Fazit Vue-Prototyp

Einleitung

Für eine vollständige Evaluation wurde ein Protoyp mit Vue entwickelt. Eine Version ohne Typescript befindet sich hier und eine mit Typescript ist hier zu finden. Diese verwendet als UI Framework die für Vue nativ entwickelte Element UI. Im folgenden werden die wichtigsten Features und Einzelheiten von Vue.js erläutert. Ähnlich wie React handelt es sich bei Vue.js um kein volles Framework wie Angular sondern nur um eine leichte JavaScript library.

Allgemeines

1. Im Kern von Vue stehen die Single File Components, wo HTML-Templates mit Javascript Funktionen und CSS-Styling verbunden werden. Eine einfache Single File Component könnte zum Beispiel so aussehen:

   SFC

   ```
   <template>
     <p>{{ text }} Single File Komponente!</p>
   </template>

   <script>
   module.exports = {
     data: function () {
       return {
         text: 'Dies ist eine'
       }
     }
   }
   </script>

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

   Somit wird HTML, Logik und das Styling kompakt in einer .vue Datei verwaltet. Der Vorteil liegt hier bei der Lesbarkeit für den Entwickler, da eine Komponente so sehr kompakt gestaltet werden kann.
2. Vue besitzt zudem Two-Way Binding mit `v-model` und `v-bind` . Ein für sich sprechendes Beispiel würde zum Beispiel so aussehen:

   Binding

   ```
   <template>
           <div>
                   <p>{{ message }}</p>
                   <input v-model="message">
           <div>
   </template>

   <script>
   module.exports = {
     data: {
       message: 'Tipp was ein!'
     }
   }
   </script>
   ```

   So kann man Inputs und sonstige Eingabefelder direkt mit den Daten verknüpfen.
3. Desweiteren bietet Vue.js auch directives wie `v-if` und `v-for` womit man Inline Conditional Rendering sehr lesbar realisieren kann:

v-for

```
<ul>
        <li
                v-for = "(item, index) in items"
                v-bind:item = "item"
                v-bind:index = "index"
                v-bind:key = "item.id"
        ></li>
</ul>
```

Hiermit könnte man eine Liste ganz einfach in Vue darstellen.

Performance

1. Ähnlich wie React implementiert Vue.js eine Virtual DOM. Diese ist schlau genug nicht bei jeder Datenänderung die komplette Seite zu updaten. Man benutzt hier die `:key` directive um Vue mitzuteilen, wann ein HTML Element geupdatet werden soll. Das bedeutet, wenn sich der `:key` eines DOM Elements ändert, oder wenn die Daten die an diesem Element gebunden sind sich ändern, weiß Vue.js, dass es die Seite updaten muss. Dadurch werden unnötige rerenderings vermieden und der Entwickler muss sich am Ende nur darum kümmern, den einzelnen Elementen eine einzigarte `:key` directive zu geben.
2. Die Performance einer Web-Applikation hängt des Weiteren an der Größe der einzelnen Komponenten ab. Vue.js ist mit **60 Kb** das kleinste der drei untersuchten Libraries / Frameworks. Eine gute Performance ist somit höchstwahrscheinlich zu erwarten.
3. Außerdem hängt die Performance von der eingebundenen UI-Komponente ab. Beim Prototyp wurde auf Element-UI gesetzt. Jedoch hat sich während der Entwicklung herausgestellt, dass diese UI-Komponente für größere Tabellen und sehr vielen Eingabefeldern nicht geeignet ist. Hier müsste man somit die für sich am Besten geeignete UI-Komponente finden und eventuell auf spezialisierte Komponenten wie den vue-virtual-scroller zugreifen.
4. **Primfaces für Vue (PrimeVue)** ist im Moment noch in der Entwicklung. Eine ausgereifte Version ist wahrscheinlich erst Ende des Jahres zu erwarten.

Zusätzliche Tools und Testing

1. Als Entwicklerumgebung empfiehlt sich entweder Visual Studio Code oder WebStorm. Diese bieten beide IDE Debuggingmöglichkeiten und VS Code hat zudem weitere Plugins wie Vetur welche Aufgaben wie Syntax-Highlighting, Linting / Error-checking und Auto-Completion übernimmt.
2. Als Testing Tools empfiehlt sich hier `vue-test-utils` oder Mocha in Verbindung mit Jest-matchern. Da das Testen sehr ähnlich zum React Testing verläuft wurde für diesen Prototyp keine Beispieltests geschrieben. Jedoch ist der Grundgedanke fürs Testing zwischen Vue.js und React gleich. Die jeweiligen Tools sind somit sehr sehr ähnlich.
3. Für das Testing empfiehlt es sich somit die Tests für React anzusehen, da es hier kaum Unterschiede geben wird.