Universal Compatibility of Framework Agnostic Web Components

Philipp Schmutterer

BACHELORARBEIT

eingereicht am

Fachhochschul-Bachelorstudiengang

Software Engineering

in Hagenberg

im Januar 2023

Advisor:

Ing. Thomas Karl Fischl BSc MSc

Declaration

Hagenberg, January 30, 2023

Philipp Schmutterer

Contents

Abstract

In modern frontend web development, most developers use frameworks like Angular, React or Vue. On a basic level, all of these frameworks use so called web components: bundles of HTML, CSS and Typescript which are independent of the rest of the code and each represented by their own custom HTML tag. The idea is to pack chunks of code which are frequently used throughout a project (i.e. Product cards in a webshop) into reusable components. This reduces the amount of code needed, eliminates the necessety for copy-pasting code segements throughout a web page and promotes “dry” code. However, since not every project will be built using the same framework and all these frameworks use components as their basic building blocks, it would be usefull to be able to build framework agnostic web components that can then be used by any framework. This way, components would only need to be created once instead of creating them in each framework separately. The subject of this Thesis is therefore to answer the question of whether it is possible to create framework agnostic web components and use them in multiple frameworks without the need for considerable amounts of extra code and to shed some light on how this universal compatibility is achieved. As an example, StencilJS will be used to create web components and the frameworks Angular, React and Vue will be used to demonstrate the universal compatibility. Apart from small example pages that contain Components created in StencilJs, a part of this Thesis is also going to be a real-world project; A tablet app for Therapists that displays essential patient data and helps with managing patients.

Chapter 1

Introduction

In der Modernen Web-Frontendentwicklung verwenden die meisten Entwickler Frameworks, wie zum Beispiel Angular, React, oder Vue. Grundsätzlich verwenden diese Frameworks alle sogenannte Web-Komponenten: voneinander unabhängige Blöcke von HTML, CSS und Typescript, welche einen eigenen HTML-tag bekommen. Die Idee dahinter ist es, Codeblöcke, die mehrmals in einem Projekt verwendet werden (z.B. Produktkarten in einem Webshop) in wiederverwendbaren Komponenten zusammenzufassen. Das reduziert die Menge an Code in einem Projekt und erleichtert das Schreiben von „dry“ Code. Da man Allerdings nicht jedes Projekt mit dem selben Framework umsetzen wird, und alle diese Frameworks Komponenten als grundlegende Bausteine verwenden, wäre es nützlich, Frameworkagnostische Komopnenten erstellen zu können, welche dann in jedem beliebigen Framework verwendet werden könnten. Dadurch müsste man Web-Komponenten nur einmal erstellen, anstatt für jedes Framework einzeln. Das Ziel dieser Arbeit ist es daher, festzustellen, ob es möglich ist, Frameworkagnostische Webkomponenten zu erstellen und diese in verschiedenen Frameworks ohne viel Boiler Plate Code zu verwenden.Als Beispiel wird Stenciljs verwendet, um Komponenten zu erstellen und die Frameworks Angular, React und Vue werden dienen Dazu, die Universale Verwendbarkeit der Komponenten zu demonstrieren. Neben einigen Testseiten, die die in Stenciljs erstellten Componenten beinhalten wird auch ein Projekt Teil dieser Arbeit sein. Es geht dabei um eine Tablet-app für Therapeuten, welche Essenzielle Patientendaten schnell und einfach zugänglich macht und das Patientenmanagement unterstützt.

Chapter 2

Fundamentals and Related Works

For context it is important to elaborate some basics about web components and the frameworks they are used in. This means going into the basic architecture of each framework, comparing the structure of their web components and those created in StencilJs.//

2.1 Web Components

At a basic level, a web component is a javascript file that defines an encapsulated piece of HTML, CSS and JavaScript code that can be interperted by a web browser and treated as an HTML element like <p> or <div>. This basic form of a web component does not depend on any framework and can be imported either in javaScript using an import command or in HTML using a <script> tag.

2.2 Angular

Angular is a framework for frontend web development and was created by Google in 2010 and is described in the official documentation as a Typescript based platform that includes a framework to build web applications, as well as many helpful libraries and tools to streamline the entire process of developing and maintaining a web application [1]. This means that Angular is not just a framework, but also has a large number of tools around the framework itself.

2.2.1 Components in Angular

While basic web components in JavaScript are complicated to implement, Angular components use TypeScript and are divided into multiple files for more structure:

- A Typescript file for the component's class (i.e. myComp.component.ts)
- An HTML file for the visual representation (i.e. myComp.component.HTML)
- an SCSS file for styling (i.e. myComp.component.scss)
- a spec.ts (Typescript) file for testing purposes (i.e. myComp.spec.ts)

These files are linked together using the metadata given in the component.ts file.

2.2.2 Basic Concepts

Services

A service is a part of a component that defines a specific behaviour or functionality and is written exclusively in Typescript. Services can be injected into components to provide functionality. This helps to make the code more modular and reuseable.

NgModules

A module in Angular represents a collection of components and services that share a certain context.

Decorators

An angular component is implemented as a Typescript class which can contain decorators with a certain type. A decorator tells the angular compiler how to use the following code (for example @NgComponent to tell the compiler that the following class is a component).

Metadata

as mentioned above, metadata tells the compiler what to do with a certain piece of code. To give a more specific example, the @NgComponent decorator's metadata contains the location of the component's HTML and CSS files. This way all files can be linked to a single component.

Templates

Templates are an enhancement of HTML featured in Angular that allows a developer to inline some functionality like hiding UI-Elements, which would normally take several additional lines of Typescript and CSS. It works by placing HTML code in a <template> tag. The UI element(s) can then be altered using event binding.

Event Binding

Event binding is a way of responding to DOM events inside the HTML code. A good example for this is the click event. By using it inside an HTML element like this:

```
<button (Click)="onClick()">Do something</button>
```

the button will call the onClick function in the Typescript class of the component when it is clicked. The data flow of event binding goes only from the HTML to the typescript class or from child component to parent component.

Property Binding

Similar to event binding, but in this case the data flow is reversed (from HTML to Typescript or from parent to child). Data is passed as a property via HTML. The properties must be defined in the Typescript class For example the path of an image can be given to an img tag with the following code:

HTML element:

```
<img [src]="source">
```

Inside the class:

```
source:string = '../assets/image.jpg'
```

Directives

Directives are also Typescript classes that are defined with the @Directive decorator and can be attached to DOM elements in order to apply a certain behavior like changing the background color of a button while it is clicked. Directives are modular and can be used multiple times.The graph below describes how these concepts work together.

Chapter 3

Working with LaTeX

Chapter 4

Figures, Tables, Source Code

Chapter 5

Mathematical Elements, Equations and Algorithms

Chapter 6

Using Literature and other Resources

[1]

Chapter 7

Printing the Manuscript

Chapter 8

Closing Remarks

Appendix A

Technical Details

Appendix B

Supplementary Materials

List of supplementary data submitted to the degree-granting institution for archival storage (in ZIP format).

B.1 PDF Files

Path: /

thesis.pdf	Master/Bachelor thesis (complete document)

B.2 Media Files

Path: /media

*.ai, *.pdf	Adobe Illustrator files
*.jpg, *.png	raster images
*.mp3	audio files
*.mp4	video files

B.3 Online Sources (PDF Captures)

Path: /online-sources

Reliquienschrein-Wikipedia.pdf

Appendix C

Questionnaire

Appendix D

LaTeX Source Code

References

Literature

[1] Hubert M. Drake, Milton D. McLaughlin, and Harold R. Goodman. *Results obtained during accelerated transonic tests of the Bell XS-1 airplane in flights to a MACH number of 0.92*. Tech. rep. NACA-RM-L8A05A. Edwards, CA: NASA Dryden Flight Research Center, Jan. 1948. URL: https://www.nasa.gov/centers/dryden/pdf/87528main_RM-L8A05A.pdf (cit. on p. 8).

Online sources

[2] *Reliquienschrein*. Aug. 29, 2022. URL: https://de.wikipedia.org/wiki/Reliquienschrein (visited on 01/13/2023).

Check Final Print Size

— Check final print size! —

width = 100mm
height = 50mm

— Remove this page after printing! —