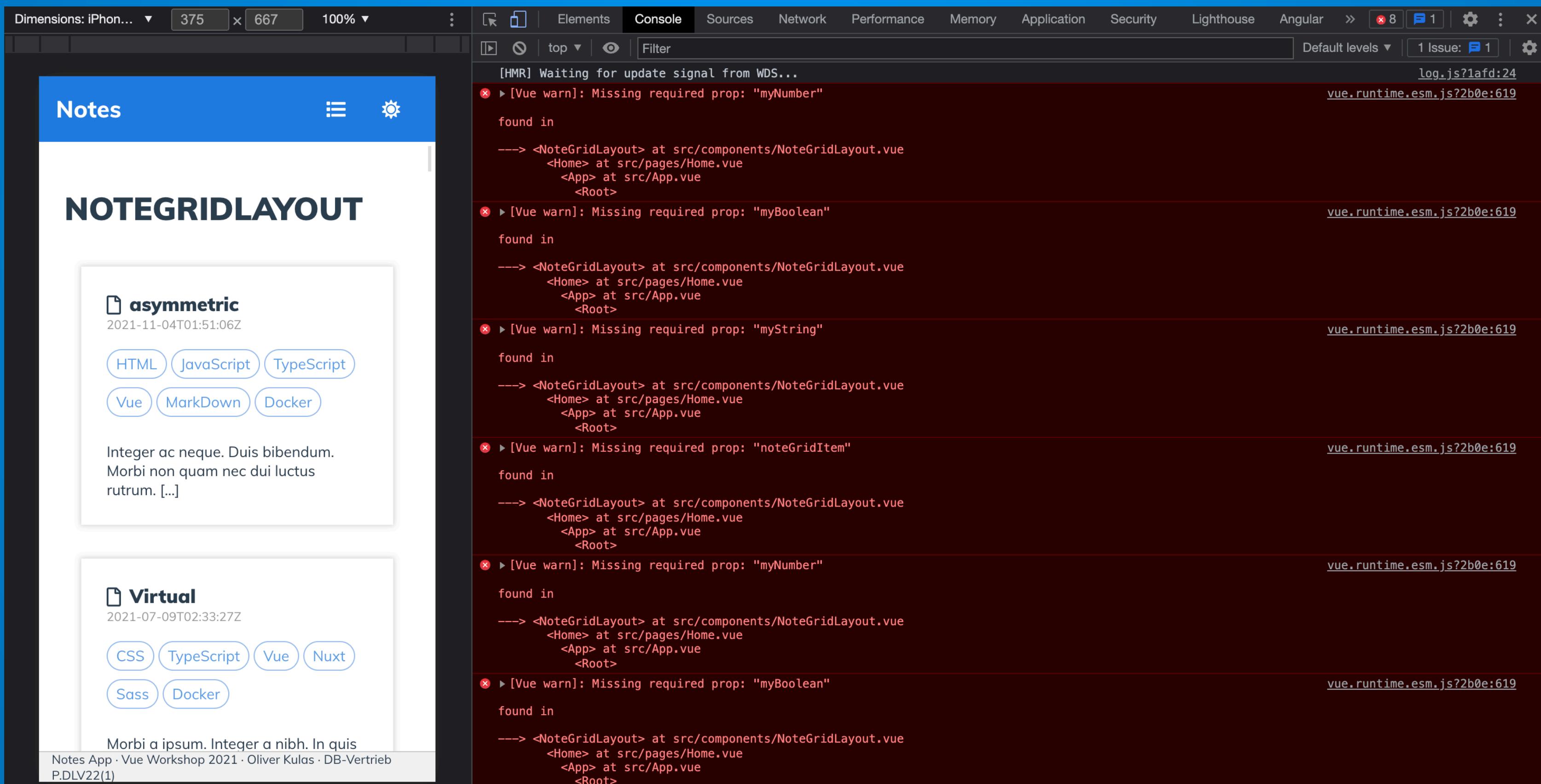
Chrome Dev Tools: <https://developer.chrome.com/docs/devtools/>

Firefox Dev Tools: <https://developer.mozilla.org/en-US/docs/Tools>

# Vue Basics

## Debugging - Browser Console

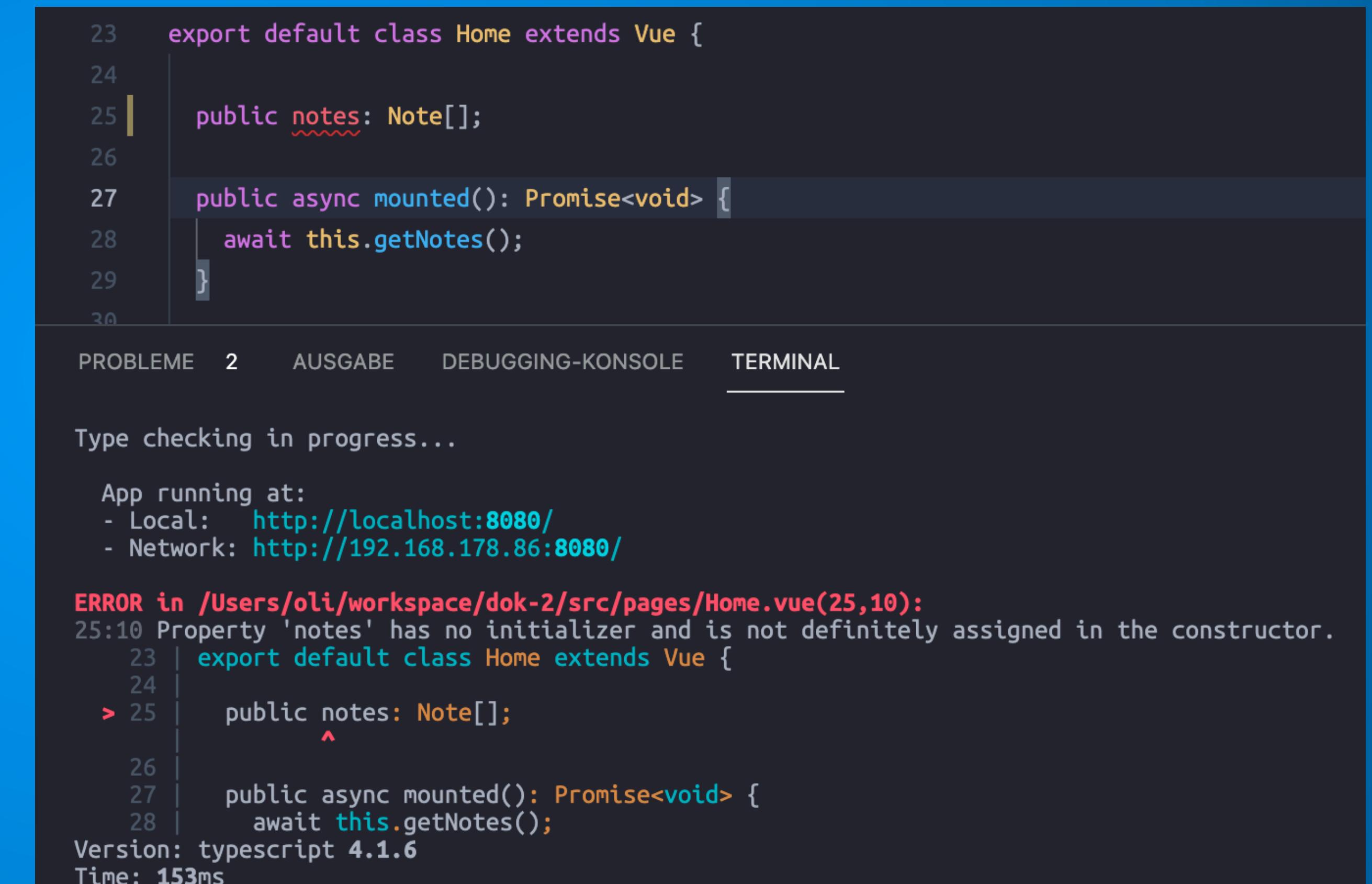
- Vue gibt aussagekräftige Warnungen und Fehlermeldungen in Browser-Console aus:



# Vue Basics

## Debugging - Node Console in Entwicklungsumgebung

- Vue-Prozess, JavaScript bzw. TypeScript-Compiler, diverse Linter und Helper geben in NodeJS Prozess aussagekräftige Hinweise, Warnungen und Fehler aus



The screenshot shows a terminal window with the following content:

```
23 export default class Home extends Vue {
24
25 | public notes: Note[];
26
27 | public async mounted(): Promise<void> {
28 | await this.getNotes();
29 }
30
PROBLEME 2 AUSGABE DEBUGGING-KONSOLE TERMINAL

Type checking in progress...

App running at:
- Local: http://localhost:8080/
- Network: http://192.168.178.86:8080/

ERROR in /Users/oli/workspace/dok-2/src/pages/Home.vue(25,10):
25:10 Property 'notes' has no initializer and is not definitely assigned in the constructor.
 23 | export default class Home extends Vue {
 24 |
> 25 | public notes: Note[];
 |
 26 |
 27 | public async mounted(): Promise<void> {
 28 | await this.getNotes();
Version: typescript 4.1.6
Time: 153ms
```

# Vue Basics

## Debugging - Style Linter Errors

- Der Linter für den CSS-Style verhindert das Rendering der App im Frontend und gibt aussagekräftige Fehler direkt im Browser aus
- Voraussetzung: Style-Linting ist eingebunden und korrekt konfiguriert

The screenshot shows a browser's developer tools console with a dark theme. At the top, it says "Failed to compile." Below that is a long stack trace of module imports related to CSS loaders. The main error message is:

```
Module build failed (from ./node_modules/sass-loader/dist/cjs.js):
SassError: Undefined variable.
```

Line 59 of the file `src/components>NoteDetail.vue` contains the following code:

```
padding: $spaceL;
```

The word `$spaceL` is underlined with a red squiggly line, indicating it is undefined. The console also shows other components of the application, such as "Virtual", "functionalities", and "Ergonomic", each with their own details and tags like CSS, JavaScript, Vue, Nuxt, Docker, etc.

# Agenda Teil 3 / 5

- Vue Basics
- **CSS-Präprozessoren**
- Vue Router
- State-Management mit Vuex
- Vue Cookies

# CSS-Präprozessoren

## Sass/SCSS

- Verschachteln von CSS-Selektoren
- Aktiveren von SCSS im Style-Block
- CSS-Scope auf Komponente begrenzt
- BEM-Notation

```
<style lang="scss" scoped>
```

```
<component-name>__<element>--<modifier>
```

Weitere Informationen & Quellen:

SASS: <https://sass-lang.com/>

BEM-Notation: <http://getbem.com/introduction/>



# CSS-Präprozessoren

## Weitere nützliche (S)CSS-Features

- Variablen
- Functions, z.B. calc()
- Mixins

```
// declare variable
$my-variable = 42px;

// use variable
height: $my-variable;

// calculate values
width: calc(100% - 42px);

// declare mixin (with calculation)
@mixin boxContentLayout { ... }

// use the mixin within a css-class
@include boxContentLayout;

// mixin example using calc() and calculate with variable
@mixin boxContentLayout {
 width: calc(100% - (#{$my-variable} * 2));
}
```

Weitere Informationen & Quellen:

SASS: <https://sass-lang.com/>

BEM-Notation: <http://getbem.com/introduction/>

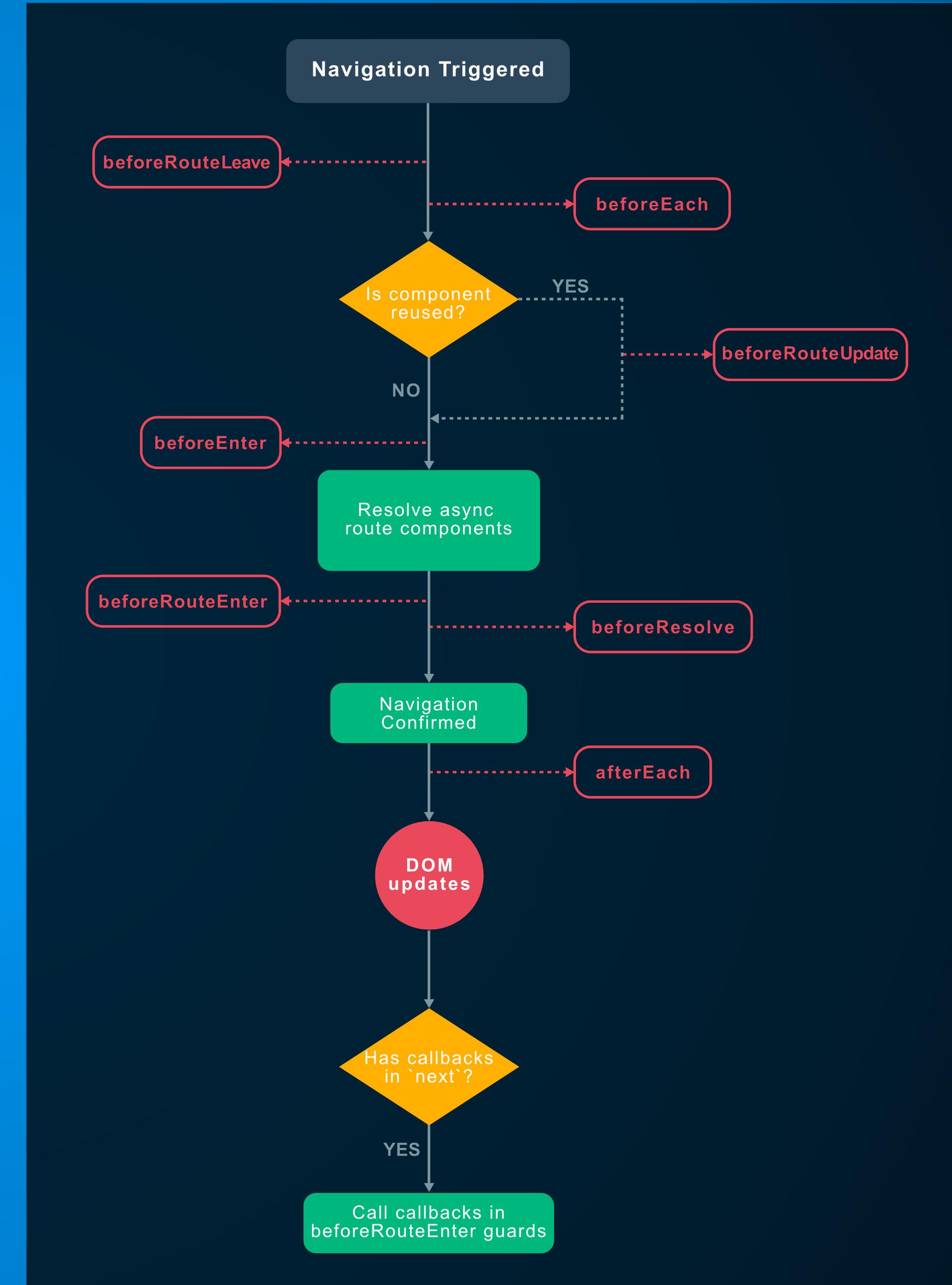
# Agenda Teil 3 / 5

- Vue Basics
- CSS-Präprozessoren
- **Vue Router**
- State-Management mit Vuex
- Vue Cookies

# Vue Router

## Vue 2.x

- Router Lifecycle inkl. Lifecycle Hooks
- „Router Hooks“ sind hier  
**NavigationGuards**



Quellen:

<https://github.com/vuejs/vue-router/issues/2184#issuecomment-393484643>

<https://router.vuejs.org/guide/advanced/navigation-guards.html#navigation-guards>

# Vue Router

## Vue 2.x , Beispiel für Router-Konfiguration inkl. NavigationGuard

```
Vue.use(VueRouter)

const routes: Array<RouteConfig> = [
 {
 path: '/',
 name: 'Home',
 component: Home,
 beforeEnter: (to: Route, from: Route, next: NavigationGuardNext<Vue>) => {
 // do something
 },
 },
 {
 path: '*',
 name: AppRouteNames.PageNotFound,
 component: PageNotFound,
 redirect: {
 path: '/'
 },
 },
]
```

# Vue Router

## Vue 2.x mit TypeScript; Beispiele für Router-Push

```
// push to router to named route "Home"
this.$router.push({name: 'Home'})
```

```
// push to router path to path /note
this.$router.push({path: `/note/${noteListItem.id}`})
```

```
// push to router to path with path parameter /notelist/:notId
this.$router.push({path: `/notelist/${noteListItem.id}`})
```

Weitere Informationen & Quellen:

Vue Router - Programmatic Navigation: <https://router.vuejs.org/guide/essentials/navigation.html>

Vue Router - Named Routes: <https://router.vuejs.org/guide/essentials/named-routes.html>



# Agenda Teil 4 / 5

- Vue Basics
- CSS-Präprozessoren
- Vue Router
- **State-Management mit Vuex**
- Vue Cookies

# State Management mit Vuex

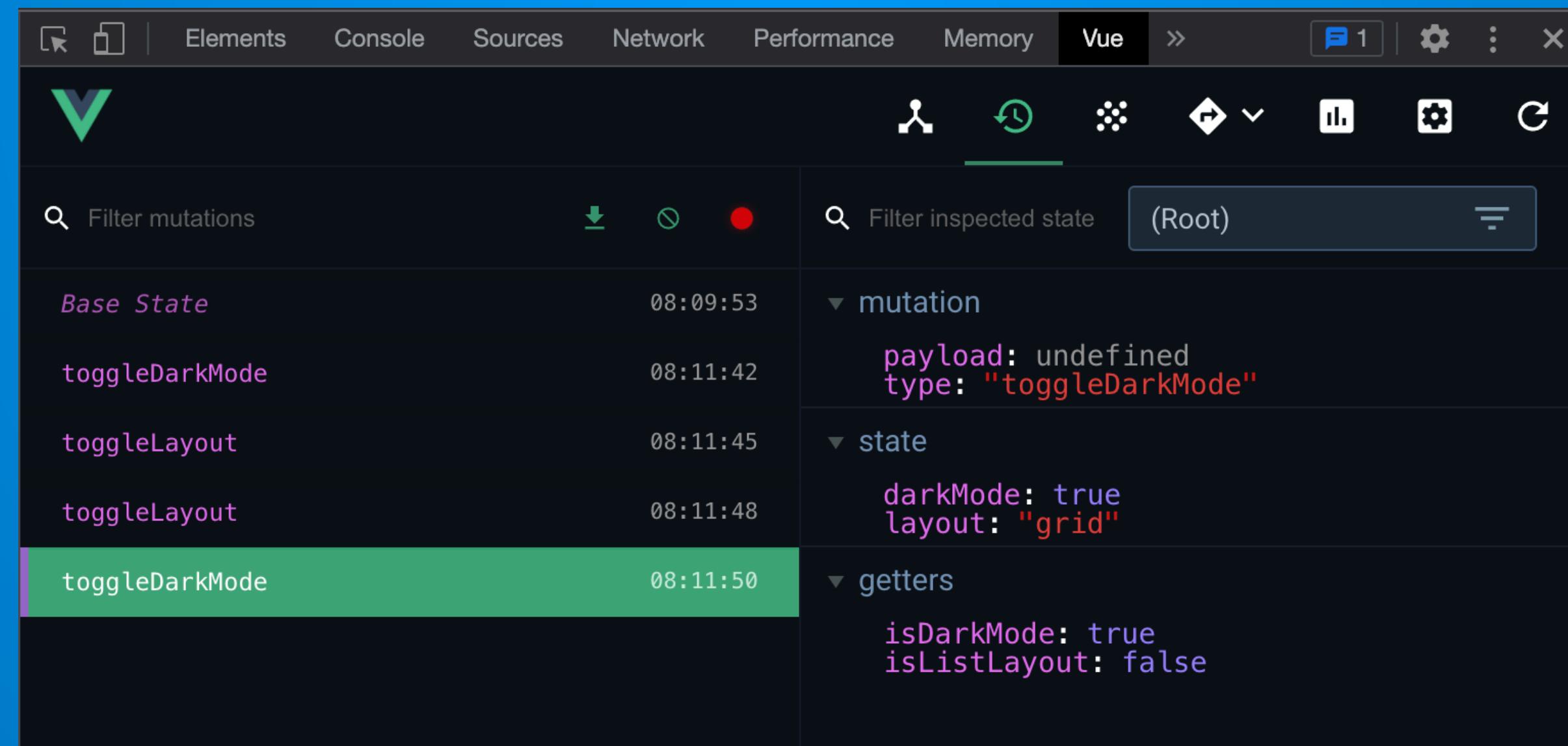
## Vuex Store

- Vuex ist ein State-Management-Pattern & Library für Vue
- Zentraler Speicher für alle Komponenten einer Anwendung → **Store**
- **Alle Daten im Vuex-Store sind reaktiv!**
  - Daten sind, ähnlich wie bei Two-Way-Data-Binding, in beide Richtung reaktiv
  - D.h. ändert man einen Wert im Store, verändert sich dieser Wert auch in den Komponenten und damit auch im View → UI-Update
  - Wird ein Wert über die UI verändert, wird automatisch der Wert im Store aktualisiert

# State Management mit Vuex

## Vuex Store

- Debugging mit Vue Dev Tools möglich
- Die Zustandsveränderungen im Store erfolgen chronologisch, navigiert man in den Dev-Tools zu einem älteren State spricht man auch von „Time Traveling“



# State Management mit Vuex

## Vuex Store

- **Wann wird Vuex eingesetzt?**

→ Vuex ermöglicht die zentrale Zustandsverwaltung, auf Kosten von einem Boilerplate Code und eines etwas komplexeren Konzepts. Es ist ein Kompromiss zwischen kurzfristiger und langfristiger Produktivität innerhalb der Anwendung.

→ Vuex eignet sich meist für mittlere oder größere Anwendungen. Bei kleineren Anwendungen kommt man auch ohne Vuex aus.

# State Management mit Vuex

## Grundprinzip

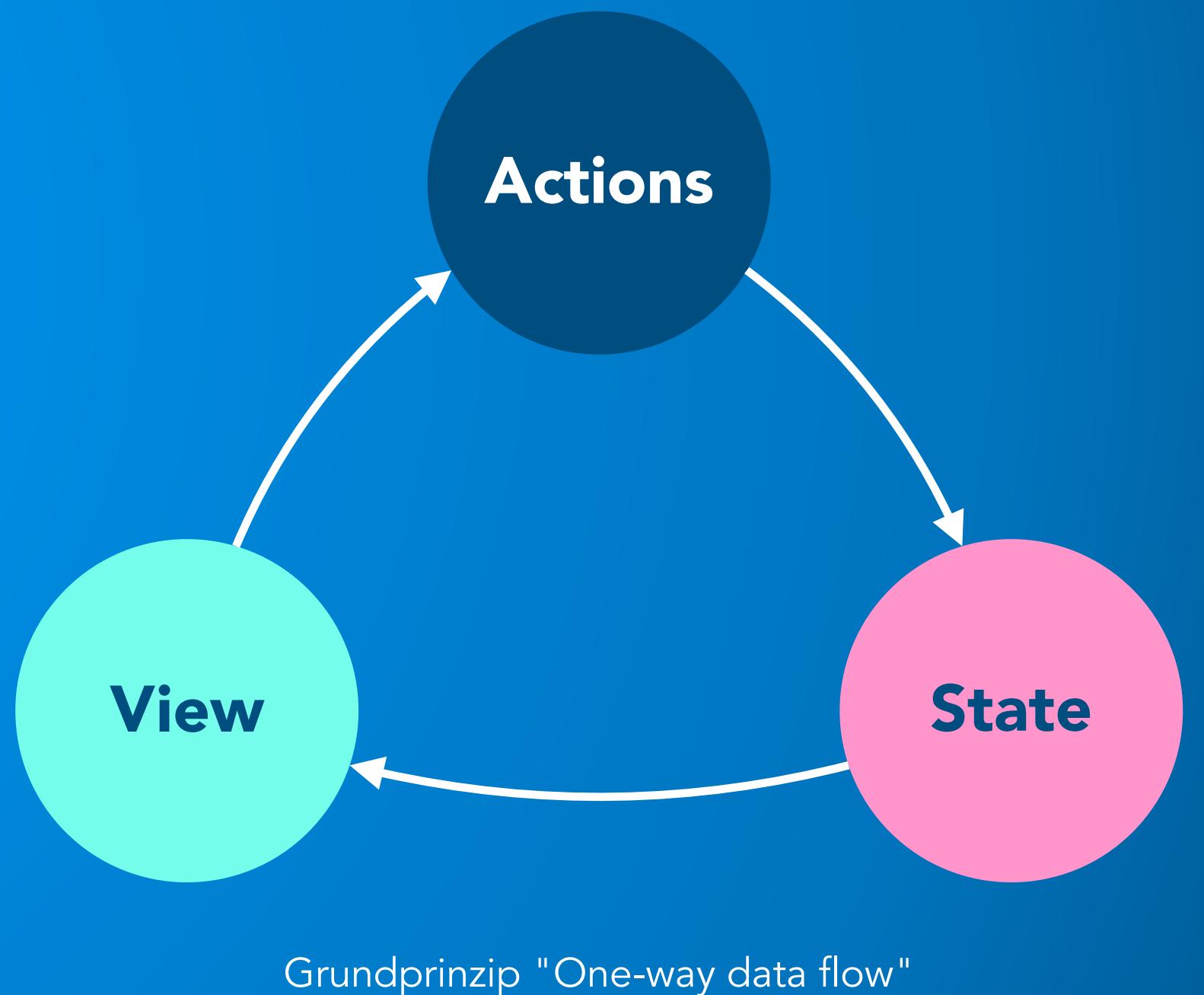
- ...ist eine in sich geschlossener Teil einer Vue-App, bestehend aus folgenden Teilen:

**State** → die Quelle der Wahrheit für die App

**View** → die deklarative Abbildung des **State**

**Actions** → die Möglichkeiten, über die der **State** als Reaktion auf Benutzereingaben den **View** verändern

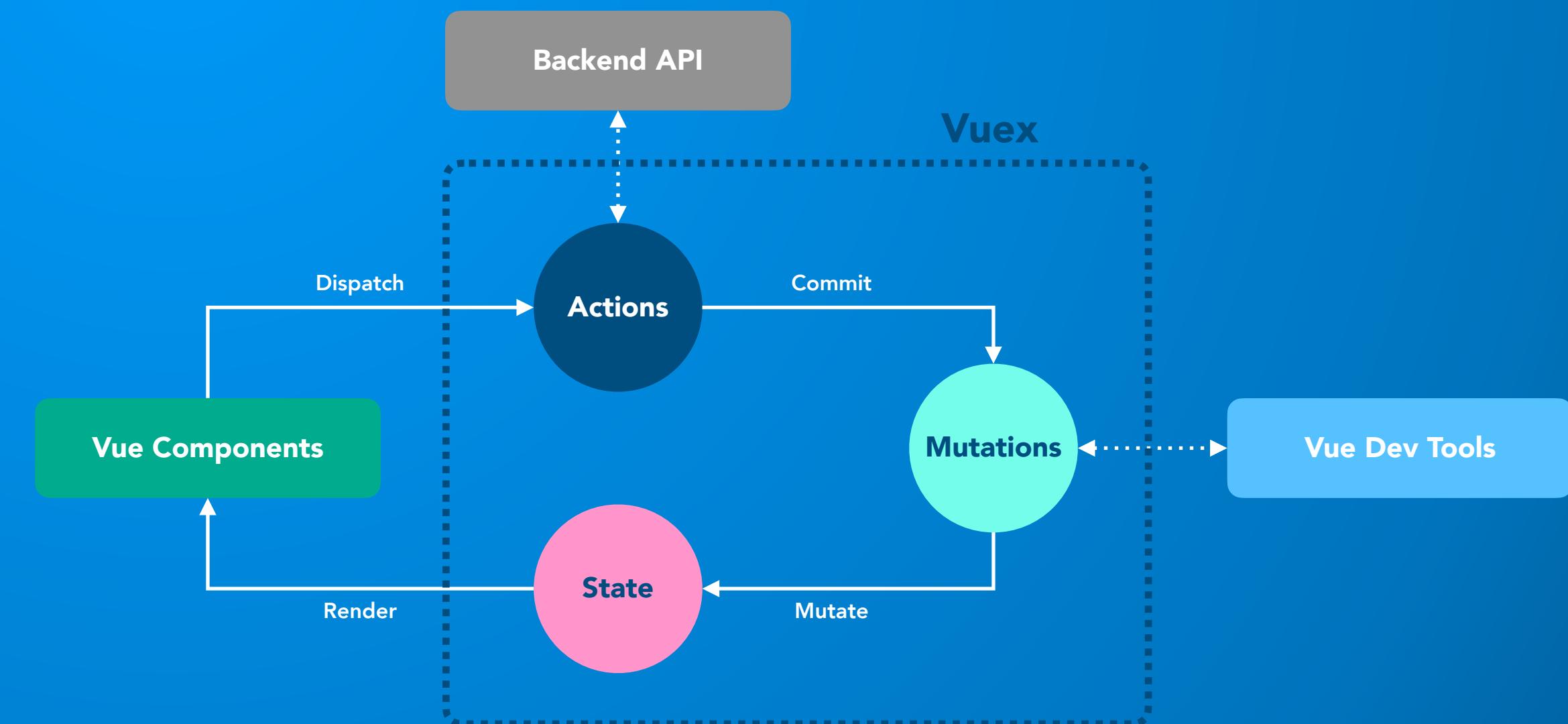
→ Sobald mehrere Komponenten den State verändern ist eine standardisiertes Pattern nötig, damit der Prozess immer auf die gleiche Weise erfolgt



# State Management mit Vuex

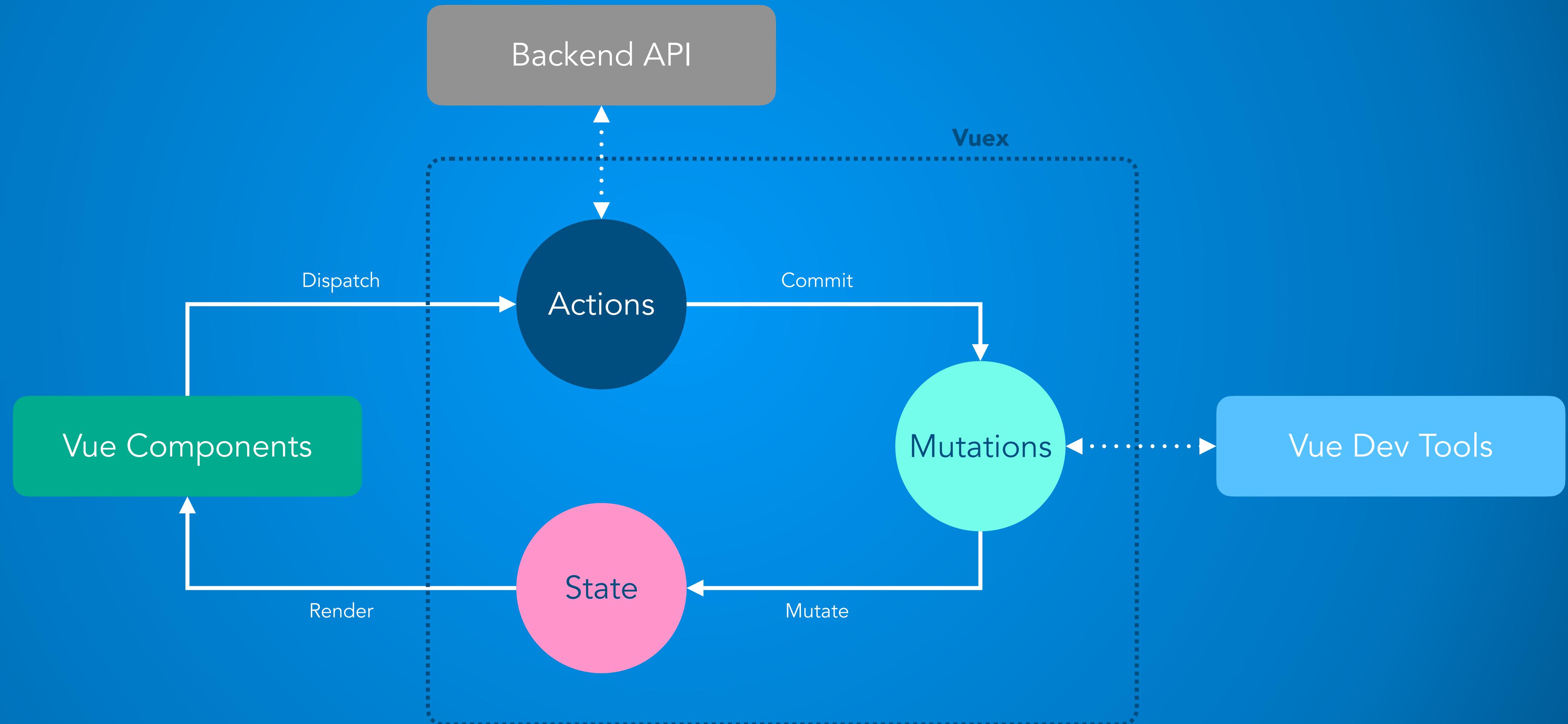
## Standard Pattern

- Vue Komponenten stoßen Updates über **Actions** an → **Dispatch**
- **Actions** fassen (komplexere) Transaktionen zusammen
- **Actions** lösen **Mutations** aus, die tatsächlichen Changes im Store
- **Mutations** verändert den State



# State Management mit Vuex

## Standard Pattern



# State Management mit Vuex

Vuex Zusammenfassung (Vue Mastery)



# State Management mit Vuex

## Vue-Setup-Beispiel mit leerem Vuex-Store (ohne State, Actions & Mutations)

```
// File: /store/index.ts

import {State} from "@/models/State";
Vue.use(Vuex);

export default new Vuex.Store<State>({
 state: {
 },
 mutations: {
 },
 actions: {
 },
 getters: {
 }
 plugins: []
});
```

import

```
// File: /main.ts

import Vue from "vue";
import App from "./App.vue";
import router from "./router/router";
import store from "./store";
import './styles/main.scss';
import VueCookies from 'vue-cookies';

new Vue({
 router,
 store,
 render: (h) => h(App)
}).$mount("#app");
```

# State Management mit Vuex

## Einfacher Vuex-Store mit State, Getters und Mutations - Beispiel

```
// File: /store/index.ts

Vue.use(Vuex);

export default new Vuex.Store<State>({
 state: {
 darkMode: false,
 layout: AppLayout.List
 },
 mutations: {
 toggleDarkMode(state) {
 state.darkMode = !state.darkMode;
 },
 toggleLayout(state) {
 state.layout = state.layout === AppLayout.List ? AppLayout.Grid : AppLayout.List
 }
 },
 actions: {},
 getters: {
 isDarkMode: state => {
 return state.darkMode;
 },
 isListLayout: state => {
 return state.layout === AppLayout.List
 }
 }
});
```

# State Management mit Vuex

## Vuex + VuexPersistence

- Um den State des Vuex-Store über die Browser-Session hinaus zu persistieren kann das Vuex-Plugin um weitere Funktionen erweitert werden
- z.B. durch **vue-persist**, ein Plugin für das Vuex-Vue-Plugin
- VuexPersistence ermöglicht das Speichern des Vuex-Store in
  - den Browser **LocalStorage** → Persistiert Daten für unbegrenzte Dauer (bis Browserdaten im Browser gelöscht werden)
  - den Browser **SessionStorage** → Persistiert Daten nur für die Dauer der Browser-Session, d.h. solange Browser-Fenster bzw. -Tab geöffnet ist

# State Management mit Vuex

## Verwenden von VuePersistence als Vuex-Plugin - Beispiel

```
// File: /store/index.ts
Vue.use(Vuex);

const vuexLocal = new VuexPersistence<State>({
 storage: window.localStorage,
});

export default new Vuex.Store<State>({
 state: {
 ...
 },
 mutations: {
 ...
 },
 actions: {
 ...
 },
 getters: {
 ...
 },
 plugins: [vuexLocal.plugin]
})
```

Instanziierung des VuexPersistence Objects

In den VuexPersistence-Options kann der Ort bzw. die Dauer der Speicherung festgelegt werden. Hier wird der Store dauerhaft im **LocalStorage** des Browsers persistiert

Alternativ kann Store mit **window.sessionStorage** im **SessionStorage** des Browsers gespeichert werden.

Verwenden des VuexPersistence-Plugin im Vuex-Store

# State Management mit Vuex

## Zugriff auf den Vuex-Store aus Vue-Komponenten

- Wie jedes Plugin steht auch der Vuex-Store innerhalb der gesamten Vue-App auf einfache Weise zur Verfügung:

```
// access store state directly without store getter
this.$store.state.myStateProperty

// use store getters
this.$store.getters.myStoreGetter

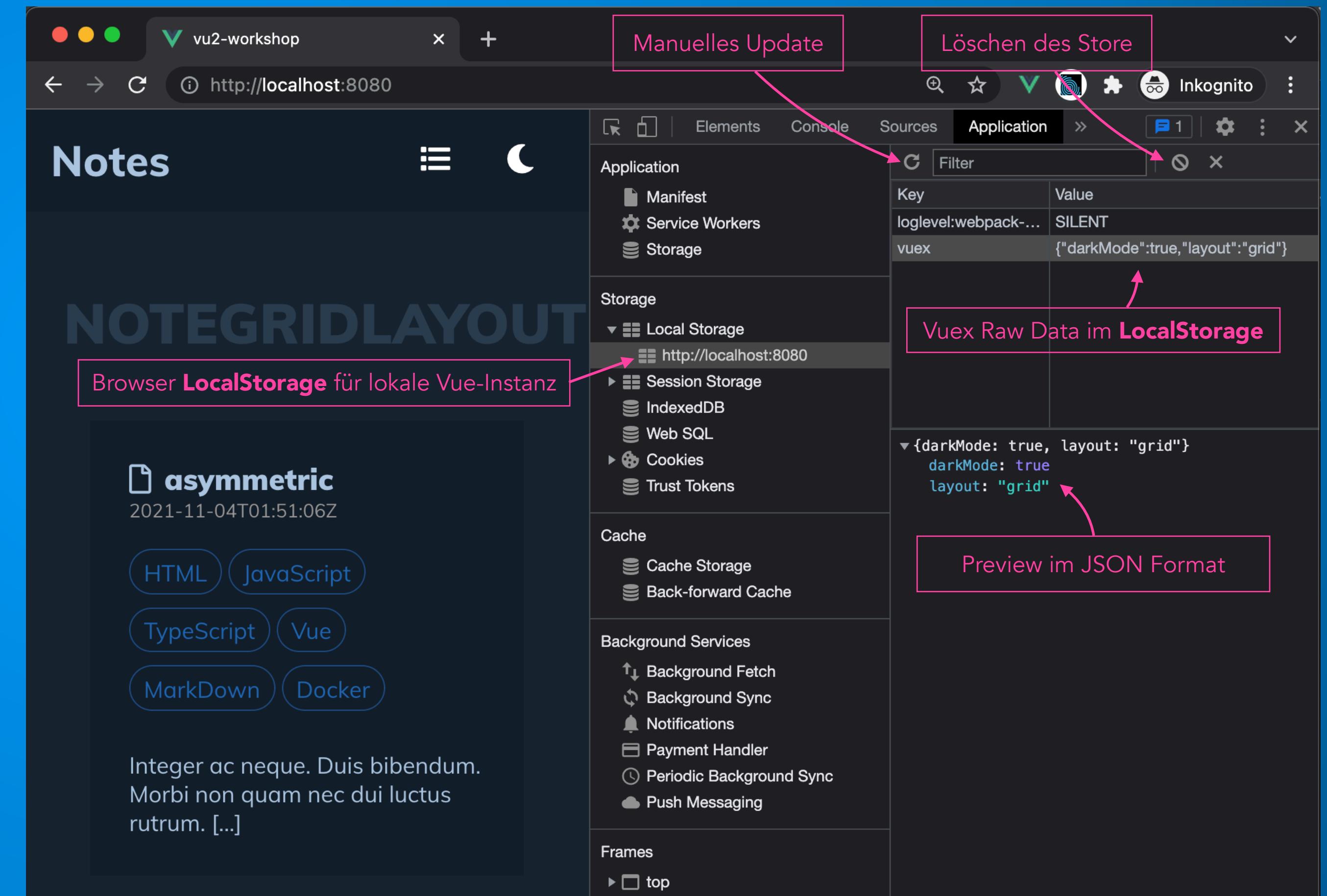
// commit mutation directly without action
this.$store.commit('myMutation', myPayload);

// Dispatch complex mutation via store action
this.$store.dispatch('myStoreAction', myPayload);
```

# State Management mit Vuex

## Vuex-Store im LocalStorage des Browsers (Browser Dev-Tools)

- Initial ist der Vuex-Store nicht in Browser Dev-Tools zu sehen
- Erst mit der ersten Änderung im Store innerhalb der Vue-App, ist Vuex im BrowserStorage zu sehen
- Manchmal ist ein manuelles Update der Ansicht nötig
- Nach bestimmten Änderungen ist es nötig den Store manuell zu löschen, damit er neu angelegt wird



# Agenda Teil 4 / 5

- Vue Basics
- CSS-Präprozessoren
- Vue Router
- State-Management mit Vuex
- **Cookies**

# Cookies

## Was sind Cookies?



- Cookies sind kleine Textdateien, die Websites lokal dem Computer speichern und es den Websites ermöglichen, während der Session Informationen zu speichern und zu nutzen
- Gewöhnlich werden Cookies verwendet zum:
  - Speichern von Einstellungen, damit User beim erneuten Besuch der Webseite die Auswahl nicht erneut eingeben müssen
  - Halten der Login-Session
  - Analysieren des User-Verhaltens, damit die User Experience verbessert werden kann
- Cookies haben festgelegtes Ablaufdatum, an dem sich automatisch gelöscht werden

# Cookies

## Arten von Cookies



- **First-Party-Cookies** werden von der eigenen Website erstellt.  
Die Site wird in der Adressleiste des Browsers angezeigt.
- **Third-Party-Cookies** sind Cookies anderer Websites (Drittanbieter).
  - Diese Art von Cookies wird normalerweise verwendet, wenn Webseiten Inhalte von externen Websites enthalten, z.B. Bannerwerbung.
  - Dies eröffnet die Möglichkeit, den Browserverlauf des Benutzers zu verfolgen und wird häufig von Werbetreibenden verwendet, um jedem Benutzer relevante Werbung zu liefern.
  - Der Tracking-Anbieter registriert die Besuche der mit ihm verbundenen Online-Shops und kann diese Besuche somit den einzelnen Benutzern zuordnen. Stellt der Tracking-Anbieter diese Informationen dem Online-Shop zur Verfügung, kann dieser auf die Interessen des Besuchers schließen und seinen Online-Shop entsprechend anpassen („personalisieren“).

# Cookies



## Vue-Plugin: vue-cookies

- Das npm-Package **vue-cookies** ist ein einfaches Vue-Plugin, um mit Browser-Cookies umzugehen
- Installation: `npm install vue-cookies --save`
- Verwendung & Setup:

```
// for e.g. within main.ts

import VueCookies from 'vue-cookies';

Vue.use(VueCookies);

...

// set global default cookie expiration time (5 days)
Vue.$cookies.config('5d');
```

# Cookies

## Verwendung von vue-cookies - Beispiele



```
// usage within any vue component (this context)

// set cookie
this.$cookies.set('myKeyName', 'primitive string value');
this.$cookies.set('myKeyName', {value: 'complex object value'});

// set cookie and override default expire time
this.$cookies.set('myKeyName', 'value', '8h')

// get cookie
this.$cookies.get('myKeyName'); // return cookie value

// remove cookie
this.$cookies.remove('myKeyName');

// check for existing cookie
this.$cookies.isKey('myKeyName'); // return false or true

// get an array of all cookie keys
this.$cookies.keys(); // return array
```