IN19 Fachvortrag

Ein Überblick zu Web Components

Dr. Thomas Meinike, Hochschule Merseburg

Motivation

Moderne Webanwendungen sind wesentlich mehr als nur miteinander verknüpfte HTML-Dokumente. Sie bestehen häufig aus einer Vielzahl an Modulen, Bibliotheken und Frameworks. Die Firma Microsoft formulierte bereits 1998 in einer W3C-Note eine Idee zur Kapselung zusammengehörender Einheiten zur Wiederverwendung [1]. Die Implementierung dieser „HTML Components“ in den Internet Explorer 5.5 war jedoch kaum von Bedeutung. Erst ab 2010 wurden im Umfeld der Etablierung von HTML5 neue Ansätze diskutiert und insbesondere durch die Firma Google vorangetrieben. Seit 2014 befinden sich die zur neuen Kategorie „Web Components“ gehörenden Arbeitsentwürfe unter der Obhut des W3C im Standardisierungsprozess [2]. Dieser Beitrag widmet sich den zur praktischen Anwendung erforderlichen Grundlagen.

Standards und Technologien

Zur Realisierung von Web Components werden bereits bekannte Technologien wie HTML5 für die Auszeichnung von Inhalten, CSS3 zur Formatierung und JavaScript als programmatisches Bindeglied eingesetzt. Dieses Fundament wird durch die nachfolgend kurz vorgestellten Zusätze erweitert.

HTML Templates

Das template-Element ist bereits Bestandteil der HTML5-Spezifikation [3]. Es ermöglicht die Ablage von Dokumentteilen, Stylesheets und Skripten zur Wiederverwendung. Inhalte und Programmcode werden vom Browser zunächst nicht gerendert bzw. ausgeführt. Templates können als Schablonen verwendet und bei Bedarf mittels JavaScript inhaltlich weiter angereichert werden.

Custom Elements

Komponentenbasierte Entwicklung wird durch eigene Elemente besonders attraktiv. Es lassen sich über das zur Verfügung stehende HTML-Vokabular hinaus neue Elemente erzeugen. Ihre Namen müssen einen Bindestrich enthalten (etwa „mein-element“) und mittels der Methode registerElement() bekannt gemacht werden. Mit weiteren Anpassungen lässt sich <mein-element> … </mein-element> schließlich wie ein vordefiniertes einsetzen.

Shadow DOM

Die Kommunikation mit HTML-Inhalten und CSS-Eigenschaften sowie das Reagieren auf Benutzerereignisse wird über das standardisierte Document Object Model (DOM) gesteuert. Das zusätzliche Shadow DOM ermöglicht die versteckte und nicht von außen zugreifbare Kapselung von Dokumentfragmenten, Stylesheets und Skriptcode und damit die getrennte Behandlung und Verarbeitung der selbst definierten Elemente. Hier kommt u. a. die Methode createShadowRoot() zum Einsatz, um unterhalb eines Host-Elements den „Schattenbaum“ anzulegen.

HTML Imports

Die genannten Techniken lassen sich zunächst innerhalb eines HTML-Dokuments zum Leben erwecken. Für den praktischen Einsatz ist die Nutzung von extern abgelegten Web Components interessanter. HTML Imports werden mit dem von der CSS-Einbindung her bekannten link-Element möglich. Über dessen href-Attribut wird die jeweilige Komponente als HTML-Datei mit Templates, Stylesheets und Skripten geladen und erst auf Anforderung verarbeitet.

Browserunterstützung

Wie in Abbildung 1 veranschaulicht, werden die genannten Konzepte bisher am besten von Chrome und seinen Chromium-Derivaten unterstützt [4, 5]. Firefox und Safari kennen zumindest Templates. Zudem kann man in Firefox Custom Elements und Shadow DOM mit der Option „dom.webcomponents.enabled | true“ aktivieren. Microsoft arbeitet noch an einer Umsetzung und wird diese künftig in den mit Windows 10 ausgelieferten Edge-Browser integrieren. Mit dem einfach einzubindenden Polyfill [6] lässt sich jeweils fehlende Funktionalität nachrüsten.

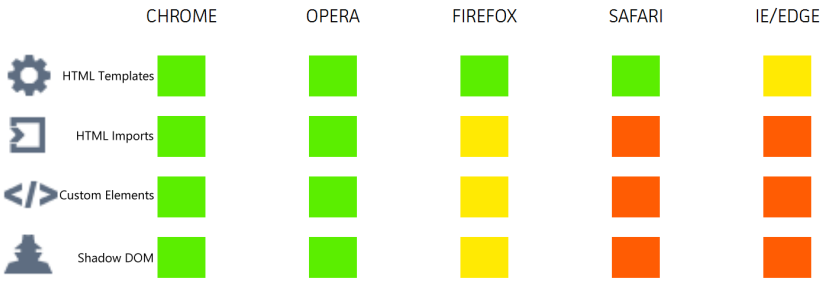

Abb. 1: Unterstützung der Technologien in Browsern (Stand: 08/15) [3]

Praktische Demonstrationen

Zum Verständnis der technologischen Zusammenhänge wurden überschaubare Beispielanwendungen erstellt. Das minimalistisch angelegte „Record-Widget“ nimmt Informationen eines Tonträgers entgegen (Element „tm-record“) und stellt diese entsprechend Abbildung 2 in Form einer kompakten Box dar.

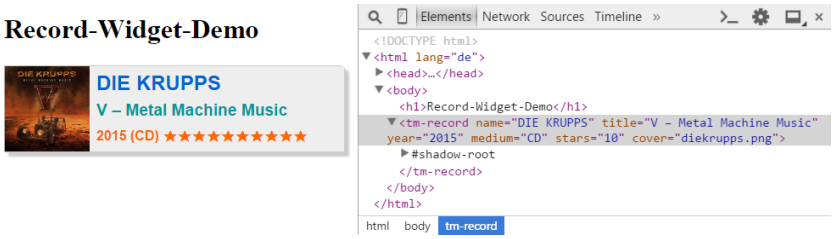

Abb. 2: Beispiel-Komponente „Record-Widget“

Mit dem etwas komplexeren Element „tm-piechart“ wird die vektorgrafische Umsetzung von Daten mit SVG-Techniken demonstriert, siehe Abbildung 3. Unter [7] sind verschiedene Fassungen und Browser-Testergebnisse zugänglich.

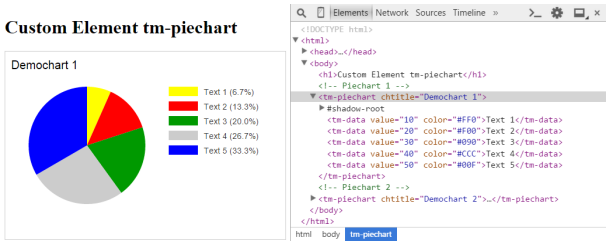

Abb. 3: Beispiel-Komponente „Pie-Chart“

Im Vortrag wird auf die Wirkungsweise des verwendeten Codes eingegangen. Als Denkansatz für den Bereich Technische Kommunikation soll eine Umsetzung von Sicherheitshinweisen dienen.

Fazit und Ausblick

Web Components bieten von der Idee her interessante Möglichkeiten zur Schaffung und Nutzung von wiederverwendbaren Bausteinen. Für die produktive Arbeit lohnt sich auch der Blick auf das Polymer-Projekt von Google [8], das die browserübergreifende Entwicklung unterstützt und bereits zahlreiche vorgefertigte Komponenten anbietet. Es gibt allerdings auch verhalten optimistische und kritische Stimmen [9, 10]. Das Thema bleibt also spannend.

Literaturangaben und Links

[1] W3C: HTML Components – Componentizing Web Applications (Note); http://www.w3.org/TR/1998/NOTE-HTMLComponents-19981023

[2] W3C: Web Components Specifications (Editor’s Drafts); http://w3c.github.io/webcomponents/

[3] W3C: HTML5 Recommendation – The template element; http://www.w3.org/TR/html5/scripting-1.html#the-template-element

[4] WebComponents.org: Browser Support; http://webcomponents.org/

[5] Can I Use: Web Components; http://caniuse.com/#search=web%20components

[6] GitHub: Polyfill webcomponents.js; https://github.com/WebComponents/webcomponentsjs

[7] Meinike, T.: Browser-Tests mit der Komponente tm-piechart; http://datenverdrahten.de/test/tm-piechart/

[8] Polymer Authors: Polymer Library; https://www.polymer-project.org/

[9] Rauschmayer, A.: What happened to Web Components?; http://www.2ality.com/2015/08/web-component-status.html

[10] entwickler.de: Web Components – vom Aussterben bedroht?; https://entwickler.de/online/web/web-components-vom-aussterben-bedroht-165829.html

für Rückfragen:
thomas.meinike@hs-merseburg.de