Handout zum Projekt im Modul Web-Programmierung

Thema: Templating mit MarkoJS

Name: Bennet Albert – MaNr: 7204217

Kurs: WWI-16-SCC

Dozent: Herr Henrich

Datum: 24.07.2018

---

1. Grundlagen und Einführung

1.1 Was ist Templating mit JavaScript?

JavaScript Templating kann als eine Möglichkeit zur Wiederverwendung von UI-Elementen beschrieben werden. Dabei findet eine Trennung zwischen der HTML-Struktur und den anzuzeigenden Daten statt. Beim Templating werden Elemente somit an zentraler Stelle implementiert und können über eine einfache Referenz wiederverwendet werden.

```
<button.example-button on-click('increment')>
    Click me!
</button>
```

Abbildung 1: Erzeugung von einem Button mit Verweis auf die on-click Methode. Wie der Button genau aussieht, wird an anderer Stelle hinterlegt.

Dabei ist zwischen *Inline-Templates* und *External-Templates* zu unterscheiden. Bei External-Templates werden Komponenten nur heruntergeladen, wenn der Client sie referenziert. So wird unnötiger Traffic vermieden. MarkoJS verwendet External-Templates.

Einsatzgebiete für Templating sind besonders Webseiten, bei denen gleiche UI-Bausteine häufig wiederverwendet, allerdings andere Datensätze in der gleichen Formatierung dargestellt werden.

1.2 Was ist MarkoJS und welche Vorteile bietet es?

MarkoJS verwendet eine eigene Syntax um UI-Elemente auf Webseiten darzustellen und Inhalte dynamisch zu verändern. Die grundlegende Funktionalität lässt sich dabei mit „Change the data backing a view and Marko will automatically and efficiently update the resulting HTML“[1] beschreiben. MarkoJS ermöglicht es

[1] Vgl. https://markojs.com/#reactive-ui-components

Templates für UI-Bausteine anzulegen, auf welche an anderer Stelle referenziert werden können, um sie in die Seitenstruktur einzubinden. Das Prinzip der Referenz sorgt für einen schlanken und möglichst wenig redundanten Code. MarkoJS wurde von einem Entwicklerteam von eBay entwickelt und steht als Open-Source Anwendung zur Verfügung.

Die Syntaxt setzt sich aus HTML und JavaScript zusammen und wird daher häufig als HTML-JS-Syntax bezeichnet. JavaScript kann direkt in HTML-Bausteine eingebunden werden und benötigt je nach Funktionalität keinen weiteren Methodenaufruf. Charakteristisch für MarkoJS ist dabei die Verwendung von $ Zeichen, um zugehörige Variablen und Platzhalter zu markieren. In der links stehenden Abbildung ist eine einfache while-Schleife in eine HTML-Liste eingebunden. Würde man die Funktion auslagern, so wird ein zusätzlicher Code notwendig und die Werte müssten per .innerHTML in ein Element der Liste geschrieben werden. Hinzu kommt, dass mehrere Daten-Platzhalter hintereinander geschrieben werden können, um komplexere Inhalte zu realisieren.

```
$ var n = 0;

<ul>
  <li while(n < 4)>
    ${n++}
  </li>
</ul>
```

MarkoJS ist nicht wie eine klassische JavaScript-Bibliothek aufgebaut. Diese tragen häufig die Eigenschaft, dass der gesamte Inhalt der Bibliothek heruntergeladen werden muss. MarkoJS erfordert nur die Einbindung der tatsächlich verwendeten Inhalte, was zu schlankeren Projekten und schnelleren Ladezeiten führen kann.

1.2 Welche Vorteile bietet MarkoJS?

Wie bereits im vorherigen Absatz erwähnt, können mit MarkoJS wiederverwendbare UI-Templates erstellt werden, in welchen Daten mit Hilfe von Platzhaltern dynamisch eingebunden und verändert werden können. Die Wiederverwendbarkeit der Templates sorgt zudem für einen möglichst redundanzfreien und schlanken Code. Jedoch lassen sich noch weitere Vorteile von MarkoJS nennen:

- Verwendung von Custom-Tags, welche vorher definiert werden können. Custom-Tags ermöglichen eigene vordefinierte Inhalte.
- Vereinfachte Syntax durch den optionalen Wegfall einiger HTML-Tags. Die HTML-Struktur wird dabei durch Einrückungen bestimmt.
- Asynchrones Rendering ermöglicht den Aufbau der Seite, auch wenn einige Elemente noch nicht erfolgreich übertragen wurden (Verwendung von JavaScript Promises). Gerade bei mobilen Verbindungen kann dies sinnvoll sein, damit der Nutzer zumindest Teile der Seite angezeigt bekommt.
- MarkoJS Files besitzen nur eine sehr geringe Größe (meist 10% der Filegröße von React).

2. Anwendung in der Flugsuche

2.1 In welchen Bereichen kann MarkoJS angewendet werden?

Ein Anwendungsbeispiel für MarkoJS kann die dynamisch erzeugte Ergebnisliste darstellen. In der ersten Implementierung wird eine klassische unsortierte Liste in einem <ul> und <li> Tag definiert und bekommt durch CSS einen Style zugewiesen. Sollte diese Liste an mehreren Stellen im Projekt vorkommen, so müsste

```
<ul class="FlightSearchResults" id="FlightSearchResult">
    <li class="hidden" id="ResultDummy">
        <span class="airportFrom"></span>&#x27A1<span class="airportTo"></span><br>
        ↗    <span class="Abflug"> </span> ↘    <span class="Ankunft"></span>
    </li>
</ul>
```

```
<Liste.Suchergebnisse id="ListResults">
    <span> ${airportFrom} </span> -> <span> ${airportTo} </span> <br>
    ↗ <span> ${$departure} </span> ↘ <span> ${arrival} </span>
</Liste>
```

sie erneut implementiert werden, was zu doppelten Code führen kann. In der MarkoJS Implementierung wird ein anderer Ansatz verfolgt: Im Template wird eine Liste definiert, welche bestimmten Eigenschaften (unter anderem auch Style-Eigenschaften) erhält. Soll diese erneut verwendet werden, so genügt ein Aufruf über die Referenz. Es wäre theoretisch sogar möglich die Elemente in der Referenz auszulassen, wenn diese bereits im Template hinterlegt sind.

Ein weiterer Anwendungsbereich in der Flugsuche wäre das Verändern UI-Elemente zur Personalisierung, oder in späteren Projektschritten ein Template für weitere Daten zu den Flügen. Zur Personalisierung könnte beispielhaft das Hinzufügen eines Namens über den Suchergebnissen genannt werden. Dabei kann auf der

```
$ var Name = document.getElementById('Namefield').value;

<div id="KopfleisteSuchergebnisse">
    Hier sind die Flüge für ${Name}!
</div>
```

Webseite ein Namensfeld oder eine Login Möglichkeit eingebunden werden, welche für eine „Personalisierung“ der Suchergebnisse verwendet werden kann. Über den Platzhalter kann der eingelesene Name gesetzt werden.

MarkoJS könnte sich auch dazu eignen, um Overlays anhand von Bedingungen einzublenden. Mit Hilfe einer

```
<if(Math.random() > 0.1)>
    <div>
        Du hast einen Flug-Gutschein gewonnen!
        ToDo: Implementiere Overlay mit Gutscheincode
    </div>
</if>
```

math.random() Methode wird dabei eine zufällige Zahl erzeugt. Liegt diese unter 0.1, so kann ein Overlay mit Gutschein angezeigt werden. Eine externe Methode ist nicht notwendig. (Anmerkung: Es handelt sich um beispielhaften Pseudocode)

2.2 Weshalb wurde sich gegen MarkoJS entschieden?

Trotz vieler potentieller Vorteile bei der Verwendung von MarkoJS, wurde sich gegen die Einbindung in das Flugsuche Projekt entschieden. Dies beruht auf mehreren Aspekten:

- Eine Einbindung von MarkoJS ist in bestehende Projekte schwer möglich, da häufig mit eigenen Files (.marko) gearbeitet wird. Dies hätte eine große Umstrukturierung nötig gemacht.
- Der Verbund aus HTML und JavaScript kann aus den oben genannten Gründen als vorteilhaft angesehen werden. Durch die Größe des Projektes macht es momentan aber noch keinen großen Unterschied, ob ein Verbund oder eine Trennung vorliegt.
- Aus persönlichen Präferenzen wird eine Trennung bevorzugt, um eine Übersichtlichkeit zu gewährleisten. (Zudem konnte es lokal aufgrund von Fehlern nicht installiert werden).
- MarkoJS bietet keine Vorteile, welche nicht auch mit gängigen Methoden umgesetzt werden können.

---

3. Fazit

MarkoJS kann eine sinnvolle Erweiterung für die Entwicklung von Webseiten mit starken UI-Fokus darstellen. Besonders wenn viele gleiche Elemente mit unterschiedlichen Daten dargestellt werden müssen, eignen sich Templates um Code Redundanzen zu vermeiden. Doch neben allen Vorteilen die MarkoJS bietet, stellt es in sich keine Funktionalitätserweiterung dar, sondern ermöglicht (nur) ein schnelleres und effizienteres Arbeiten.

Schließlich läuft es auf projektbezogene Eigenschaften hinaus, ob der Einsatz von MarkoJS sinnvoll ist, oder ob auf eine klassische Syntax und Struktur gesetzt werden sollte. Auch persönliche Präferenzen können bei dieser Entscheidung eine große Rolle spielen.

Quellen:

1. https://www.youtube.com/watch?v=i6eZA8Y_GgA
2. https://markojs.com/docs/getting-started/
3. https://www.reddit.com/r/javascript/comments/3yntpt/why_is_markojs_template_engine_not_so_popular/

Logo: https://raw.githubusercontent.com/marko-js/branding/master/marko-logo-medium-cropped.png