HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN
UNIVERSITY OF APPLIED SCIENCES

Das JavaScript-Framework Vue.js

Prof. Dr.-Ing. Thomas Wiedemann
E-Mail: wiedem@informatik.htw-dresden.de

B. Sc. Vladimir Veblium
M. Sc. Elena Mulear

Fakultät Informatik/Mathematik

Gliederung 1/2

- Historie / aktueller Status
  - Autor(en) und Lizenz
  - Aktuelle Version
  - Vergleich mit anderen JS-Frameworks
- Typische Eigenschaften
  - Dokumentation, Lernkurve
  - Flexibilität, Performance
  - Tooling, Community
- Grundlagen der Arbeit mit Vue
  - Vorbereitung
  - Vue-Instanz und generelle Vue Syntax
  - v- Direktiven und v-on Shorthands
- Code-Beispiel
  - Lagerbestands-App mit Lagerbestandsverwaltung und aktueller Uhrzeit
  - Daten aus einer API holen auf der Basis von JSON-Daten
  - Lifecycle created() und mounted()

Gliederung 2/2

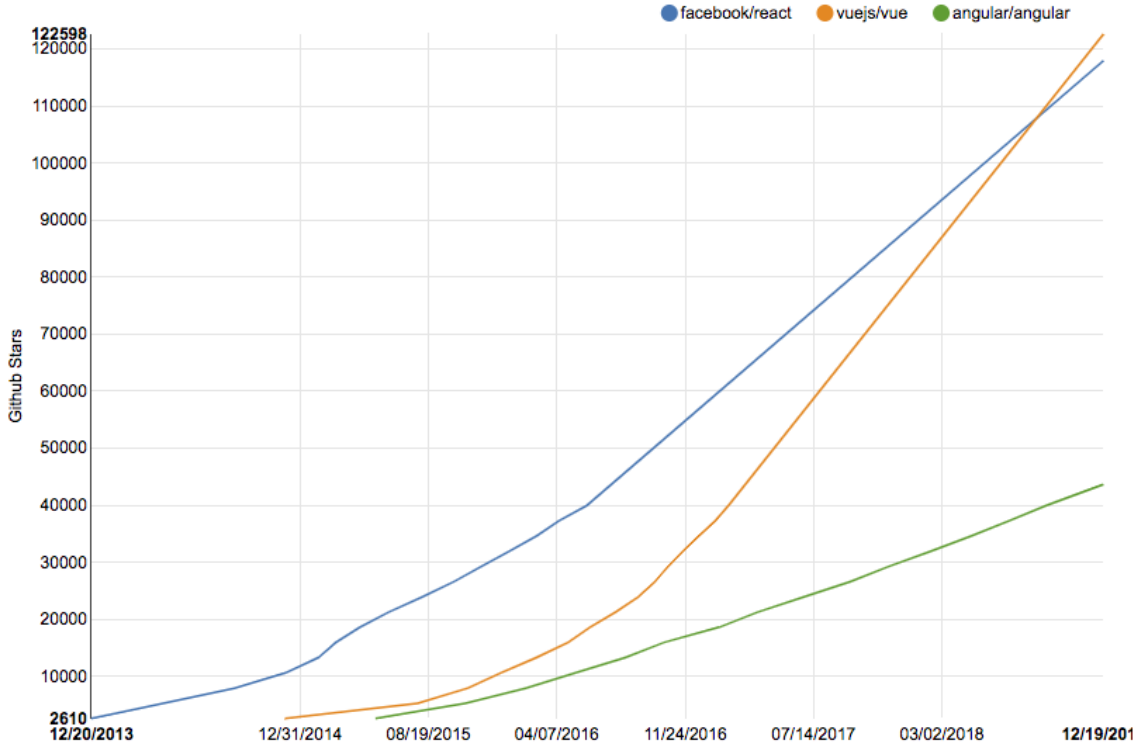
- Vue-Components
  - Einführung mit Components - Beispiel
  - Props und Anwendungseispiel
- Transitions & Animation
  - Einführung
  - Transitions - Beispiel
  - Animation - Beispiel
- Reusability & Weiteres
  - Mixins
  - Weiterführendes (Custom Directives, Render und JSX, Plugins)
- Vue- Ressourcen und Community
- Ausblick Praktikumsaufgabe
  - Großhändler Bookshop (Architektur, Beschreibung, APIs, Design)

Historie und aktueller Status

- veröffentlicht am 11. Februar 2014 von Evan You (ehemaliger AngularJS Entwickler bei Google) [1][2]
- Lizenz: MIT License (Open Source) [3]
- Vue.js Homepage: **https://vuejs.org/**
- aktuelle stabile Version 2.6.10 (Release: 20.03.2019) [4]
- sehr populär bei GitHub (Stand 2019) [5][6]

Bildquelle: vuejs.org/v2/guide/team.html

React
vs.
Vue
vs.
Angular



Typische Eigenschaften von Vue

- Eine sehr einsteigerfreundlich gehaltene Dokumentation [7] mit zahlreichen Best-Praxis-Beispielen sowie einer sehr übersichtlichen API [8]
- Für den Einstieg werden nur JavaScript, HTML und CSS Kenntnisse benötigt (kein TypoScript, JSX o.ä.)
- ein neuer,, besserer Mix von bekannten Technologien aus Angular und React mit einem MVVM (Model View ViewModel) und Virtual DOM, was deutlich leichtgewichtiger und weniger komplex ist [9]
- kann jederzeit in ein vorhandene Projekt integriert und mit anderen Frameworks kombiniert werden
- performanter als Angular und React, weil es sehr schlank ist [10]
- optional mächtiges Vue-Comand-Line-Interface (CLI) mit zahlreichen Features, welches das Entwickeln von Webapplikationen stark vereinfacht [11] [12], jedoch nicht zwingend notwendig ist
- Eine sehr aktive und hilfsbereite Community im Forum, erreichbar über die offizielle Vue.js Webseite [13] - https://forum.vuejs.org/

Grundlagen: Vorbereitung

- Voraussetzungen: nur HTML, CSS und JS Kenntnisse
- **HTML-Datei** *index.html* mit einem `<div>` Tag und einer `id` zum Mounten der Vue-Instanz:

```
<div id="app"></div>
```

- **JavaScript-Datei** *main.js* mit der Vue-Instanz mit Optionen (Daten, Methoden, …):

```
var app = new Vue({
    el: "#app",
    data: { //... },
    methods: { //... }
});
```

- Einbinden des Vue.js Frameworks und der Vue-Instanz mittels des `<script>` Tags in der *index.html* Datei:

```
<script src="https://unpkg.com/vue"></script>
<script src="./main.js"></script>
```

Grundlagen: Vue Instanz

- Die **Vue-Instanz** bildet die Wurzel der Webapplikation und ermöglicht das Speichern von Daten sowie das Ausführen von verschiedenen Aktionen:

```
var app = new Vue({options})
```

- Verknüpfung mit einem DOM-**Element** (=„el“) mittels `el` in der Vue-Instanz durch Bindung über das **`id`** -Attribut aus dem HTML Dokument:

```
el: "#app",
```

- Daten in der Vue-Instanz speichern, z.B.:

```
data: {
  product: "Buch"
},
```

- Daten aus der Vue-Instanz an HTML mittels „Mustache“-Syntax übertragen (sogenannte Text Interpolation) :

```
<div id="app">
    <h1>{{product}}</h1>
</div>
```

- Java-Script-Ausdrücke können innerhalb der Mustache-Syntax verwendet werden, wie z.B. :

```
{{ number + 1 }}
{{ result ? 'Ja' : 'Nein' }}
{{ message.split('').reverse().join('') }}
```

- Zum Übertragen von HTML-Attributen aus der Vue-Instanz wird die **v-bind** -Direktive verwendet (Mustaches können hier nicht verwendet werden):

html-Datei:

```
<h1>{{product}}</h1>
<img v-bind:src="myimage" width="150px"/>
```

js-Datei:

```
data: {
  product: "Buch",
  myimage: "img/book.PNG"
},
```

Grundlagen: v- Direktiven

- Die Direktive **v-if** erlaubt das Überprüfen einer Bedingung innerhalb eines HTML-Tags. Kann verwendet werden um z.B. ein Element einzublenden oder auszublenden:

```
<p v-if="status">Wieder sichtbar</p>
```

- DOM-Event-Listener können mittels der **v-on** Direktive einem HTML-Element zugewiesen werden, z.B. onclick Event:

```
<a v-on:click="myfunction"> ... </a>
```

- Elemente eines Arrays können mittels **v-for** Direktive durchlaufen und in einer HTML-Liste dargestellt werden:

```
<li v-for="book in books">
    {{ book.title }}
</li>
```

- Two-Way-Binding eines Eingabefeldes erfolgt mittels der **v-model** Direktive (bidi. Verknüpfung einer Vue-Variable mit dem Inputfeld) :

```
<input v-model="message" placeholder="Nachricht">
```

Grundlagen: v-bind und v-on Shorthands

- Vue.js bietet Shorthands für die v-bind und v-on Direktiven an:

Full syntax	***Shorthand***	***Shorthand Dynamic arguments (2.6.0+)***
`v-bind:href`	`:href`	`:[key]`

```
z.B.:
<a v-bind:href="url"> ... </a>
<a :href="url"> ... </a>
<a :[key]="url"> ... </a>
```

`v-on:click`	`@click`	`@[event]`

```
z.B.:
<a v-on:click="doSomething"> ... </a>
<a @click="doSomething"> ... </a>
<a @[event]="doSomething"> ... </a>
```

- Mehr Vue.js Grundlagen und Beispiele auf: https://vuejs.org/v2/guide/
- Dynamic Arguments Beispiel: https://openbox.pt/vuejs/vue-js-2-6-dynamic-directive-arguments/

Beispiel: Lagerbestand

- Das Vue.js Beispiel holt Daten aus einer API, welche in einer HTML Liste dargestellt werden
- Die Anzahl der Artikel kann mit entsprechenden Buttons verändert werden (die App verhindert das Einstellen einer Anzahl kleiner 0)
- Bei einer Anzahl von 0 wird ein Hinweis eingeblendet, dass der Artikel ausverkauft ist!
- In der App wird das aktuelle Datum mit Uhrzeit angezeigt

Beispiel: Daten aus einer API holen

- Anlegen der Vue Instanz mit Variablen und Methoden zum Holen von JSON-Daten aus einer API von einem externen Server

```
1  var app = new Vue({
2      el: "#app",
3      data: {
4          products: [],
5          timestamp : ""
6      },
7      methods: {
8          fetchData(){
9              fetch("https://api.myjson.com/bins/74163")
10             .then(response => response.json())
11             .then((data) => {
12               this.products = data.products;
13             })
14         },
```

Beispiel: JSON-Datenstruktur

- Die Daten liegen im JSON-Format auf dem Server als Datei vor und verfügen über drei Attributschlüssel `id`, `quantity` und `name`


```
{
  "products": [
    {
      "id": 1,
      "quantity": 1,
      "name": "Compass"
    },
    {
      "id": 2,
      "quantity": 0,
      "name": "Jacket"
    },
    {
      "id": 3,
      "quantity": 5,
      "name": "Hiking Socks"
    },
    {
      "id": 4,
      "quantity": 2,
      "name": "Suntan Lotion"
    }
  ]
}
```


← Primärschlüssel, auto increment
← Artikelanzahl auf Lager
← Artikelbezeichnung

API URL für ein GET-Request:
https://api.myjson.com/bins/74l63

Beispiel: Lagerbestand erhöhen und verringern

- Anlegen von Methoden zum Erhöhen und Verringern des Lagerbestandes von jedem Artikel aus der Liste
- Das Übergeben der Positionierung (Index) des Artikels im Array geschieht beim Aufruf der Methode in der HTML-Datei in einer `v-for` Schleife

```
15        increase(index){
16            this.products[index].quantity++;
17        },
18        decrease(index){
19            if(this.products[index].quantity > 0)
20                this.products[index].quantity--;
21        },
```

```
<li class="list-group-item" v-for="(product, index) in products">
  Produkt: "{{ product.name }}" ,
  Anzahl:
  <span class="badge badge-primary badge-pill">{{ product.quantity }}</span>
  <button class="btn btn-outline-success ml-3" @click=increase(index)>+</button>
  <button class="btn btn-outline-danger" @click=decrease(index)>-</button>
  <span style="color:red" v-if="product.quantity == 0">Ausverkauft!</span>
</li>
```

Beispiel: Aktuelles Datum und Uhrzeit

- Erstellen einer eigenen Funktion zum Ermitteln des Datums und der aktuellen Uhrzeit im gewünschten Datenformat
- Das Datum und die Uhrzeit werden dem definierten Property `timestamp` zugewiesen, welches in der HTML-Datei final angezeigt wird

```
22          now(){
23              const today = new Date();
24              const date = today.getDate()
25                              + '.'+(today.getMonth()+1)
26                              + '.'+today.getFullYear();
27              const time = today.getHours()
28                              + ":" + today.getMinutes()
29                              + ":" + today.getSeconds();
30              const dateTime = "Datum: " + date +' Uhrzeit: '+ time;
31              this.timestamp  = dateTime;
32          }
```

```
<div id="app">
    <h1 class="display-4">Vue.js Beispiel</h1>
     <h2>{{ timestamp }}</h2>
```

Beispiel: Lifecycle created() und mounted()

- Das Datum und Uhrzeit müssen jede Sekunde aktualisiert werden und die API Daten müssen asynchron in die Webanwendung geladen werden
- Während für reaktive synchrone Operationen die `created()` Vue-Funktion (sogenannter Hook) verwendet wird, sollte für asynchrone Abfragen die `mounted()` Vue-Funktion benutzt werden, welche auf asynchrone Requests wartet [14]
- Andere Lifecycle-Hooks sind [15]: `beforeCreate, created, beforeMount, mounted, beforeUpdate, updated, activated, deactivated, beforeDestroy, destroyed`

```
34      created () {
35              setInterval(this.now, 1000);
36      },
37      mounted () {
38              this.fetchData();
39      },
40  });
```

Vue-Components

- *Allgemein (und unabhängig von Vue) gilt*: Bei komplexeren Webanwendungen oder der Erstellung vieler ähnlicher , aber nicht gleicher Applikationen ist die Arbeit mit **wiederverwendbaren Softwarekomponenten** sinnvoll.

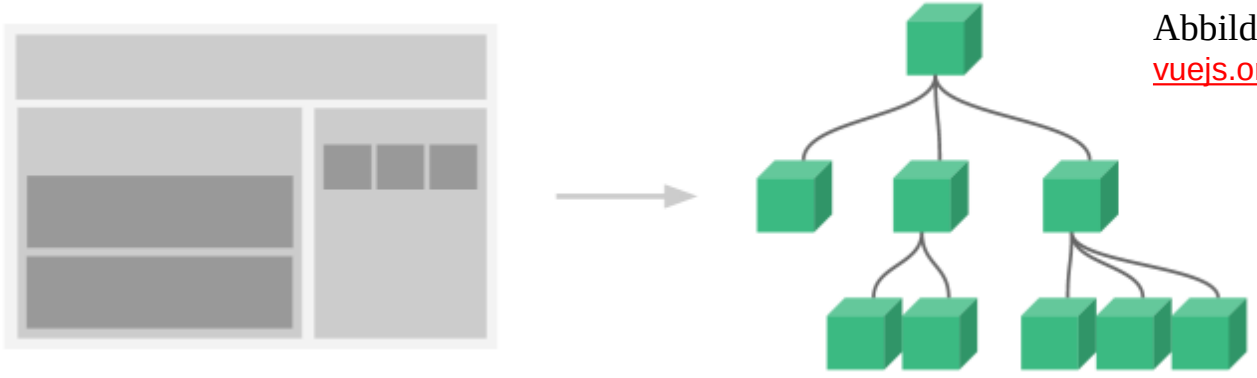


Abbildung Quelle und Details dazu:
vuejs.org/v2/guide/components.html

- Components sind wiederverwendbare Vue-Instanzen mit einem Namen
- sie beinhalten die Funktionalität einer Vue-Anwendung z.B. ein Navigationsmenü, eine Schaltfläche oder ein formatierter Text
- Vue-Components werden mittels der Funktion `component()` definiert und verfügen i.d.R. über eine `data` Funktion und ein `template` Attribut
- Eine definierte Komponente kann als ein Custom-HTML Element beliebig oft im HTML Code wiederverwendet werden

Components - Beispiel

Im folgenden Beispiel wird eine Komponente mit der Bezeichnung ***button-counter*** definiert, welche beim Klick auf den Button den Zähler hochzählt und als ein Custom-HTML Element verwendet wird:

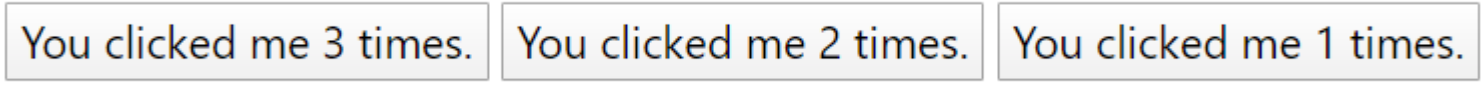
js-Datei:

```
Vue.component('button-counter', {
  data: function () {
    return { count: 0 }},
  template:
  '<button v-on:click="count++">You clicked me {{ count }} times.</button>'
})

new Vue({ el: '#app' })
```

html-Datei:

```
<div id="app">
  <button-counter></button-counter>
  <button-counter></button-counter>
  <button-counter></button-counter>
</div>
```

Browser:



Components - Props

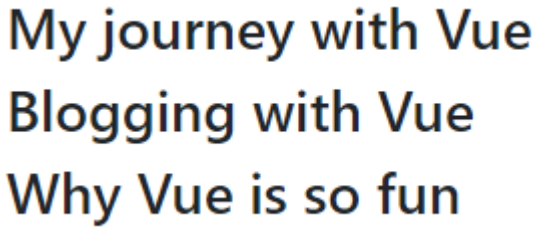
- Props sind Custom-Attribute eines Components
- Nach der Registrierung von Props können Daten an ein Component übergeben werden
- Im Folgenden Beispiel werden verschiedene Titel an ein neu definiertes Component in der HTML-Datei übergeben:

```
Vue.component('blog-post', {
  props: ['title'],
  template: '<h3>{{ title }}</h3>'
})
```

js-Datei

```
<blog-post title="My journey with Vue"></blog-post>
<blog-post title="Blogging with Vue"></blog-post>
<blog-post title="Why Vue is so fun"></blog-post>
```

html-Datei



Browser

Mehr auf: https://vuejs.org/v2/guide/components.html

Transitions

- Ermöglichen Übergangseffekte (transitions) bei DOM-Manipulationen (Hinzufügen, Aktualisieren und Entfernen von DOM-Elementen)
- können zum Beispiel verwendet werden, wenn ein Element abhängig von der Bedingung ein- oder ausgeblendet bzw. angezeigt wird
- Vue.js stellt 6 Transition-Klassen mit dem `v-`Präfix (default) zur Steuerung von Übergängen (Element entfernen/hinzufügen) zur Verfügung
- Diese Klassen sind: `v-enter`, `v-enter-active`, `v-enter-to`, `v-leave`, `v-leave-active`, `v-leave-to`

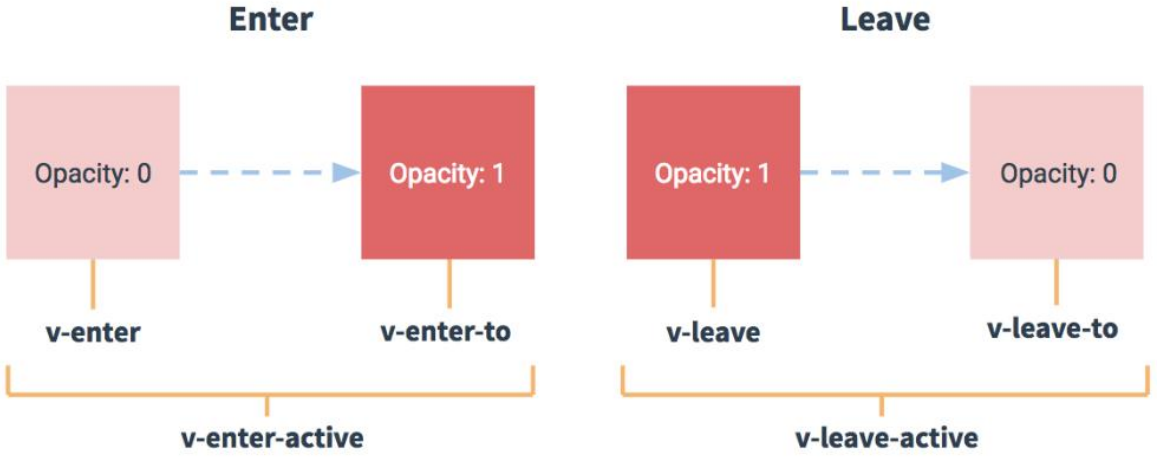


Abbildung: https://vuejs.org/v2/guide/transitions.html#Transition-Classes

Transitions - Beispiel

- Im folgenden Beispiel wird eine einfache Transition für Ein- und Ausblenden eines Elements mittels `v-if` und `v-show` und Transition-Klassen demonstriert:

html-Datei:

```
<div id="app">
  <button v-on:click="show = !show">
    Toggle
  </button>
  <transition name="fade">
    <p v-if="show">hello</p>
  </transition>
</div>
```

js-Datei:

```
new Vue({
  el: '#app',
  data: {
    show: true
  }
})
```

css-Datei:

```
.fade-enter-active, .fade-leave-active {
  transition: opacity .5s;
}
.fade-enter, .fade-leave-to {
  opacity: 0;
}
```

Browser:

Mehr auf: https://vuejs.org/v2/guide/transitions.html

Animations - Beispiel

- CSS Animationen werden genau so wie CSS Transitions verwendet, mit dem Unterschied, dass das eingefügte Element animiert wird (Bewegung)
- Das vorherige Beispiel könnte folgendermaßen erweitert werden:

```
<transition name="bounce">
  <p v-if="show">Lorem ipsum dolor sit amet</p>
</transition>
```

html-Datei

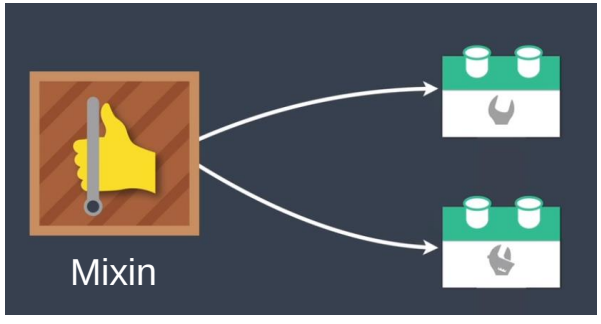
```
.bounce-enter-active {
  animation: bounce-in .5s;
}
.bounce-leave-active {
  animation: bounce-in .5s reverse;
}
@keyframes bounce-in {
  0% {
    transform: scale(0);
  }
  50% {
    transform: scale(1.5);
  }
  100% {
    transform: scale(1);
  }
}
```

css-Datei

Mehr auf: https://vuejs.org/v2/guide/transitions.html#CSS-Animations

Mixins - Beispiel

- Verschiedene Komponenten können ähnlich funktionieren oder an einigen Stellen die selbe Funktionalität beinhalten
- Mixins erlauben eine Funktionalität zwischen verschiedenen Komponenten zu teilen
- Ein definiertes Mixin-Objekt kann in einer Komponente verwendet werden



Komponente A

Komponente B

```
// Mixin Objekt wird definiert
var myMixin = {
  created: function () {
    this.hello()
  },
  methods: {
    hello: function () {
      console.log('Ich bin Mixin!')
    }
  }
}
```

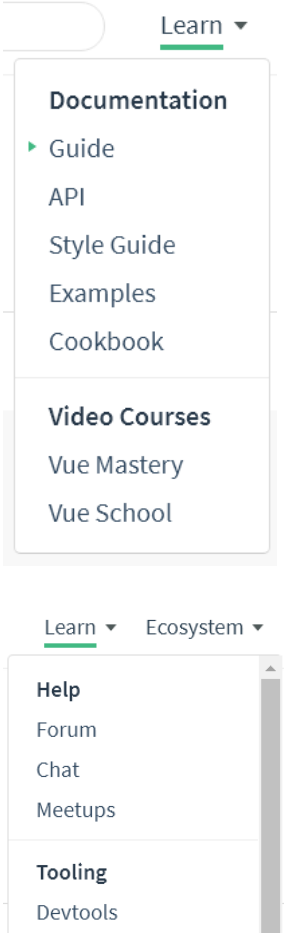
```
// Eine Komponente nutzt das Mixin Objekt
var Component = Vue.extend({
  mixins: [myMixin]
})

var component = new Component()
// => "Ausgabe: Ich bin Mixin!"
```

Mehr dazu: https://vuejs.org/v2/guide/mixins.html

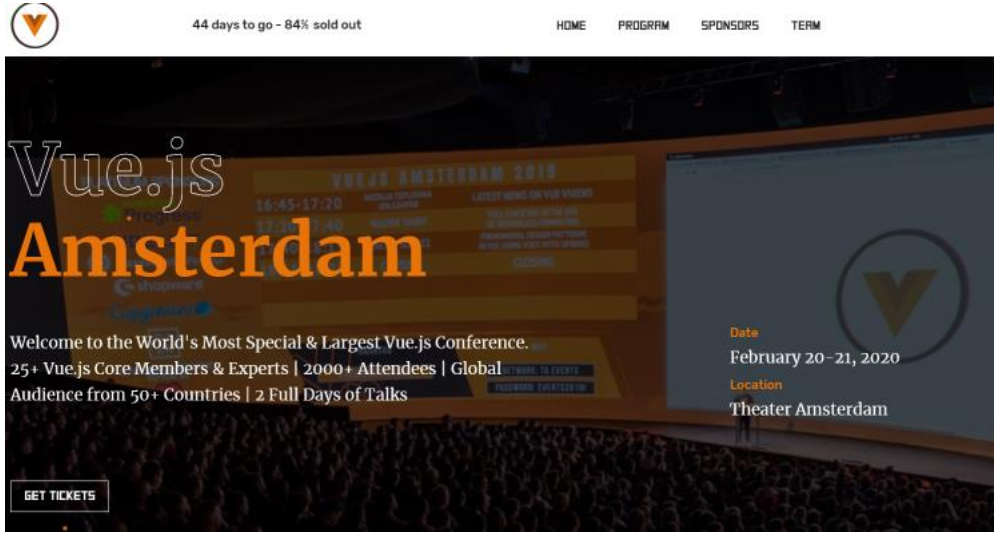
- Zu den bereits vorhandenen Direktiven (`v-model`, `v-show`) erlaubt es Vue, eigene benutzerdefinierte Direktiven zu definieren und zu registrieren, welche mittels `Vue.directive('name')` Anweisung erfolgt (mehr auf: https://vuejs.org/v2/guide/custom-directive.html)
- Es wird empfohlen Templates im HTML Code zu verwenden, jedoch kann es in bestimmten Situationen gewünscht sein, stattdessen eine `render`-Funktionen zu verwenden `render: function (createElement)`

  Mittels **JSX** (syntakt. Erweiterung von JS) kann der Code weiter reduziert werden (mehr dazu auf: https://vuejs.org/v2/guide/render-function.html)
- Mithilfe von Plugins können Vue Funktionalitäten global erweitert werden. Die Verwendung eines Plugins erfolgt mit `Vue.use(MyPlugin)`
  (mehr dazu: https://vuejs.org/v2/guide/plugins.html)
- mehr weiterführende Themen sind in der offiziellen Dokumentation unter https://vuejs.org/v2/guide/ zu finden
- Alle bisher gezeigten Vue-Technologien sind reine Frontend-Technologien allein zur Verwendung im Browser und i.d.R. auch unabhängig von einem Server lauffähig – in Kombination mit weiteren Frameworks wie Nuxt sind komplette Client-Server Applikationen auf JS- und Vue-Basis umsetzbar (=>NodeJS-VL)

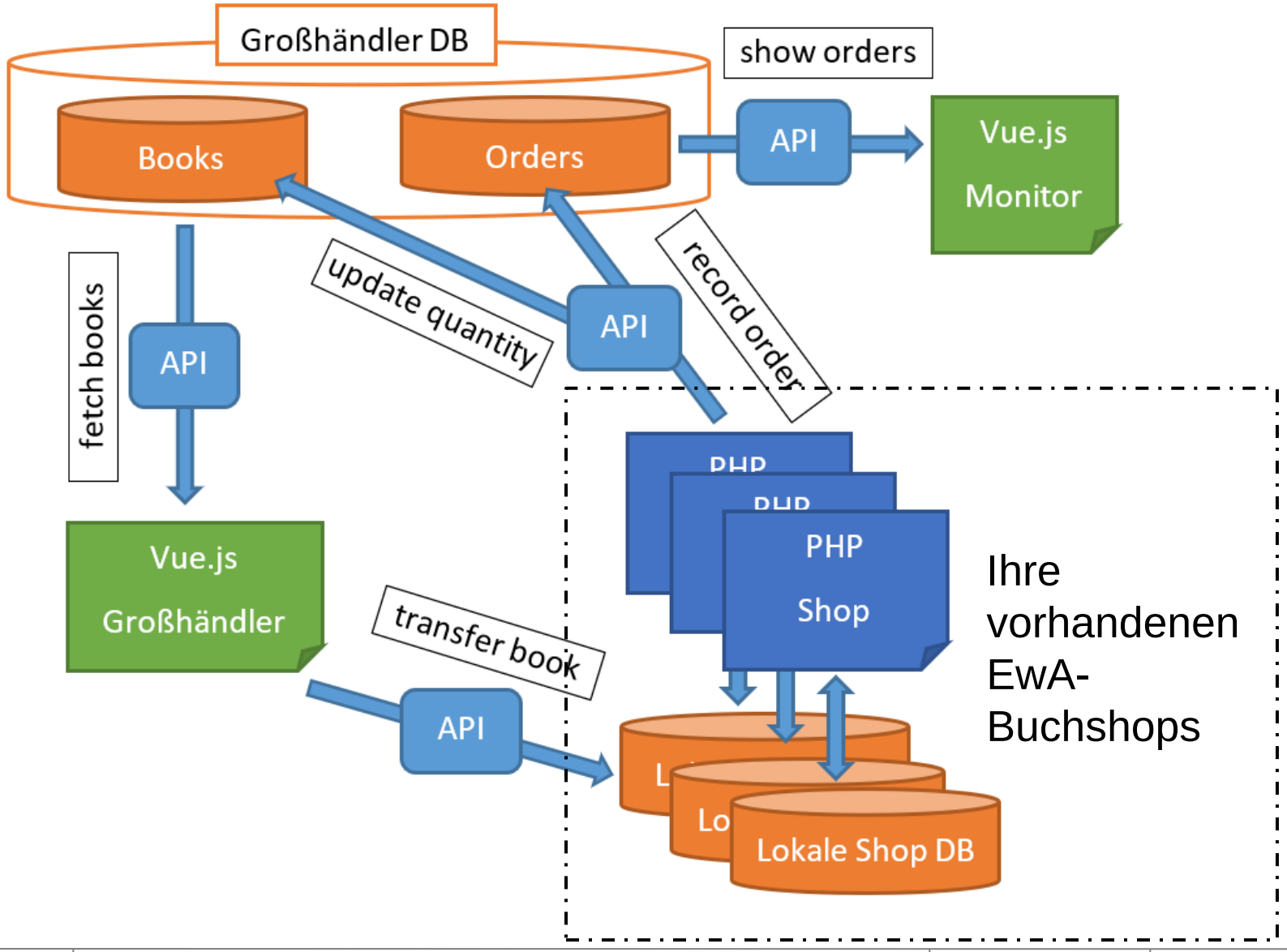
Vue- Ressourcen und Community



- sehr gute und breit angelegte Dokumentationen und Video-Kurse
- sehr ausführlich und gut dokumentierte API
- „Style Guide“, „Examples“ und „Cookbook“ mit sehr vielen Praxisbeispielen
- viele sofort testbare Beispiel auf https://jsfiddle.net/
- sehr gutes Forum + Chat + Meetups
- viele zusätzliche Tools
- größte JavaScript-Konferenz mit ca. 4000 Teilnehmern jährlich im Februar in Amsterdam - https://vuejs.amsterdam/



Ausblick Praktikumsaufgabe: Architektur



Quellen 1/2

- [1]: https://blog.evanyou.me/2014/02/11/first-week-of-launching-an-oss-project/
- [2]: https://github.com/customer-stories/yyx990803
- [3]: https://github.com/vuejs/vue/blob/dev/LICENSE
- [4]: https://github.com/vuejs/vue/releases
- [5]: https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190
- [6]: https://camo.githubusercontent.com/6d4cc486ddc54a14ef24c19549203040716dac4a/68747470733a2f2f692e696d6775722e636f6d2f596b55694f76772e706e67
- [7]: https://vuejs.org/v2/guide/
- [8]: https://vuejs.org/v2/api/
- [9]: https://javascript-days.de/blog/vue-js-ein-mix-aus-angular-und-react-mit-mvvm-und-virtual-dom/

Quellen 2/2

- [10]: https://rawgit.com/krausest/js-framework-benchmark/master/webdriver-ts/table.html
- [11]: https://cli.vuejs.org/guide/
- [12]: https://blog.doubleslash.de/vue-js-und-die-vue-cli-3-fuer-beginner/
- [13]: https://forum.vuejs.org/c/help
- [14]: https://lavalite.org/blog/created-and-mountedin-vuejs
- [15]: https://vuejs.org/v2/api/#Options-Lifecycle-Hooks
- [16]: https://vuejs.org/v2/examples/modal.html
- [17]: https://vuejs.org/v2/guide/computed.html#Computed-Properties