Erstellung von adaptiven Web Components

Christoph Kleber

2. November 2016

Kurzfassung

In dieser Arbeit geht es um Web Components.

Abstract

This thesis is about Web Components.

Listing Verzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

API Advanced Programming Interface

DOM Document Object Model

HTML Hypertext Markup Language

URL Uniform Resource Locator

div division

HTML5 Hypertext Markup Language Version 5

CSS Cascading Style Sheets

1 Inhaltsverzeichnis

2 Adaptivität

2.1 Begriffsklärung

Das ist erster Text Test

2.2 Adaptivität bei User Interfaces

3 Web Components

3.1 Was sind Web Components

3.2 Geschichte von Web Components

3.3 Gegenüberstellung Webentwicklung ohne und mit Web Components

3.4 Technik der Web Components

3.4.1 Custom Elements

Das *Custom Element* ist eine *Advanced Programming Interface (API)*, welches das Bilden eigener, voll funktionstüchtiger *DOM* Elemente ermöglicht.[1] Die *API* beschreibt in diesem Zusammenhang eine Schnittstelle, welche einem anderen Programm ein Tool zur Verfügung stellt, um sich an das eigene Softwaresystem anbinden zu können.[2] Somit ermöglicht eine *API* einen Austausch von Informationen zwischen verschiedenen Programmen oder Systemen. Die *Custom Element API* ermöglicht den Nutzern die Auszeichnungsprache *Hypertext Markup Language (HTML)* zu erweitern.[3] Es können bestehende *HTML* Elemente erweitert, oder neue hinzugefügt werden. Jedes neue oder erweiterte Element wird unter einem *Tag* Namen registriert. Dies ermöglicht eine Kapselung des erstellen Programmiercodes in Elemente. In Listing 3.1 ist ein *JavaScript* Programmcode dargestellt, welcher ein leeres *Custom Element* definiert und und unter dem Namen „new-custom-element“ registriert wird. Für *Custom Elements* sind mehrere *Callbacks* verfügbar. *Callback* Funktionen beschrei-

[1] vgl. Denicola. 2016.
[2] vgl. Behrendt. 2016.
[3] vgl. Argelius. 2016.

```
class NewCustomElement extends HTMLElement {
        constructor() {
                super();
        }
}
customElements.define('new-custom-element', NewCustomElement);
```

Listing 3.1: Custom Element JavaScript

ben hier Funktionen, die bei bestimmten Ereignissen des *Lifecycle* von außerhalb des *Custom Elements* aufgerufen werden. Im folgenden werden diese Funktionen aufgelistet.[4]

connectedCallback() Diese Funktion wird aufgerufen wenn das *Custom Element* an den *DOM* angehängt wird.

disconnectedCallback() Diese Funktion wird aufgerufen, wenn das *Custom Element* vom *DOM* wieder losgelöst wird.

attributeChangedCallback(name, prevValue, newValue) Diese Funktion wird aufgerufen, wenn sich ein Attribut ändert. Sie wird jedoch nur für Attribute aufgerufen, welche in einer statischen *get* Funktion mit Namen *observedAttributes* definiert wurden.

3.4.2 HTML Imports

HTML Imports ist eine Technologie zum Importieren von externen HTML Dokumenten in ein HTML Dokument. Hier ist zu unterscheiden zwischen importierenden und importierten HTML Dokumenten. Die importierenden Dokumente besitzen einen Link, welcher mindestens die Uniform Resource Locator (URL) des *Imports* und die Eigenschaft *rel=„import“* besitzt,also ein Link eines bestimmten Typ ist, siehe Listing 3.2.[5] Die importierten Dokumente haben keinen außergewöhnlichen Aufbau im Vergleich zu normalen HTML Dokumenten, sie können aus HTML, *Style* oder *Script* Elementen bestehen. Es kann auch die *Template* Technologie verwendet werden, dazu mehr in 3.4.3. Um auf den Inhalt des importierten Dokuments zuzugreifen wird *JavaScript*

[4] vgl. Argelius. 2016.
[5] vgl. Glazkov und Morrita. 2016.

```
<link rel="import" href="/imports/imported-document.html">
```

Listing 3.2: Standard HTML Import

verwendet. Wie in Listing 3.3 dargestellt, wird zuerst nach dem Link Element gesucht, welches die Eigenschaft *rel=„import“* besitzt. Daraufhin wird dieses Dokument importiert und ein bestimmter Teil des Dokuments als *JavaScript* Variable *„elemt“* gespeichert. Hier wird ein division (div) Element, welches die Klasse *„element“* besitzt gespeichert. Dieses kann dann in der importierenden Seite genutzt werden.

```
var link = document.querySelector('link[rel=import]');
var importedDocument = link.import;
var elem = importedDocument.querySelector('div.element');
```

Listing 3.3: JavaScript Code für Zugriff auf Inhalt des importierten Dokuments

3.4.3 Templates

Das Hypertext Markup Language Version 5 (HTML5) Feature *Templates* ermöglicht Teile einer Seite unabhängig vom DOM zu erstellen. Diese können dann später programmatisch zum DOM hinzugefügt werden.[6] Dies bedeutet, dass der Inhalt des *Templates*, bis er zum DOM hinzugefügt wird, nicht in der Webanwendung angezeigt wird und auch nicht über DOM Selektoren angesteuert werden kann. Gegebenenfalls im *Template* enthaltene Bilder werden nicht geladen und Skripte nicht ausgeführt.[7] In

```
var inhalt = document.querySelector("template").content;
document.querySelector("body").appendChild(inhalt);
```

Listing 3.4: JavaScript Code für dem hinzufügen eines Templates in das DOM

3.4 sieht man den *JavaScript* Code um ein vorhandenes *Template* zum DOM hinzuzufügen. In Zeile eins wird der Inhalt des *Templates* zur *JavaScript* Variable „inhalt“

[6] vgl. Cameron. 2015, S. 177.
[7] vgl. Potschien. 2013.

hinzugefügt, um dann in der nächsten Zeile an den *Body* der Seite, also dem Inhalt hinzugefügt zu werden. In diesem Moment werden auch die Bilder des *Templates* geladen und die Skripte ausgeführt.

3.4.4 Shadow DOM

„Das *Shadow DOM* beschreibt die Fähigkeit eines Browsers, eine neue, völlig abgekapselte Knotenstruktur im bestehenden DOM zu erzeugen.“[8] Dies bedeutet, dass neben dem normalen *Document tree*, wessen Wurzelknoten ein Dokument ist, noch der *Shadow tree* besteht. Der Wurzelknoten des letzteren ist kein Dokument, sondern der *Shadow root*. Dies ist in Abbildung 3.1 dargestellt.[9] Die Folge dieser Kapselung

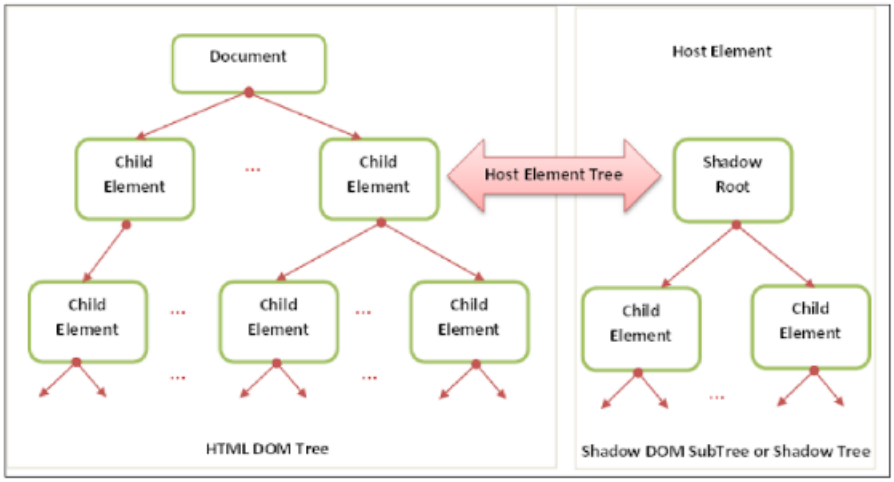

Abbildung 3.1: DOM und Shadow DOM

ist, dass alles was dem *Shadow tree* hinzugefügt wird, nur lokal Einfluss auf diesen hat. Die Gestaltung von Webelementen im *Shadow root* wird dadurch vereinfacht. Cascading Style Sheets (CSS) Selektoren können somit nicht von außerhalb des *Shadow roots* auf dieses Zugreifen und Selektoren, welche innerhalb dieses definiert werden haben keinen Einfluss auf den normalen DOM.[10]

[8] Gasston. 2014, Kap. 11.1.4.
[9] vgl. Patel. 2015, S. 22.
[10] Bidelman. 2016.

4 Methodik dieser Arbeit

5 Adaptive Web Components

5.1 Identifikation passender Web Components

5.1.1 Identifikation Web Components

5.1.2 Identifikation passender Preference Terms

5.2 Preference Sets zur Adaptivität

5.3 Adaptivität der bestehenden Web Components

5.3.1 Web Component Eins

5.3.1.1 Konzeption zur Adaptivität

5.3.1.2 Umsetzung Programmierung

5.3.2 Web Component Zwei

5.3.3 Web Component Drei

6 Vergleich

6.1 Vergleich mit Polymer

Literatur

[1] Andreas Argelius. *Create reusable and future-proof UI components with Custom Elements v1 and Web Components*. 2016. URL: https://onsen.io/blog/create-reusable-future-proof-ui-components-with-custom-elements-v1-web-components/ (besucht am 01. 11. 2016).

[2] Björn Behrendt. *Application-Programming-Interface (API) Definition*. 2016. URL: http://www.gruenderszene.de/lexikon/begriffe/application-programming-interface-api (besucht am 01. 11. 2016).

[3] Eric Bidelman. *Shadow DOM v1: Self-Contained Web Components*. 2016. URL: https://developers.google.com/web/fundamentals/getting-started/primers/shadowdom (besucht am 02. 11. 2016).

[4] Dane Cameron. *HTML5, JavaScript, and jQuery 24-Hour Trainer*. 2015.

[5] Domenic Denicola. *Custom Elements*. 2016. URL: https://www.w3.org/TR/2016/WD-custom-elements-20161013/ (besucht am 01. 11. 2016).

[6] Peter Gasston. *Moderne Webentwicklung: Geräteunabhängige Entwicklung – Techniken und Trends in HTML5, CSS3 und JavaScript*. dpunkt.verlag, 2014. ISBN: 9783864914652.

[7] Dimitri Glazkov und Hajime Morrita. *HTML Imports*. 2016. URL: https://www.w3.org/TR/html-imports/ (besucht am 02. 11. 2016).

[8] Sandeep Kumar Patel. *Learning Web Component Development*. Community experience distilled. Packt Publishing Ltd, 2015. ISBN: 9781784395568.

[9] Denis Potschien. *HTML5: Wie das Template-Element komplexe Vorlagen ermöglicht*. 2013. URL: https://www.drweb.de/magazin/html5-wie-das-template-element-komplexe-html-vorlagen-ermoeglicht-40414/ (besucht am 02. 11. 2016).