Cheatography

Webcomponents - native Javscript Cheat Sheet

by Huluvu424242 (FunThomas424242) via cheatography.com/19991/cs/20426/

Grundbegriffe

Vanilla Javascript: Das pure, reine Javascript im Gegensatz zu jQuery, Angular & Co. siehe https://wiki.selfhtml.org/wiki/Vanilla-JS

Vanilla JS Ein Javascript Framework

Webcomponents Spezielle, kleine, autonome auslieferbare Einheiten für Webseiten welche nach dem Webkomponentenstandard des W3C erstellt wurden. Sie können sowohl mit als auch ohne Frameworks erstellt und bereitgestellt werden. Sie basieren auf der Verwendung folgender Standards:

* The Custom Elements specification
* The shadow DOM specification
* The HTML Template specification
* The ES Module specification

Shadow DOM Wird an der Stelle der Webkomponente eingehängt. Meine Vorstellung davon. Der normale DOM ist ein 2D Baum und da wo unsere Komponente eingehängt ist wird ein neuer Baum dran gehängt der nach hinten geht - es wird also ein 3D Baum :)

Der Shadow DOM kann als offen oder geschlossen definiert werden.

Closed Shadow DOM

Kein externes Javascript (also was nicht Teil der Komponente ist) kann auf Elemente des Shadow DOM der Komponente zugreifen. Damit funktionieren diverse Testframeworks nicht bei Webkomponenten. Die Lösung ist: offener Shadow DOM :) Das video Tag wurde per Closed Shadow DOM realisiert.

Open Shadow DOM

Hier kann externes Javascript auf Elemente des Shadow DOM zugreifen. Daher lassen sich dann Webkomponenten testen. Daher ist das aktuell üblich.

Light DOM Ist alles was nicht Shadow DOM ist.

Property Eine Eigenschaft die über Javascript gesetzt wird. Erkennbar daran, dass get und set Definitionen in der Komponente vorliegen.

Attribute Eine per HTML definierte Eigenschaft. Eignet sich nur zur Übergabe von String, Number oder Boolean Werten.

Gold Standard of Webcomponents:

https://github.com/webcomponents/gold-standard/wiki

Eine lebende Checkliste mit der man prüfen kann ob die entwickelte Webcomponente eine gute Qualität besitzt.

Quellen:

* https://www.webcomponents.org/specs
* https://dev.to/thepassle/web-components-from-zero-to-hero-4n-4m#-setting-properties

Vanilla WebComponents Tutorials

* https://dev.to/thepassle/web-components-from-zero-to-hero-4n-4m#-setting-properties
* https://dev.to/bennypowers/lets-build-web-components-part-3-vanilla-components-4on3

Projektstruktur

Projektstruktur (cont)

Es ginge wenn die HTML Imports Spezifikation durchgekommen wäre aber leider hat die Bereitstellung einer Webkomponente über mehrere Dateien, ohne Frameworks nur für kurze Zeit funktioniert: https://www.chromestatus.com/feature/5144752345317376 -> Die HTML Imports Spezifikation gilt als deprecated und wird von den Browserherstellern nicht weiter unterstützt.. **Single File Webcomponents**

Webkomponenten, welche als einzelne Datei realisiert werden, funktionieren aktuell ganz gut. Hat den Vorteil, dass die Komponente auch klein genug bleiben muss, sonst blickt man nicht mehr durch. Eine Art natürlicher, fachlicher Schnitt ;)

Quelle: https://www.chromestatus.com/feature/5144752345317376

Webcomponents - API

customElements.registerElement -> deprecated -> use customElements.define('tag-name', ClassName)

this.createShadowRoot -> deprecated -> use var shadowroot = element.attachShadow(shadowRootInit);

this.shadowRoot -> vererbte Variable in welcher der ShadowDom gespeichert ist

Parameterübergabe

* Per Property: z.B. el.data = myObj;
* Per Attribute: z.B. <my-el data="[{a:1},{a:2}....]"></my-el>
* Per Kindelemente: analog zu selection und options
* Per URL fetch: analog zum src von img

Versuch per Construktor Parameter auszulesen

* Das geht schief wenn unsere Webkomponente durch Javascript erst in den DOM eingehängt wird. Der Grund ist, dass es beim Erstellen noch keine Verbindung zum DOM und damit auch keine Attribute, parent nodes oder childs gibt.
* Das wird klappen wenn Du die Komponente bereits im DOM per HTML definiert hast. Üblicherweise wird hier in der Hostseite ein <template> Tag definiert und dieses im Konstruktur der Webcomponente dann per jQuery gesucht.

Quelle: https://stackoverflow.com/questions/50404970/web-components-pass-data-to-and-from

Properties vs. Attribute

Properties werden per Javascript verändert und in der Komponente müssen getter und setter existieren

Attribute werden per HTML definiert und es können nur Werte vom Typ String, Number und Boolean übergeben werden.

Quelle: https://alligator.io/web-components/attributes-properties/

ES Module

Quelle: https://flaviocopes.com/es-modules/

Styling

Eine Komponente besteht aus Javascript (Verhalten), HTML Template (Struktur) und CSS (Style).

Aufteilung auf mehrere Dateien

Rein theoretisch kann man jede Sprache in einer eigener Datei realisieren. In der Praxis habe ich es nicht hinbekommen die Komponente dann zu laden - da bräuchte es irgendeinen Loader oder so.

Folgende Dinge sind beim Styling zu beachten:

* Display der Komponente ist immer inline wodurch sich Höhe und Breite standardmäßig nicht setzen lassen. Über die Pseudoklasse :host einen Style z.B. flex zuweisen und es geht.

By **Huluvu424242**
(FunThomas424242)

cheatography.com/funthomas424242/
github.com/Huluvu424242

Published 29th August, 2019.
Last updated 7th September, 2019.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com