Web Components: Wiederverwendbarkeit ohne Framework-Grenzen

Sebastian Dorn
virtual7 GmbH
München

Schlüsselworte
Angular, Web Components, Angular Elements, Custom Elements, JavaScript, TypeScript

Einleitung

Portierung auf eine andere Webtechnologie oder mehrere simultan nutzen (war) eine Horrorvorstellung für Projektverantwortliche. Die Bereitstellung von entwickelten Bausteinen für eine andere Webtechnologie war in der Vergangenheit mit großen Anstrengungen verbunden. Mit Web Components gelingt dies heute fast mühelos, wie dieser Vortrag zeigen soll. Web Components ist ein Standard für das Definieren von neunen HTML Elementen auf eine Framework-agnostische Weise. Die eigenen Elemente erweitern die Möglichkeiten in HTML durch die Verwendung selbstdefinierter Tags, welche durch JavaScript erstellt und kontrolliert werden. In einem Fallbeispiel wird exemplarisch gezeigt, wie mit Hilfe von Angular eigene Web Components definiert werden.

Angular

Angular ist ein Open-Source-Projekt von Google. Mit diesem performanten Framework können leistungsstarke und robuste Webanwendungen erstellt werden. TypeScript – eine Obermenge von JavaScript – ist die Standardentwicklungssprache. Entwickler können damit modernen JavaScript-Code schreiben, der in „State of the Art“ Browsern wie Chrome, Firefox und Edge standardmäßig unterstützt wird und dennoch lauffähig in ältere Browser wie Internet Explorer 11 ist. Angular ist eine Plattform für Browser, hybrid-mobile Frameworks, Desktop-Anwendungen und serverseitige gerenderte Ansichten.

Angular unterstützt durch den Komponenten-basierten Aufbau einen leicht lesbaren Programmcode. Denn ein knapper Code ist der Feind der Wartbarkeit und kommt nur dem ursprünglichen Autor zugute. Ausführlicher, entkoppelter, zusammenhängender und gekapselter Code ist der Schlüssel für die zukünftige Überprüfung des Programmcodes. Der Komponenten-basierten Aufbau erleichtert die parallele Entwicklung der Webanwendung und fördert die Wiederverwendbarkeit von Komponenten. Ein Export als Web Components ist seit der Angular Version 6 möglich.

Web Components

Web Components lassen sich mit wiederverwendbare User Interface Widgets veranschaulichen. Web Components nutzen Web-Technologien und sind damit Bestandteil des Browsers. Es sind keine externen Bibliotheken notwendig. Die Einsatzmöglichkeiten sind unbegrenzt und sind nur durch HTML, CSS und JavaScript limitiert. Web Components sind portabel Komponenten und sind wiederverwendbar. Web Components bedient sich bei vier Technologien, die auch einzeln verwendbar sind: Custom Elements, HTML Templates, Shadow DOM & HTML Imports.

Custom Elements kapseln Ihre Funktionalität auf einer HTML-Seite - anstatt mit einem langen verschachtelten Stapel von Elementen auskommen zu müssen, die zusammen eine benutzerdefinierte Seitenfunktion bereitstellen. `CustomElementRegistry` ist der Controller von benutzerdefinierten

Elementen und ermöglicht es Custom Elements zu registrieren oder Informationen über registrierte Elemente zurückzugeben. Für die Registrierung ist der Name des Elements und einem Verhalten bestimmende JavaScript Klasse notwendig.

Das HTML Element `<template>` ist ein Mechanismus für das clientseitiges Vorhalten von Inhalten – ohne dargestellten zu werden. Die Inhalte können zur Laufzeit per JavaScript instanziiert werden.

Ein wichtiger Aspekt von Web Components ist die Kapselung - Struktur, Styles und Verhalten zu verstecken. Mit der Trennung von anderem Code auf der Seite entstehen keine Kollisionen. Der Programmcode ist sauber und leicht lesbar. Die Shadow-DOM-API bietet die Möglichkeit, ein verborgenes, getrenntes DOM an ein Element anzuhängen. Die Shadow-DOM-Struktur beginnt mit einer Wurzel (shadow root), unter dem Sie wie beim normalen DOM beliebige Elemente anfügen können.

Der HTML Import kann ein Bündel von CSS-, JavaScript- und HTML-Code laden. Damit ist es ein hervorragendes Werkzeug zum Laden von eigenständigen Komponenten – wie Web Components.

Angular Elements

Die Unterstützung für benutzerdefinierte Elemente ist enorm. Mit Angular Elements können Sie eine Angular-Komponente codieren und diese Komponente in jeder anderen Webanwendung mithilfe einer beliebigen Webtechnologie wiederverwenden. Dabei wird im Wesentlichen Ihr eigenes benutzerdefiniertes HTML-Element deklariert. Diese benutzerdefinierten Elemente sind mit jeder HTML-basierten Toolchain kompatibel, einschließlich anderer Webanwendungsbibliotheken oder Frameworks.

Durch die Umwandlung einer Komponente in ein benutzerdefiniertes Element ist die gesamte benötigte Angular-Infrastruktur Vorrausetzung für die Lauffähigkeit im Browser. Das Erstellen eines Custom Elements ist einfach und unkompliziert. Es verbindet die komponentendefinierte Sicht automatisch mit Änderungserkennung und Data Binding. Die Angular-Funktionalität wird zu der entsprechenden nativen HTML-Entsprechungen zugeordnet.

So funktioniert es

Durch die Methode `createCustomElement()` wird eine Komponente in eine Klasse konvertiert, die als benutzerdefiniertes Element im Browser registriert werden kann. Nachdem die konfigurierte Klasse in der benutzerdefinierten Elementregistrierung des Browsers registriert ist, kann das neue Element wie ein integriertes HTML-Element verwendet werden, in dem es direkt in das DOM eingefügt wird.

```
<magic-button label="Hello World"></magic-button>
```

Wenn das benutzerdefinierte Element auf einer Seite platziert wird, erstellt der Browser eine Instanz der registrierten Klasse und fügt es dem DOM hinzu. Der Inhalt wird durch das Komponenten Template bereitgestellt, welches die Angular-Syntax verwendet. Mithilfe der Komponenten und DOM-Daten wird der Inhalt gerendert. Eigenschaften und Konfiguration der Komponente werden über entsprechen Eingabeattributen `@Inputs` an das Element übergeben.

Angular stellt die Funktion `createCustomElement()` zum Konvertieren einer Angular-Komponente zusammen mit ihren Abhängigkeiten in ein benutzerdefiniertes Element bereit. Die Funktion sammelt die zu beobachtbaren Properties der Komponente mit der Angular-Funktionalität zusammen, die der Browser benötigt, um Instanzen zu erstellen und zu zerstören sowie Änderungen zu erkennen und darauf zu reagieren.

Der Konvertierungsprozess implementiert die `NgElementConstructor`-Schnittstelle und erstellt eine Klasse, die so konfiguriert ist, dass eine Self-Bootstrapping-Instanz Ihrer Komponente erstellt wird.

Verwendet eine JavaScript-Funktion `customElements.define()`, die den vorkonfigurierten Konstruktor und den zugehörigen benutzerdefiniertes Element-Tag mit der `CustomElementRegistry` Browser registriert. Wenn der Browser das Tag für das registrierte Element findet, verwendet er den Konstruktor zum Erstellen einer benutzerdefinierten Elementinstanz.

Mapping

Ein benutzerdefiniertes Element hostet eine Angular-Komponente und stellt eine Brücke zwischen den Daten und der Logik dar, die in den Komponenten- und Standard-DOM-APIs definiert sind. Komponenten Properties und Logik werden direkt in HTML-Attribute und das Event-System des Browsers abgebildet. Die Property-Namen werden so transformiert, dass diese Benennung mit benutzerdefinierten Elementen kompatibel ist, die keine Unterscheidungen zwischen Groß- und Kleinschreibung erkennen. Die resultierenden Attributnamen verwenden durch Bindestrich getrennte Kleinbuchstaben. Komponenten Outputs werden als HTML Custom Event gesendet, wobei der Name des benutzerdefinierten Ereignisses dem Ausgabenamen entspricht.

Browserunterstützung

Benutzerdefinierte Elemente sind eine Web-Plattform-Funktion, die derzeit Chrome, Opera und Safari unterstützt. In Firefox muss der Benutzer Einstellungen ändern und die Entwickler von Edge arbeiten noch an diesem Feature.

In Browsern, die Custom Elements nativ unterstützen, erfordert die Spezifikation, dass Entwickler ES2015-Klassen verwenden, um benutzerdefinierte Elemente zu definieren. Dafür ist eine Anpassung in der `tsconfig.json`-Datei des Projekts erforderlich, dass die nach der Kompilierung von TypeScript die JavaScript Spezifikation "`es2015`" erfüllt werden soll. Da die Unterstützung für benutzerdefinierte Elemente und ES2015 möglicherweise nicht in allen Browsern verfügbar ist, können Entwickler stattdessen einen Polyfill verwenden, um ältere Browser und ES5-Code zu unterstützen.

Mit der Angular Version 7 wird die Ivy-Rendering-Engine veröffentlicht. Die Basisbündelgrößen reduziert sich damit auf nur 2,7 KB (von 65 KB), was zu blitzschnellen Ladezeiten führt.

Erstellung einer Web Component – mit Angular

Für die Erstellung einer Angular Web Komponenten wird in dem Bespiel die Angular CLI genutzt. Es handelt sich um ein offizielles Angular-Projekt, das sicherstellt, dass neu erstellte Angular-Anwendungen eine einheitliche Architektur aufweisen. Die gesammelten Erfahrungen werden in leicht verwendbaren Best Practices perfektioniert. Die Angular CLI unterstützt den Entwickler durch Code-Generierung bei neunen Komponenten, Direktiven, Pipes, Services, Module und weiteren Bausteinen.

Ein neues Angular Projekt wird mit dem folgenden Befehl in der Konsole angelegt. Die Voraussetzung ist, dass die Angular CLI installiert ist.

```
ng new web-component
cd web-component
ng add @angular/elements –name=web-component
```

Eine Komponente wird angelegt, die am Ende des Beispiels als Web Component bereitgestellt wird.

```
ng generate component magic-button --inline-style --inline-template
```

Die dazu hörigen Codebeispiele sind im Anhang zu finden.

Angular CLI unterstützt den Entwickler auch während der Entwicklungsphase mit Funktionen zum erneuten Laden von Daten, damit schnell die Ergebnisse von Änderungen zu sehen sind. Angular CLI kann auch optimierte Versionen des Codes für eine Produktionsversion testen, filtern und erstellen.

Zum Erstellen des Custom Elements ist das Angular Projekt mit dem folgenden Befehl zu bauen.

```
ng build
```

Durch die Einbindung der erzeugten JavaScript-Dateien kann das benutzerdefinierte Element in einer Webanwendung verwendet werden. Das dazugehörige Beispielcode ist im Anhang zu finden.

Kontaktadresse:
Sebastian Dorn
virtual7 GmbH
Reichenbachstr. 1
80469 München

Telefon:	+49 (0) 151 42 13 67 53
Fax:	+49 (0) 89 122 28 31 20
E-Mail	sebastian.dorn@virtual7.de
Internet:	www.virtual7.de
Podcast:	www.happy-angular.de

Programmcode zu den Beispielen

Die Angular Komponente „Magic Button“, die als Web Component bereitgestellt wird.

```
import {Component, Input, ViewEncapsulation} from '@angular/core';

@Component({
  selector: 'app-magic-button',
  template: `
    <button type="button">{{label}}</button>
    <h1>Hello World</h1>
  `,
  styles: [],
  encapsulation: ViewEncapsulation.Native
})
export class MagicButtonComponent {
  @Input() label: string;
}
```

Für den Export als benutzerdefinierte Elemente ist das zentrale `AppModule` angepasst werden.

```
import {BrowserModule} from '@angular/platform-browser';
import {Injector, NgModule} from '@angular/core';
import {createCustomElement} from '@angular/elements';

import {AppComponent} from './app.component';
import {MagicButtonComponent} from './magic-button/magic-button.component';

@NgModule({
  declarations: [
    AppComponent,
    MagicButtonComponent
  ],
  imports: [
    BrowserModule
  ],
  entryComponents: [MagicButtonComponent],
})
export class AppModule {
  constructor(private injector: Injector) {
    const customButton = createCustomElement(
                           MagicButtonComponent, {injector}
                         );
    customElements.define('magic-button', customButton);
  }

  ngDoBootstrap() {}
}
```

Für die Verwendung der Webkomponente sind die JavaScript Dateien einzubinden.

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title>Web Component</title>
  <script src="runtime.js"></script>
  <script src="polyfills.js"></script>
  <script src="scripts.js"></script>
  <script src="main.js"></script>
</head>

<body>
  <magic-button label="Hello World!"></magic-button>
</body>

</html>
```