Umsetzung des Matrix-Clients

Vorstellung der Frameworks
Vue, Svelte und React

Vue.js

Lensch, Jannes

Überblick

- “Leichtgewichtiges” Framework zum Entwickeln grafischer Oberflächen für Webanwendungen
- von kleinen Team entwickelt.
- Inspiriert durch Angular:

  I figured, what if I could just extract the part that I really liked about Angular and build something really lightweight. - Evan You

Gründer Evan You

Single-File Components (SFC)

.vue-Dateien

- enthalten <template>, <script> und <style>
- durch SFCs können große Anwendungen in kleine Komponente unterteilt werden
  -> separation of concerns

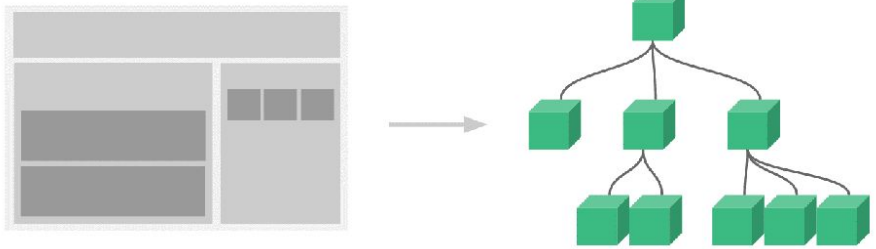
Hello.vue

```vue
<template>
  <p>{{ greeting }} World!</p>
</template>

<script>
module.exports = {
  data: function () {
    return {
      greeting: 'Hello'
    }
  }
}
</script>

<style scoped>
p {
  font-size: 2em;
  text-align: center;
}
</style>
```

Line 21, Column 1 Spaces: 2 Vue Component



Lensch, Jannes

Template

- nicht JavaScript im Mittelpunkt, sondern HTML.
- HTML Grundgerüst wird durch Attribute erweitert, die das rendern steuern: v-for, v-if, v-on und v-bind.
- Zusätzlich Eventhandler, wie @click.

```
<ul>
  <li v-for="product in products">
    {{product.name}}: {{product.quantity}}
     <button @click="product.quantity += 1">+</button>
     <button @click="product.quantity -= 1">-</button>
     <span v-if=”product.quantity < 5”>{{ product.name }} werden knapp!</span>
  </li>
</ul>
```

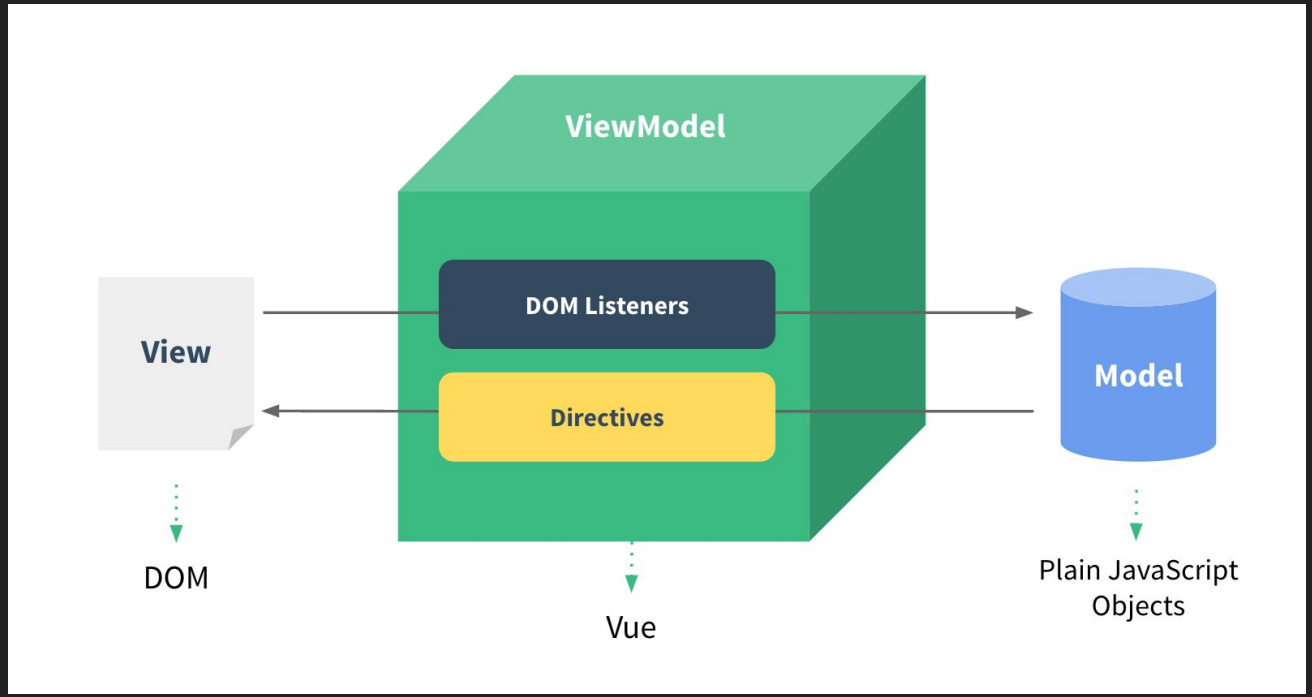
Code Beispiel:

```
<script>
export default {
  data() {
    return {
      count: 0
    }
  }
}
</script>

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

<style scoped>
button {
  font-weight: bold;
}
</style>
```

MVVM-Pattern



https://012.vuejs.org/guide/

Vor- und Nachteile - Vue.js

Vorteile:

- leicht zu verstehen und zu erlernen
- leichtgewichtig
- gute Dokumentation
- Flexibilität
- gute Performance
- Tool System

Nachteile:

- kleiner im Vergleich zu React oder Angular -> weniger Ressourcen
- (oft für kleinere Projekte genutzt)
- zu viel Flexibilität
  - verschiedene Programmier Ansätze könnten zu Inkonsistenz und Problemen führen
- relativ jung -> Mangel an Erfahrung

Krüger, Lukas M.

Überblick

- Entwickelt 2016 von Rich Harris
- Ziel: leichtgewichtig und einfach sein
- Javascript-Framework zur Erstellung von SPA
- Unterstützt auch TypeScript
- Verwendet keine Virtual DOM und nur natives Javascript
  => geringere bundle size als z.B. React/Angular
  => “bessere” Performance
- Von Stackoverflow zum “meist geliebten” Framework gewählt

Single File Components (.svelte)

- Funktion ähnlich vue.js
- Script, Styles und HTML-Body in .svelte
- nur <script> und <style> müssen deklariert werden
- Reaktivität **nur** bei Wertzuweisungen
- “Svelte-Funktionalitäten“ über { } in HTML-Tags

```
1  <!--Body -->
2  <script>
3      export var number = 0;
4  </script>
5  <!--Body-->
6  <style>
7      h1 { color: red}
8  </style>
9  <!--Body-->
10 <h1>{number}</h1>
```

Single File Components (.svelte)

- Schleifen und Abfragenlogik im HTML-Möglich
- einleiten mit **#**, fortsetzen mit **:** und beenden mit */*
- event-Listener ähnlich nativem HTML
- per **bind:***wert* auch two-way-binding möglich

```
<ul>
    {#each testelems as elem}
    {#if elem.number < 10}
    <li on:click="{setPrev(elem)}" >{elem.title}
    {:else if elem.number > 50 }
    <li on:click="{setPrev(elem)}" >{elem.title}
    {:else}
    <li on:click="{setPrev(elem)}" >{elem.title}
    {/if}
    {/each}
</ul>
```

Single File Components (.svelte)

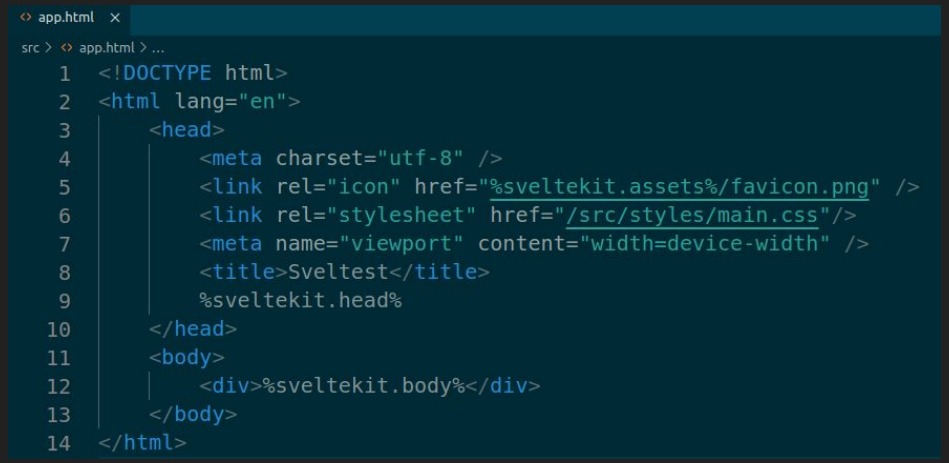
- Einfügen von Components *nach* importieren möglich.
- exportierte Werte können “mitgegeben” werden
- Lifecycle-Hooks müssen von ‘svelte’ importiert werden
  z.B: onMount() onDestroy()
- Auch übergänge/animationen werden von svelte angeboten.

```
import { onMount } from 'svelte'
import { fade } from 'svelte/transition'
import ShowNumber from './shownumber.svelte'
```

```
{#if (prev !== undefined) && visible}
<div transition:fade class = "half">
    {prev.title} contains
    <ShowNumber number={prev.number}/>
</div>
{/if}
```

Weitere Besonderheiten

- App.html ist quasi Container für die Components
- CSS ist i.d.R. auf Components gescoped, CSS in app.html ist jedoch übergreifend.
- Components können über <svelte:head> den <head> verändern, so z.B. den Titel.
- Svelte enthält standardmäßig zwar viele Funktionen, jedoch funktionieren auch viele JS-Bibliotheken mit Svelte.



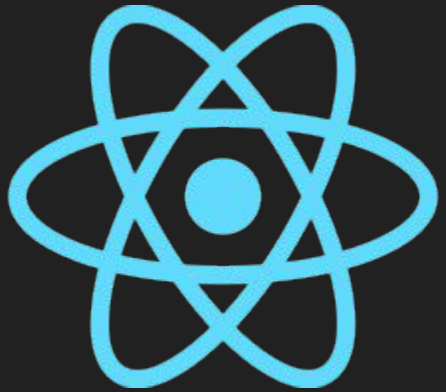
Vor- und Nachteile

Vorteile

- Lightweight
- relativ einfacher Syntax
- standardmäßig viel funktionalität
- TypeScript oder JavaScript möglich
- gute Dokumentation und interaktive Tutorials auf svelte.dev

Nachteile

- kleine Nutzerbasis
- in Unternehmen noch nicht so verbreitet wie z.B. React oder Vue
- kleineres Ökosystem als Vue und React
- SvelteKit ist gerade noch in der Betaphase



React.js

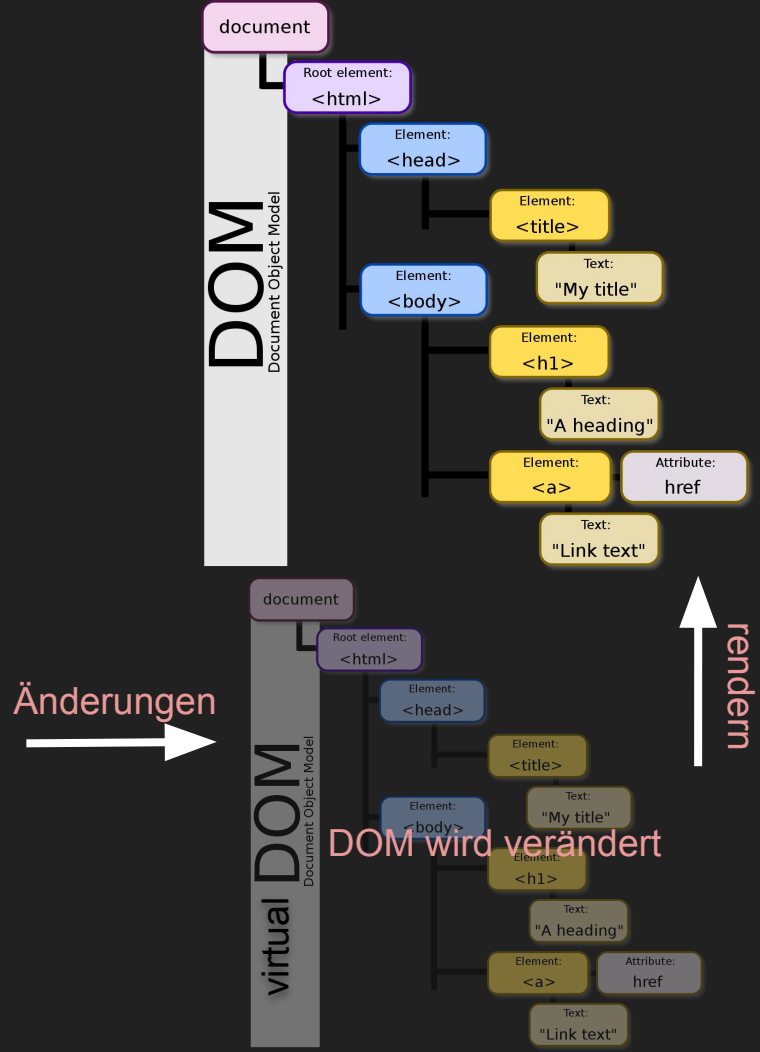
Schwunk, Alexander

Überblick

- entwickelt von Jordan Walke bei Facebook
  - wurde im Facebook Newsfeed und für Instagram verwendet
- 2013 von Facebook als Open-Source-Projekt weitergeführt
- wird üblicherweise für Single-Page-Anwendungen verwendet

Funktionsweise

- parallel zum Browser DOM erzeugt React ein virtuelles DOM
- Änderungen finden erst im virtuellen DOM statt
- React aktualisiert Browser DOM
  - React ändert im Browser DOM nur das, was auch wirklich geändert werden muss



Verwendung & Syntax

- Elemente der grafische Oberfläche werden heruntergebrochen auf einzelne Komponenten
- eine Komponente ist in einer einfachen Javascript-Funktion definiert
- einfaches Beispiel:
  ShoppingList von Mark:

```
class ShoppingList extends React.Component {
  render() {
    return (
    );
  }
}

// Example usage: <ShoppingList name="Mark" />
```

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<h1>Hello, world!</h1>);
```

Schwunk, Alexander

Verwendung & Syntax

- Rückgabe der Funktion in “JSX”
  - in Javascript eingebetteter HTML Code

- React reagiert auf Änderungen
  - Attribut name der ShoppingList ändert sich
  - Wert von {this.props.name} wird aktualisiert
  - virtuelles DOM wird gerendert
  - React aktualisiert den Text des h1-Tags im Browser DOM

```
class ShoppingList extends React.Component {
  render() {
    return (
      <div className="shopping-list">
        <h1>Shopping List for {this.props.name}</h1>
        <ul>
          <li>Instagram</li>
          <li>WhatsApp</li>
          <li>Oculus</li>
        </ul>
      </div>
    );
  }
}

// Example usage: <ShoppingList name="Mark" />
```

Verwendung & Syntax

- Werte ändern mit einem Hook
- Ein Hook beinhaltet einen Wert und eine Funktion zum Ändern des Werts
  - im Beispiel: name und setName

```
function changeName() {
  const [name, setName] = useState("Mark");

  return (




  )
}
```

Verwendung & Syntax

- als Rückgabe wieder mit JSX
  - ein HTML Form für das Eingabefeld
  - value-Attribut für den aktuellen Zustand des Eingabefeldes
  - onChange-Attribut mit dem Funktionsaufruf von setName

- React.js bietet Hooks für viele übliche Anwendungsfälle

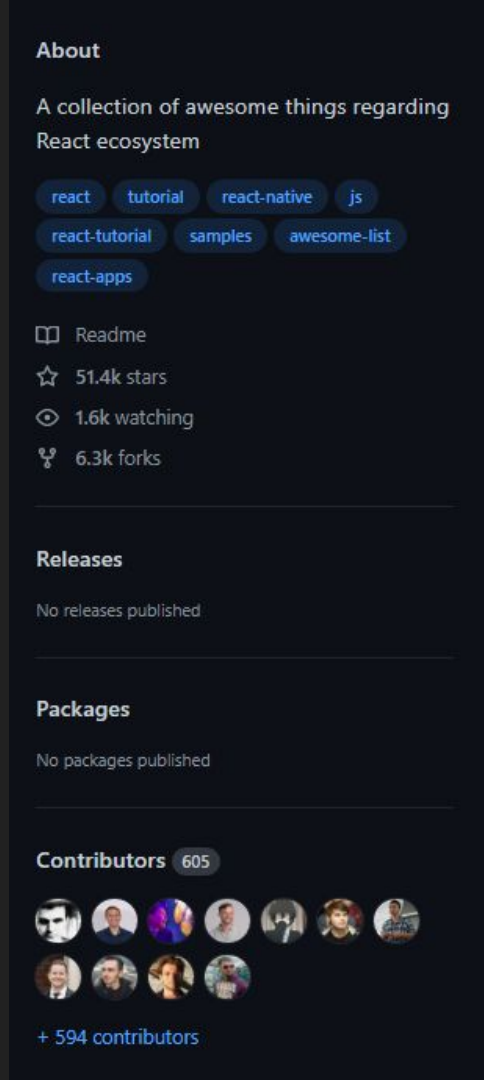
```
function changeName() {
  const [name, setName] = useState("Mark");

  return (
    <form>
      <label>Enter your name:
        <input
          type="text"
          value={name}
          onChange={(e) => setName(e.target.value)}
        />
      </label>
    </form>
  )
}
```

Das React-Ökosystem

github.com/enaqx/awesome-react

- große Open-Source-Community
- sehr viele Bibliotheken die man benutzen darf
  - nimmt Arbeit ab
  - ist wahrscheinlich schöner umgesetzt



Vorteile

- einfach zu lernen
- viele Tutorials und Dokumentationen
- bessere Seitenperformance durch virtual DOM
- Komponenten können während der Entwicklung meist wiederverwendet werden
- einfacher im Testen
- am häufigsten verwendetes JS Framework
  - großer Markt für Entwickler
  - gut für die Weiterentwicklung

Nachteile

- entwickelt sich schnell weiter
  - könnte “häufig” geupdatet werden müssen
- Syntax nicht ganz so simpel wie z.B. Vue
- Entwicklung dauert üblicherweise länger