



ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

**Faculty of
Engineering, Computer
Science and Psychology**
Databases and Information
Systems Department

Evaluating State-of-the-Art Web Component Frameworks

Bachelor's thesis at Universität Ulm

Submitted by:

Stefan Engelmayer

stefan.engelmayer@uni-ulm.de

Reviewer:

Prof. Dr. Manfred Reichert

Supervisor:

Dr. Johannes Schobel

2019

Version from May 29, 2019

© 2019 Stefan Engelmayer

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/de/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Composition: PDF- \LaTeX 2 ϵ

Abstract

Websites are getting more complex over the years because it is not just used to display information, but to use some applications. One challenge is to maintain these services and keep them up to date. To reuse existing code, Web Components are introduced to outsource complex structures as an HTML element including its functionality. Therefore, some frameworks available which help to create Web Components by providing some useful tools and make the development process easier for developers. Because there are numerous different frameworks available for the development, it is not easy to find the right one for the own project. Because the standards are changing fast in computer science, the development process should always be *State-of-the-Art*.

The aim of this thesis is to give a brief overview over Web Component frameworks and find out which framework is a good choice for given use cases. First some frameworks are introduced. Three selected frameworks are more detailed introduced. For the evaluation of these frameworks, the used criteria are introduced. With the use of the analytical hierarchy process, three scenarios are evaluated to get the best framework for each scenario. This shows that when the requirements differ, also the choice of the framework can change.

Acknowledgment

Acknowledge everyone that helped during the creation of the thesis here.

At this point I want to thank everyone who helped me during the creation of the thesis.

First, I want to thank my reviewer Prof. Dr. Manfred Reichert.

Second, I want to thank Johannes Schobel who was a great supervisor.

Finally, special thanks go out to my family and my friends who supported me whenever they could.

Contents

1	Introduction	1
2	Fundamentals	3
2.1	Web Components	3
2.1.1	Concepts and Usage	3
2.2	Languages	10
2.3	NPM	10
2.4	Testing	11
3	Related Work	13
4	Web Component Frameworks	15
4.1	Polymer	15
4.1.1	Features	15
4.1.2	First Steps	19
4.1.3	Export	19
4.2	Stencil	19
4.2.1	Features	19
4.2.2	First Steps	20
4.2.3	Export	21
4.3	Vue.JS	21
4.3.1	Features	21
4.3.2	First Steps	22
4.3.3	Export	23
4.4	Additional Frameworks	24
4.4.1	Hybrids	24
4.4.2	SkateJS	24
4.4.3	SlimJS	24
4.4.4	Glimmer	25
4.4.5	Dojo.io	25

Contents

5	Evaluation	27
5.1	Criteria	27
5.1.1	Framework Selection	27
5.1.2	Weak Criteria	28
5.1.3	Ease of Setup	28
5.1.4	Ease of Export	28
5.1.5	Strong Criteria	28
5.2	Evaluation of the Frameworks	32
5.2.1	Inheritance	32
5.2.2	Linter/Formatter	33
5.2.3	Documentation Generation	34
5.2.4	Testing Frameworks	35
5.2.5	Induction	35
5.2.6	Auto Recompile	36
5.2.7	GUI/UI	36
5.2.8	CSS	37
5.2.9	CSS Preprocessing	37
5.3	AHP	38
5.3.1	Criteria Weights	40
5.4	Use Cases	41
5.4.1	Small Project	41
5.4.2	Medium Project	43
5.4.3	Large Project	44
5.5	Conclusion	45
6	Summary	47
A	Sources	51

1

Introduction

The Web is still growing and with its popularity the possibilities of developer and end user are getting more. The internet is not just used to get information, it is commonly used to communicate, buying products and much more. To reduce the complexity and keep maintainability various frameworks are available to support developer for their development. With frameworks it is easier to set up a project.

Web Components are another level of reducing the complexity of a website for example. The idea is to reduce the needed code to a minimum by hiding all required information into one or multiple external files which can be included and reused as a custom element, defined by the developer. This reusable element should not interfere with the rest of the website or vice versa, the styling information and functionality is separated from the outside of the Web Component.

First it must be decided what could be possible criteria for selecting a good framework and are important for a project. Then it should be considered if there are some other restrictions because one framework needs to support a specific feature to limit the list of possible frameworks. Time must be investigated to find out which features are included in the frameworks. Otherwise it is not possible to compare the frameworks. After the criteria are set, they can be structured to make the decision process easier.

For the use of the analytical hierarchy process, each framework which is considered as an alternative has to be rated according to all criteria. When this is done, the criteria get weighted depending on how important it is for a specific scenario. After all priorities

1 Introduction

are set, it can be seen how well each framework performs. To understand why the frameworks perform so different, the AHP method is a good choice because it structures the problem well and the process of the decision can be visually shown in a tree.

The thesis begins with the explanation of the most important fundamentals in Chapter 2. Followed by the introduction of related work in Chapter 3. After that, the frameworks are introduced in Chapter 4 by explaining how they can be used to create Web Components, and which tools the frameworks include. After that, in Chapter 5 the used criteria are explained and followed by this the frameworks are evaluated. To complete this, the results are used to evaluate which framework is the best in three different use cases. Finally, in Chapter 6 the thesis is summarized.

2

Fundamentals

This chapter conveys important knowledge which is used during the thesis. It starts with some basic details about Web Components and its structure in Section 2.1 and continues with a brief overview of the programming languages which could be used to write these Web Components in Section 2.2. After the Node Packet Manager is introduced in Section 2.3. Last but not least the testing is getting explained in Section 2.4

2.1 Web Components

Web Components are used to write a more maintainable code. A bunch of code can be replaced with a simple tag so that there is less code on the main site. This makes it easier to get an idea of what the code is about.

A developer wants to reuse as much code as possible. For websites it is not easy to reuse written code because it is not isolated from the rest of the application, especially its style. Web Components can isolate the HTML code, its style information and its functionality so that this element does not change its behavior or design.

2.1.1 Concepts and Usage

Reusing code is a good idea and this is exactly why there are Web Components. They are designed to be reused multiple times without any complications and collisions. Therefore, it combines three technologies which are „Custom Elements“, „Shadow DOM“ and „HTML Templates“ [1]

2 Fundamentals

Custom Element

A Custom Element is a set of JavaScript APIs which allows you to define your own custom element with self defined functionality and use it in your user interface. [2] A custom element is created with JavaScript.

There are two different types of custom elements. One of them are *autonomous custom element* and the other type is called *customized built-in element*. [3] The general structure of such custom elements is listed in the sources. The key difference between these two elements is that with an autonomous custom element you create a new element like in Listing 2.1 but with a built-in custom element you can extend an element based on an existing element. For example you can extends from a `Button` element in the register function and add some functionality to it as in this custom button example in Listing 2.2.

```
1 class CustomElement extends HTMLElement {
2   constructor() {
3     super();
4     // code of the custom element
5   }
6 }
7
8 // register and assigns a tag to this element
9 customElements.define('element-tag', CustomElement);
```

Listing 2.1: Autonomous Custom Element

```
1 class ChangeColor extends HTMLButtonElement {
2   constructor() {
3
4   }
5 }
6 customElements.define("change-color", ChangeColor, {
7   extends: "button"
8 });
9
10 const changeColor = document.createElement("button", {
11   is: "change-color"
12 });
13 changeColor.textContent = "Click meScript!";
14 document.body.appendChild(changeColor);
```

Listing 2.2: Built-in Custom Element

You can also extend the JavaScript class from an other component. It has the same behavior as the parent one. This is useful if you want a similar element with only a few changes. To override a method you just have to type the method head again and define a different body.

There are some rules for creating a new custom element which are described in [4]:

1. The name of a custom element must contain a dash (-). So `<x-tags>`, `<my-element>`, and `<my-awesome-app>` are all valid names, while `<tabs>` and `<foo_bar>` are not. This requirement is so the HTML parser can distinguish custom elements from regular elements. It also ensures forward compatibility when new tags are added to HTML.
2. It is not possible to register the same tag multiple times. Attempting to do so will throw a `DOMException`.
3. Custom elements cannot be self-closing because HTML only allows a few elements to be self-closing. Always write a closing tag (`<app-drawer></app-drawer>`).

Shadow DOM

The Shadow DOM often finds its usage to hide complex structures behind a simple element. This Shadow DOM is hidden in the element so that a developer gets less code displayed. Because of the Shadow DOM hides a bunch of code, it also protects for the hidden code from the application. For example a style guide which is set in an application outside the Shadow DOM does not affect any element inside the Shadow DOM. This protection is the reason why it is not possible to change the appearance of an already built element. To Change the appearance the element has to be rewritten. Each shadow DOM has the following structure: [2]

The **Shadow Host** is the only visible element for the user, e.g. the `<video>` tag. This is the root node on which the actual shadow DOM is attached to.

The **Shadow Root** is the root node of the Shadow Tree and represents the connection node between the Shadow Host and the Shadow Tree. It is not visible for the user but it is rendered in the browser.

2 Fundamentals

Shadow Boundary: Every HTML- and CSS-Code inside the Shadow Root is protected by a barrier to its parent document. Styles only have an effect on the component, other styles of the rest of the site or other components do not affect the own style.

Shadow Tree: This is the Shadow Root and its content child nodes the shadow DOM as shown in Figure 2.1.

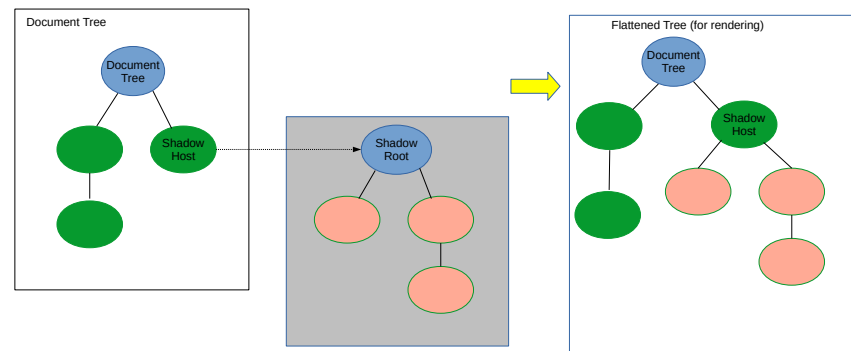


Figure 2.1: Redawn Shadow Tree inside the DOM from [2]

The Figure 2.2 shows the code needed for the audio element. There are only a few lines of code, which will show the user a simple UI to play, pause, jump to a specific time or download the audio file. If one browser is not capable of playing this audio file, a custom message is displayed to the user and he can access the audio file with a link. The second picture in Figure 2.3 shows the hidden Shadow Tree which contains the actual code of the audio element. This is all the code which a developer would have to write if the element does not exist.

HTML Template

The HTML Template is a raw HTML skeleton that can be used multiple times. This template is not rendered by default, but it is a client-side template to be filled with content,

```

<!DOCTYPE html>
<html>
  <head>
    <title></title>
  </head>
  <body>
    <audio id="dieyoung" controls src="Kesha - Die Young.mp3" type="audio/mp3">
      Your browser does not support playing this audio file.
      You can access the audio file with this links: <br/>
      <a href="Kesha - Die Young.mp3">Kesha - Die Young.mp3</a>
    </audio>
  </body>
</html>

```

Figure 2.2: Clean <audio> Tag which has to be written by the developer

```

<!doctype html>
<html>
  <head>...</head>
  <body>
    <audio id="dieyoung" controls src="Kesha - Die Young.mp3" type="audio/mp3">
      ...
      <div pseudo="-webkit-media-controls" class="phase-ready state-stopped">
        <div pseudo="-webkit-media-controls-overlay-enclosure">
          <input pseudo="-internal-media-controls-overlay-cast-button" type="button"
            style="display: none;">...</input>
        </div>
        <div pseudo="-webkit-media-controls-enclosure">
          <div pseudo="-webkit-media-controls-panel">...</div>
        </div>
        <div role="menu" aria-label="Options" pseudo="-internal-media-controls-text-track-
          list" style="display: none;">...</div>
        <div pseudo="-internal-media-controls-overflow-menu-list" class="closed" style=
          "display: none;">
          <label pseudo="-internal-media-controls-overflow-menu-list-item" tabindex="0"
            class="animated-0" style="display: none;">...</label>
          <label pseudo="-internal-media-controls-overflow-menu-list-item" tabindex="0"
            style="display: none;">...</label>
          <label pseudo="-internal-media-controls-overflow-menu-list-item" tabindex="0"
            style class="animated-0">...</label>
          <label pseudo="-internal-media-controls-overflow-menu-list-item" tabindex="0"
            class="animated-2" style="display: none;">...</label>
          <label pseudo="-internal-media-controls-overflow-menu-list-item" tabindex="0"
            class="animated-1" style="display: none;">...</label>
          <label pseudo="-internal-media-controls-overflow-menu-list-item" tabindex="0"
            class="animated-0" style="display: none;">...</label>
        </div>
      </div>
      "
      Your browser does not support playing this audio file.
      You can access the audio file with this links: "
      <br>
      <a href="Kesha - Die Young.mp3">Kesha - Die Young.mp3</a>
    </audio>
  </body>
</html>

```

Figure 2.3: Audio element with a part of its Shadow DOM expanded

2 Fundamentals

and inserted into the DOM and to be rendered. It is not mandatory to use a HTML template, but it helps to keep the code clearer.

For example if a userlist has to be displayed, the same structure is used every time. To make the code better maintainable it is a good idea to use a HTML template. If, for example, a field has to be added later, only the template and the JavaScript code has to be added. In Figure 2.4 is an example for a simple HTML template. To fill the template with some data and display it on a HTML page see the example code in Figure 2.5. The resulting rendered site is shown in Figure 2.6


```

<template id="html-template">
  <hr/>
  <div class="user-name"></div>
  <div class="user-surname"></div>
  <div class="user-telephone"></div>
</template>

```

Figure 2.4: HTML template for a list of users

```

<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Template-Demo</title>
  </head>
  <body>
    <div id="placeholder"></div>
  </body>
</html>

<script>
  // Users which are displayed on the rendered site
  var users = [
    {id: '0', 'name': 'Stefan', 'surname': 'Engelmayer', 'telephone': '+491856xxx'},
    {id: '1', 'name': 'Flora', 'surname': 'Watson', 'telephone': '+4419746xxx'},
    {id: '2', 'name': 'Max', 'surname': 'Muster', 'telephone': '+49123456xxx'}
  ];

  // Reference to the placeholder
  var userList = document.getElementById('placeholder');

  // For each entry fill in the template
  for (var i = 0; i < users.length; i++) {
    var user = users[i];
    // clones the node - the boolean in cloneNode() specifies if the node included its child nodes
    var template = document.getElementById('html-template').content.cloneNode(true);
    template.querySelector('.user-name').innerText = user.name;
    template.querySelector('.user-surname').innerText = user.surname;
    template.querySelector('.user-telephone').innerText = user.telephone;
    userList.appendChild(template);
  }
</script>

```

Figure 2.5: Code to use the HTML template and fill it with data

```

-----
Stefan
Engelmayer
+491856xxx
-----
Flora
Watson
+4419746xxx
-----
Max
Muster
+49123456xxx
-----

```

Figure 2.6: The rendered page

2.2 Languages

In this thesis are some programming languages mentioned which are quite the same. They are introduced in short in this section.

JavaScript is a famous scripting language which is interpreted and executed directly. There is no need to compile JavaScript code before it can be executed. The scripting language has a formal standardization which is called ECMA Script and has multiple released versions.

TypeScript is a programming language which is quite the same as JavaScript but it is an object oriented programming language and errors are printed out during the compiling process and not at runtime. Compiling a TypeScript file will output JavaScript so it is compatible to standard JavaScript applications.

Babel is very close to TypeScript but in addition to the TypeScript features Babel also adds and supports some features which are not yet supported by the browsers by default.

2.3 NPM

The Node.js Package Manager is a large JavaScript software registry. It is possible for everyone to register as a free user and upload some snippets so that everyone can use those open source snippets. The website can be used to search uploaded software packages which can be included as a dependency into an existing project. This has the advantage that there is no need to include a dependency manually into a project and push it into its repository. To add an existing dependency into a project it is only necessary to run the command 'npm install package-name --save'. After this command is executed the dependency gets installed and added into the 'package.json' file in the project directory. Hence it is possible to easily reinstall all needed dependencies which are required for this project to compile and run on another computer. Another big advantage is that npm prints out a message when some dependencies can be updated or if there are some known vulnerabilities in the used package of a project. [5]

2.4 Testing

Web Components are designed that they only have to be included within a project and can be inserted into the DOM by adding its defined tag at the proper place. The used Web Components should be tested especially if someone else writes a component and another dev uses it. If there are already tests included, a developer has less effort to test the component. There are some frameworks, which helps to test websites like the JavaScript based Mocha framework or Jest. They also help to write automated tests which can also run in the browser.

In addition to these powerful testing frameworks there is the Selenium framework. Selenium runs the tests at various different browsers and browser version which can be specified by the developer. So the developer does not have to manually install all the different browser versions and run the test for each browser where the application should run.

3

Related Work

In this section related work is listed. Because there is no close related work about Web Components, also further work is considered as a kind of related work. The following section contains some related work which compares and evaluates frameworks and tools, but they are not about creating Web Components.

In [6] the author wants to find out, which machine tool is the best alternative in market. Fuzzy AHP is based on the analytical hierarchy process for the decision making process. This kind of extension of AHP provides a more accurate description of the decision making process.

Strukturierter Vergleich aktueller Frameworks zur Entwicklung mobiler Anwendungen
[7] evaluates frameworks to develop mobile applications. In this thesis three selected cross-platform frameworks for mobile applications are introduced and compared. For the evaluation the analytical hierarchy process is used. The result is that there is no best framework for everything, the framework has to be chosen by its specific use case.

[8] compares JavaScript frameworks with self defined criteria. The author programs a To-Do application with each framework he evaluates to get an overview of what the frameworks are capable of. For the evaluation itself the author first defines what he explains what the meaning of each criteria is and after this explanation he describes if the frameworks fulfill the criteria. Though he does not explain why he choose this method for his evaluation. The conclusion of this thesis is that AngularJS and VueJS are good frameworks for small projects and BackboneJS for bigger ones.

3 Related Work

In [9] three WebGL frameworks are evaluated by programming a demo application with each framework. Therefore, he explained some criteria which are used for the evaluation. After that he introduced the frameworks, analyzes them based on the criteria and described how his implementation of a demo application worked. For his result of the evaluation he showed what each framework supports and that they all have different approaches how they are used.

There is no thesis which is about evaluation state-of-the-art Web Component frameworks. The related work is about general decision making approaches which is related because they use a multiple criteria decision technique. The further work is all about a comparison of frameworks. [8, 9] do not specify on what their decision of which framework is better, is based on.

4

Web Component Frameworks

To get a brief overview which functionality these three selected frameworks offer, In the following sections Polymer, Stencil and Vue.js are introduced and it is explained how to create Web Components with these frameworks. Each Web Component can be published via NPM besides the described method of exporting them on the local computer.

4.1 Polymer

The Polymer framework version 3.x is mainly developed by Google devs and other people on GitHub. The Polymer-CLI can be installed through NPM. Besides JavaScript it is also possible to develop Web Components with TypeScript. The following subsection is about some features which are listed on the official website ¹.

4.1.1 Features

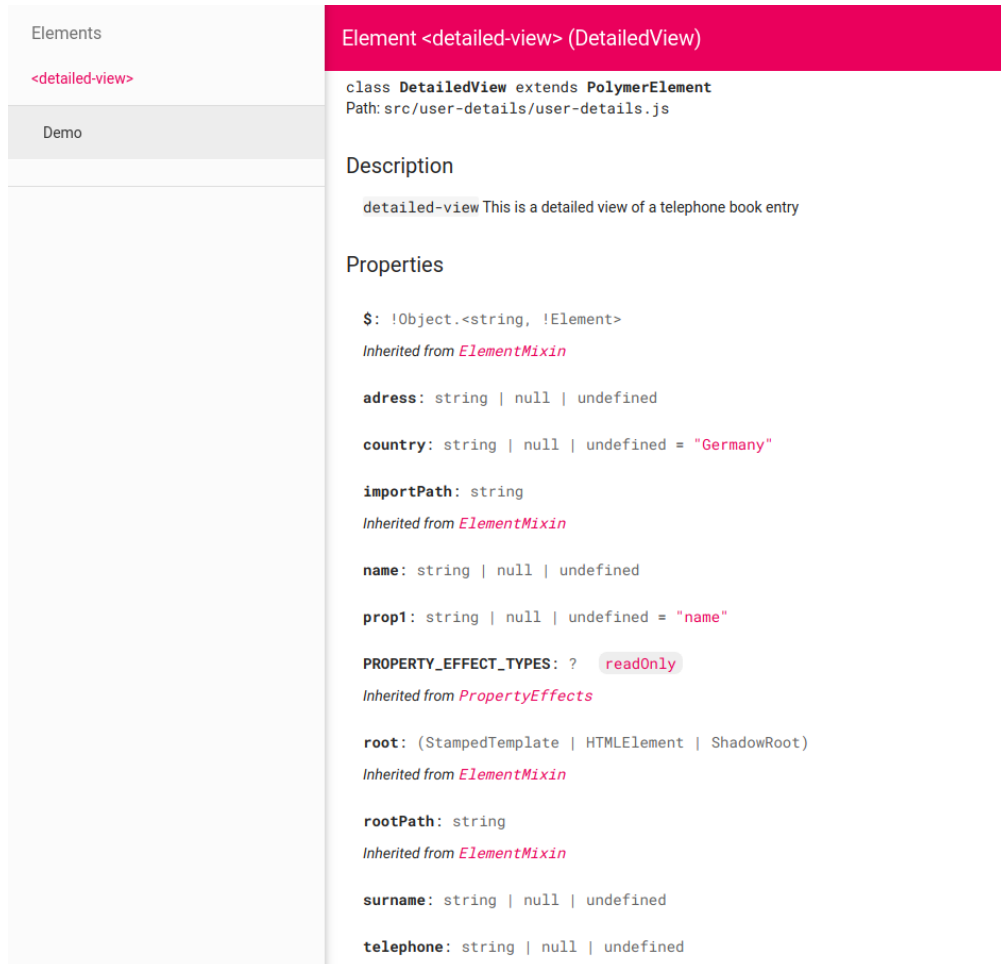
The Polymer framework has multiple frameworks included so it is not necessary to search for other frameworks which can be used to improve the coding experience and they are also tested that they work together. Some testing frameworks like Mocha, Chai, Sinon, Selenium and Accessibility Developer Tools are already included. To make sure that the Web Component is compatible with older browsers it is possible to include Polyfills into a project.

Another feature allows to derive the documentation from existing code. This requires an

¹<https://polymer-library.polymer-project.org/3.0/docs/>, last viewed 15 Mai 2019

4 Web Component Frameworks

additional dependency and manually add a line of code to a `docs.html` to let the CLI generate the documentation. It is only required to write comments before some classes, methods, properties and events.



The screenshot shows a documentation page for the `<detailed-view>` web component. The page is divided into several sections:

- Elements:** Lists the component tag `<detailed-view>`.
- Demo:** A placeholder for a live demonstration of the component.
- Element `<detailed-view>` (DetailedView):** A red header bar.
- class DetailedView extends PolymerElement:** The class definition, with the file path `Path: src/user-details/user-details.js`.
- Description:** A paragraph stating: `detailed-view` This is a detailed view of a telephone book entry.
- Properties:** A list of properties with their types and inheritance information:
 - `$:` `!Object.<string, !Element>` (Inherited from `ElementMixin`)
 - `adress:` `string | null | undefined`
 - `country:` `string | null | undefined = "Germany"`
 - `importPath:` `string` (Inherited from `ElementMixin`)
 - `name:` `string | null | undefined`
 - `prop1:` `string | null | undefined = "name"`
 - `PROPERTY_EFFECT_TYPES:` `? readOnly` (Inherited from `PropertyEffects`)
 - `root:` `(StampedTemplate | HTMLElement | ShadowRoot)` (Inherited from `ElementMixin`)
 - `rootPath:` `string` (Inherited from `ElementMixin`)
 - `surname:` `string | null | undefined`
 - `telephone:` `string | null | undefined`

Figure 4.1: Generated doc for the properties.

The Polymer framework also supports tracking gesture events, for example to get the coordinates of the movement of a finger on the screen. To improve the user experience the PRPL pattern is used in the framework. PRPL stands for: [10]

- Push critical resources for the initial route.
- Render initial Code.

Elements

<detailed-view>

Demo

Element <detailed-view> (DetailedView)

Methods

`static createPropertiesForAttributes(): void`
Inherited from PropertyAccessors

Generates property accessors for all attributes in the standard static `observedAttributes` array. Attribute names are mapped to property names using the dash-case to camelCase convention

`static finalize(): void`
Inherited from PropertiesMixin

Finalizes an element definition, including ensuring any super classes are also finalized. This includes ensuring property accessors exist on the element prototype. This method calls `_finalizeClass` to finalize each constructor in the prototype chain.

`attributeChangedCallback(name: string, old: ?string, value: ?string, namespace: ?string): void`
Inherited from PropertiesChanged

Implements native Custom Elements `attributeChangedCallback` to set an attribute value to a property via `_attributeToProperty`.

- `name` Name of attribute that changed
- `old` Old attribute value
- `value` New attribute value
- `namespace` Attribute namespace.

`connectedCallback(): void`
Inherited from ElementMixin

Provides a default implementation of the standard Custom Elements `connectedCallback`.

The default implementation enables the property effects system and flushes any pending properties, and updates shimmed CSS properties when using the ShadyCSS scoping/custom properties polyfill.

Figure 4.2: Generated doc for some methods.

4 Web Component Frameworks

- Pre-cache remaining routes.
- Lazy-load and create remaining routes on demand.

These are all methods to make the user experience better and the application more stable. For Browser which doesn't support HTTP2 Push it is possible to compile with the `--bundle` flag to create multiple bundles (the shell and one for each fragment) so that the data transfer is optimized.

To take advantage of such a caching it is possible to use a service worker. Such a service worker can improve the performance by delivering some content of the cache and event work when the client is offline.

The structure of such an initial Polymer app is shown in Figure 4.3

The `index.html` is the entry point which loads the web component located at `src/my-application-app/my-application-app.js`. Tests for the web component are located in the test directory. This simple directory structure is very familiar in comparison with other project structures.

```
polymer-app/  
├── index.html  
├── manifest.json  
├── package.json  
├── package-lock.json  
├── polymer.json  
├── README.md  
├── src  
│   └── my-application-app  
│       └── my-application-app.js  
└── test  
    └── my-application-app  
        └── my-application-app_test.html
```

Figure 4.3: Structure of a simple Polymer-3 application

4.1.2 First Steps

With the installed Polymer-CLI a new polymer-3-application can be initialized by using the command `polymer init`. In its created directory the `src/` directory is a folder of the Web Component.

To get an additional component just add another folder in the `src/` directory with another `.js` file. The new Web Component can be included as a link in the `index.html` in the root directory of the project.

4.1.3 Export

To export a finished Web Component the command `polymer build` has to be executed. After the build process has finished, a build directory exists in the project directory. There is a default folder with the build ready Web Component. The content can be copied into the root of a web server and used.

4.2 Stencil

A quite young framework is Stencil, its first commit on GitHub is dated to the 15th February of 2017. Stencil, which can be installed through NPM, builds reusable Web Components. This Framework supports TypeScript to develop Web Components. The following subsection is about features of Stencil which are mentioned on its website ².

4.2.1 Features

It is possible to write a component using standard JavaScript, CSS and HTML. The TypeScript support is optional. The framework also tries to auto adjust written components when there are updates so that a developer does not have to make any adjustments manually. If they have to make any changes stencil tries to tell the developers what

²<https://stenciljs.com/docs/my-first-component/>, last viewed 15 Mai 2019

4 Web Component Frameworks

exactly they have to change in the code. Another feature of Stencil is that the develop server tries to recompile the project and serves this changes if a file gets changed . The Web Browser automatically reloads the content so the changes take effect immediately.

Stencil can generate a `Readme.md` file, which contains a table of all the properties including their attribute, property description, type and default value. To activate generating this `Radme.md` the `stencil.config.ts` has to be edited as described in [11] or the build command can be executed with an additional `--docs` flag. A sample of an generated property list (`.tsx` and `.css` file) is shown in Figure 4.4

my-component

Properties

Property	Attribute	Description	Type	Default
address	address	Address of a person	string	undefined
isActive	is-active	Shows if the modal is visible or not	boolean	false
name	name	Name of a person	string	undefined
surname	surname	Surname of a person	string	undefined

Built with [StencilJS](#)

Figure 4.4: Auto Generated Readme.md with a List of Used Properties.

The testing framework integrated into Stencil is Jest and supports both isolated unit-testing and the more realistic end-to-end testing. The latter runs in an actual browser to get mores realistic results.

4.2.2 First Steps

After stencil gets installed through `npm`, a new project can be initialized via the command `npm init stencil`. The option `component` is enough for simple Web Components. The component can then be edited under `src/`. Stencil can recompile the project if it detects a file change and send the changes to the browser. Because of this changes are visible immediately with a little delay because it needs to recompile.

To add a second component to the bundle another `.tsx` or `.js` file has to be created in the `src/` folder. If a CSS preprocessor support is needed a plugin has to be installed via NPM. For example if sass should be used in the project it can be installed with `npm`

`install @stencil/sass --save-dev` and additionally the plugin has to be added to the plugins list in the file `stencil.config.ts`.

4.2.3 Export

For the build no further configuration is needed. Wait, Stencil may need a build command with `npm run build`. This build process creates files in the `dist/` folder. The only required files are the `component-name.js` and its directory. All other files are optional and may be used if you want to use the Web Component e.g. for older web browsers. The required files can also be picked from the `www/` directory which are minified after the build process.

4.3 Vue.JS

Vue is one of the most popular frameworks, which has a lot of features included. It is possible to write a simple website, a Web Component or a Progressive Web Application, which is an alternative to a native application. All of the following information is based on version 2.0. The following subsection is about some features which can be found on its website ³.

4.3.1 Features

For the main app Babel, TypeScript or plain JavaScript is supported. To apply style, native CSS files are supported and optionally CSS preprocessors like Sass/SCSS, Less or Stylus are available.

One useful feature of this framework is that Vue comes with a user interface which can be started through the Command Line and is accessible through a web browser. With the Vue UI multiple projects can be added and managed and all operations to start an

³<https://vuejs.org/v2/guide/>, last viewed 15 Mai 2019

4 Web Component Frameworks

app, test or stop an application are implemented. It is also possible to search and add some needed plugins with the UI.

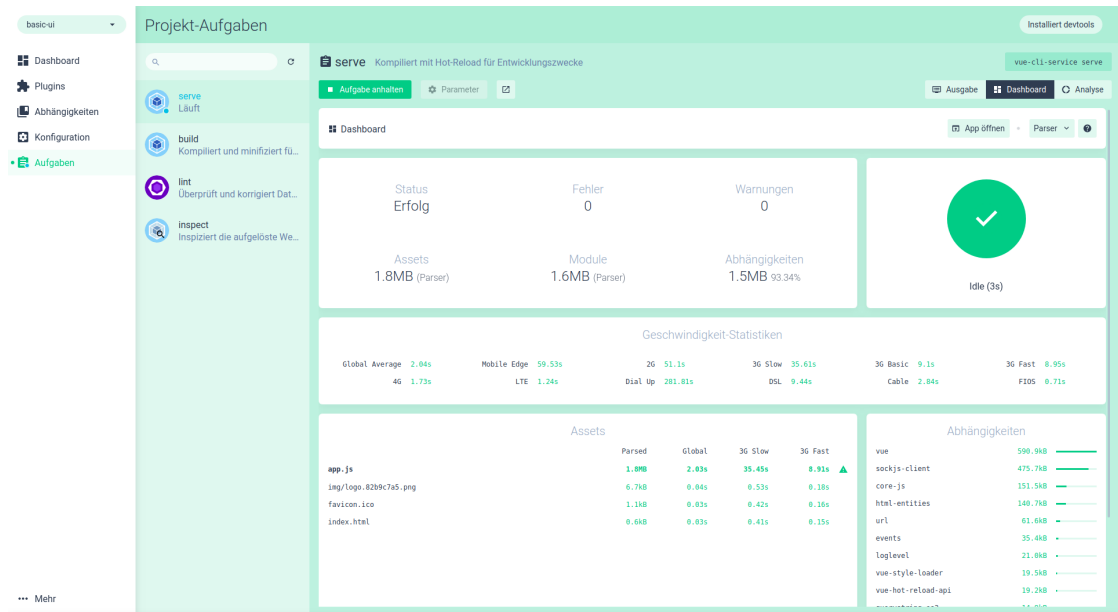


Figure 4.5: UI Shows Serving a Project

To help to write high quality code Vue has some linter and formatter included. Linter helps to detect some errors in the code before running the app. Prettier helps to format the code to make it easier to read. Vue supports TSLint or ESLint with four different configs: Only error prevention, Airbnb, Standard or Prettier.

Some testing frameworks for E2E and Unit testing are also included. For the E2E testing Cypress and Nightwatch is already included in Vue. Shipped Unit Testing frameworks are Mocha, Chai and Jest.

During the initialization of a new Vue application everything is created in a local git repository.

4.3.2 First Steps

To initialize a new project the Vue CLI has to be installed. After that the project can be initialized with `vue create appname`. The supported features can be manually selected

in the command line as shown in Figure 4.6

```
Vue CLI v3.6.3
? Please pick a preset: Manually select features
? Check the features needed for your project:
   Babel
   TypeScript
   Progressive Web App (PWA) Support
   Router
   Vuex
   CSS Pre-processors
   Linter / Formatter
   Unit Testing
   E2E Testing
```

Figure 4.6: Feature selection with the Vue CLI.

The used formatter or testing framework are selected later in the initialization process. After the initialization is finished the project can be started and the development of the Web Component can begin. The Web Component is located under `src/components/`. To create a new Component only a new `component.vue` file has to be created in this directory. To display the component, the `App.vue` file has to be edited that the content of this component gets loaded.

4.3.3 Export

To export a single Web Component on the command line, it is possible to execute a command like `vue build --target wc --name <wc-name> <path-to-file>`. This creates one readable and a minified version in the `dist/` folder. The Web Component is already included in the `demo.html` file to provide a working example. To use this Web Component in a project, Vue has to be included first via the script include `<script src="https://unpkg.com/vue"></script>`.

4.4 Additional Frameworks

The following frameworks can also be used to create Web Components. For each of the following frameworks a short overview is provided based on the information of its website or GitHub repository and why these frameworks are not considered for further evaluation.

4.4.1 Hybrids

The Hybrids framework ⁴ looks promising, it is easy to get into it because they provide some live examples on their GitHub page. The used language to create Web Components in Hybrids is JavaScript, TypeScript support is missing. Because it has only 1k stars on GitHub so this framework is not being evaluated in this thesis.

4.4.2 SkateJS

SkateJS ⁵ is a light framework which helps to write Web Components. But it needs more preparation to get started with SkateJS because some libraries, like the render library LitHTML has to be installed manually. This framework is good if a developer only wants libraries installed which are really needed. The standard documentation could be more detailed so it takes longer to get familiar with this framework. SkateJS has only 3k stars on GitHub and does not meet the criteria listed in Section 5.1.1

4.4.3 SlimJS

The SlimJS framework ⁶ is a small framework which adds some functionality to reduce boiler plate code if of a Web Component. The setup is easy because only to import a `.js` file has to be imported into your project and then the framework can be used to create JavaScript based Web components. A disadvantage would be that this framework

⁴<https://hybrids.js.org/>, last viewed 15 Mai 2019

⁵<https://skatejs.netlify.com/>, last viewed 15 Mai 2019

⁶<http://slimjs.com/>, last viewed 15 Mai 2019

is only useful for a small project to test what Web Components are. There is no way to add more functionality without switching to another framework. SlimJS is not evaluated in this thesis because it has only 519 stars on GitHub.

4.4.4 Glimmer

Glimmer⁷ is a young framework which supports Typescript. This framework has also a rendering testing suite included. It has only 483 stars on GitHub, which is why this framework is not evaluated in this thesis.

4.4.5 Dojo.io

Dojo.io⁸ supports TypeScript but has only 139 stars on GitHub. The documentation of this framework contains many live examples which helps to get into it. Dojo.io has a strong focus on its user interface so if the main topic is a good design maybe this framework may be a good choice.

⁷<https://glimmerjs.com/>, last viewed 15 Mai 2019

⁸<https://dojo.io/>, las viewed 15 Mai 2019

5

Evaluation

In this Section the three frameworks Polymer, Stencil and Vue are evaluated using the analytical hierarchy process (AHP) [12] which is an enhanced Multiple Criteria Decision Analysis method. With AHP it is possible to set priorities to the criteria in comparison. This helps to decide which of the three frameworks is the best for a project.

5.1 Criteria

To evaluate frameworks, it is needed to define criteria. These criteria used in the evaluation are split up in two groups: Weak criteria are introduced in Section 5.1.2 and strong criteria as in Section 5.1.5. In each Subsection is explained why these criteria are used and what exactly the meaning of each criterion is.

5.1.1 Framework Selection

These Criteria are used to sort out which framework should not be evaluated, and which are evaluated. The frameworks which do not meet the following requirements are listed in Section 4.4.

Target Platforms

To cover most use cases the produces Web Components should be able to run on quite every Soft- and Hardware. Therefore, one requirement is that the framework can export the Web Component that it can be imported in a project as a `.js` file. So, it is guaranteed

5 Evaluation

that these Web Components can be reused in large projects as in small projects, which may not have a big hardware setup.

GitHub Stars

Another criterion to sort out some frameworks is the amount of GitHub Stars in the Git repositories. They are a good indicator to see if a framework is good or not. The minimum to 5.000 stars is set to make sure that the framework is either not a newcomer or an older framework which is not used or supported anymore.

5.1.2 Weak Criteria

Here are criteria listed which have no impact on the decision itself, but their aspect is important for selecting a good framework.

5.1.3 Ease of Setup

With all three frameworks it takes only a few steps to get them installed and running. Every framework can be installed via NPM, they are all equal in this category.

5.1.4 Ease of Export

With all three frameworks it is quite easy to export the finished project as a Web Component, which can be imported into another project. There are some differences in how this has to be done which are explained in Chapter 4 at Section 4.1.3, 4.2.3 and 4.3.3.

5.1.5 Strong Criteria

In this section are the strong criteria listed and described. Those are used to compare the three frameworks which meets the requirements of the selection criteria mentioned in Section 5.1.1.

Features

Each framework contains multiple other tools. This section lists some features which are later compared by their importance.

1. **Inheritance:** Web Components are easy to reuse but if some logic has to be changed the Web Component has to be modified. Some frameworks support the inheritance of another component. This makes it easy if only methods or functionality has to be changed.
2. **Linters/Formatter:** Writing code can be difficult. This can be supported by tools, which helps to format the written code so that it is more readable. Linter also helps to prevent errors by inspecting the code. This has the advantage that errors can be detected before it gets executed. It is compared how much of these features are available in the framework.
3. **Docs generation:** A programmed Web Component needs to be documented. Each framework has its own documentation framework to generate a documentation which is up to date and does not need to be updated manually. The rating with this framework displays how easy it is to generate the docs and how good the generated documentation is structured.
4. **Testing Frameworks:** Most Web Component frameworks also includes some tools to test their components. To rate these frameworks is not easy because every of the evaluated frameworks includes testing frameworks. To rate this criterion how much tools the frameworks offer.

User Experience

Another important topic is the user experience. If a framework has a good user experience and the induction time is very low, the framework gets a better acceptance from the developers. This is a very subjective criteria so the rating can be different depending on who is asked how good or bad the user experience in his opinion is.

5 Evaluation

1. **Easy Setup:** The first initialization of a project and get a starter project running. It depends on what it takes to get started, how many software has to be installed and if the framework requires any special configuration.
2. **Induction:** Every framework is a bit different so to get into a new framework it takes some time depending on what knowledge a dev already has and how well the documentation of the framework is. A well-structured and documentation and of course a easy to use concept get a better ranking.
3. **Auto Recompile:** This criterion represents the ability to check whenever a file has been changed that the developer server recompiles the project and serves the changes to the browser. Such a features saves a lot of time because there is no need to manually recompile the project or restart the server to see the changes.
4. **GUI/UI:** Some developer strongly prefers a GUI in comparison to a simple user interface on the command line. If there is a GUI like a web interface or similar available, the framework may get accepted by more developers.
5. **Ease of Export:** After a Web Component is finished it needs to be exported that it can be used in other projects. This criterion represents how easy or hard this task is compared to the other frameworks.

Styling

This category is needed because the question, if a framework only supports plain CSS or if the advantage of CSS Preprocessing frameworks can be used, would not fit to any other category.

1. **CSS:** Does the framework support standard CSS. All Web Component frameworks should support plain CSS, this criterion is needed because we have to compare it with the following CSS Preprocessing criterion.
2. **CSS Preprocessing:** Gets a higher score depending on how many preprocessing frameworks the framework includes. Possible frameworks would be Sass/SCSS or Less.

All criteria are sorted into categories as shown in 5.1. In the first level of the Criteria are *Features*, *Styling* and *User Experience* as categories. The White criteria are on the second level and the gray criteria are not considered for the evaluation.

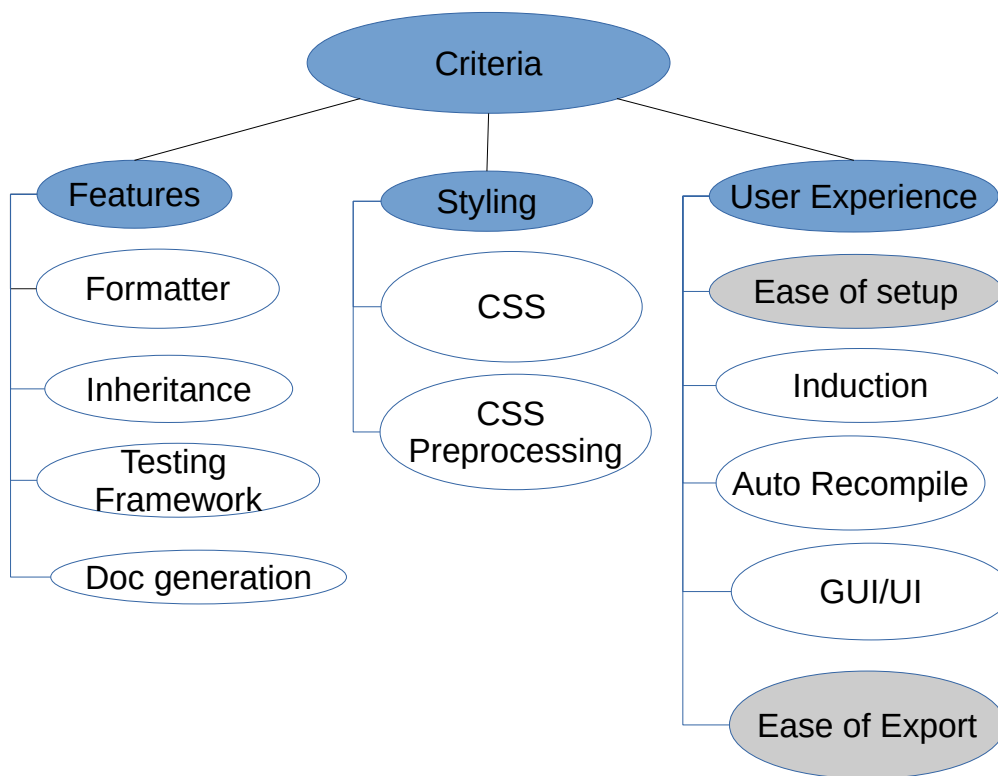


Figure 5.1: Criteria with their categories. The gray soft criteria are not evaluated

For the selection process specific criteria are set to decide which frameworks are evaluated in Chapter 4.

The Framework should be able to build the Web Component as **stand-alone JavaScript Web Components** so that it can be used on all platforms. The second criteria is that the framework should have a minimum of **5k stars on GitHub** because this is a good identifier that the framework is used by many developers.

5.2 Evaluation of the Frameworks

In this section the three frameworks Polymer, Stencil and Vue are compared based on each criterion. The results of the evaluation could differ based on of which aspect the frameworks are evaluated. At the end of each criterion, Polymer, Stencil and Vue get ranked based on how well they meet the criterion. This rank is later used during the evaluation of the alternatives with AHP in Section 5.3

5.2.1 Inheritance

With **Polymer** the inheritance is quite easy. To inherit from a custom element, the `.js` file must be included and the class extended by this element name as in Listing 5.1. Extending from a built-in element such as the standard `<button>` element is not supported now because not all browsers support this standard.

In **Vue** the use of an inheritance could be recreated with Mixins. It is not possible to extend another Web Component, but some code can be included from a Mixin into multiple Web Components.

Stencil is a bit behind because it currently does not support the inheritance. There is an open issue on GitHub ¹. Polymer fulfills the criterion the most, Vue is a bit behind but supports an alternative and Stencil does not meet the criterion at all.

```
1 | import {MyElement} from './my-element.js';
2 |
3 | export class ExtendedElement extends MyElement {
4 |   static get is() { return 'extended-element'; }
5 |
6 |   static get properties() {
```

¹<https://github.com/ionic-team/stencil/issues/1060>, last viewed 15 Mai 2019


```

7   return {
8     thingCount: {
9       value: 0,
10      observer: '_thingCountChanged'
11    }
12  }
13  }
14  _thingCountChanged() {
15    console.log(`thing count is ${this.thingCount}`);
16  }
17  };
18
19  customElements.define(ExtendedElement.is, ExtendedElement);

```

Listing 5.1: Inheritance example of the Polymer documentation [13]

Rank 1	Polymer
Rank 2	Vue
Rank 2	Stencil

Table 5.1: Ranking of Inheritance

5.2.2 Linter/Formatter

Stencil does not want to support a third-party application for linting referring to a GitHub issue ².

Vue supports Linter/Formatter as described in 4.3.1.

Polymer has a linter included for Polymer 1, 2 and 3.x. The linting process has to be executed through the Polymer-CLI. It uses its own rule set which focuses on Web Components and Polymer.

The rules (polymer-1, polymer-2 or polymer-3) has to be passed to the CLI either with the `--rules` flag or it can be set in the `polymer.json` like in Listing 5.2

²<https://github.com/ionic-team/stencil/issues/163>, last viewed 15 Mai 2019

5 Evaluation

```
1 {  
2   "lint":  
3   {  
4     "rules": ["polymer-3"],  
5     "ignoreWarnings": []  
6   }  
7 }
```

Listing 5.2: Lint Rules of Polymer 3

Rank 1	Vue
Rank 2	Polymer
Rank 3	Stencil

Table 5.2: Ranking of Linter/Formatter

5.2.3 Documentation Generation

The documentation generation for **Vue** is currently a Work in Progress. The not finished tool is called VuePress³. **Polymer** has a good-looking HTML documentation which can be generated as in 4.1 described. This documentation is very detailed without the need that the developers write much comments.

Stencil can generate a documentation of methods, props in ASCII style which can be used for e.g. a README.md as shown in 4.4. Depending on which kind of docs should be generated, the choice is either Stencil or Polymer. The ASCII documentation seems to fit a bit better because a short documentations often can give a better overview as very detailed documentations.

Rank 1	Stencil
Rank 2	Polymer
Rank 3	Vue

Table 5.3: Ranking of Documentation Generation

³<https://vuepress.vuejs.org/guide/#todo>, last viewed 15 Mai 2019

5.2.4 Testing Frameworks

Polymer has a lot of testing frameworks. Included are Mocha, Chai, Sinon, Selenium and Accessibility Developer Tools. Because of this wide integration of testing frameworks Polymer has a good support of testing frameworks.

In **Vue** are Jest and Mocha integrated which both aim at unit testing. This is enough for testing the frameworks, but the choice is a bit restricted due to only two integrated frameworks.

Stencil has Jest and Puppeteer integrated. Jest is used for unit testing while Puppeteer can be used if a browser environment end-to-end test is needed.

Rank 1	Polymer
Rank 2	Stencil
Rank 3	Vue

Table 5.4: Ranking of integrated Testing Frameworks

5.2.5 Induction

The **Polymer** framework has a good tutorial which inducts into its capabilities. The rest of the documentation is well structured and does not need any extra explanation which it is why it is easy to get started with Stencil.

At the **Vue** documentation on the website it is sometimes a bit hard to extract the wanted information which maybe is because Vue is a big framework. It is not easy to present a huge amount of information and structure it that users can find it fast. For this framework definitively more time needs to be investigated to get into it.

The **Stencil** documentation is well structured. A developer gets guided through the creation of a sample Web Component which makes it easy to get familiar with the capabilities and features Stencil has. The navigation through the documentation is intuitive and makes the induction easy.

Rank 1	Stencil
Rank 2	Polymer
Rank 3	Vue

Table 5.5: Ranking of Induction difficulty

5.2.6 Auto Recompile

Auto recompile is supported by all three frameworks in its core but there are some differences in detail which can be compared.

Vue and **Stencil** both supports hot reload which serves the changes after saving (or eventually recompiling) a file. The changes are visible immediately without having to reload the whole application. This makes it easier especially for some work on the design because there is no need to reload the page to see the effect after each change.

In **Polymer** this does not work because they have no hot reload included and don't plan to include such a feature soon as a contributor replied to a GitHub issue ⁴.

Rank 1	Vue
	Stencil
Rank 2	Polymer

Table 5.6: Ranking of Auto Recompile

5.2.7 GUI/UI

Through the command line, all three frameworks can be controlled, which includes a user interface e.g. to select what kind of project should be initialized.

In addition, **Vue** has a graphical user interface which can be accessed through the web browser. This GUI can display some additional information in a structures way as seen on Figure 4.5 which is a big advantage.

⁴<https://github.com/Polymer/polymer/issues/5425>,last viewed at 15 Mai 2019

Rank 1	Vue
Rank 2	Stencil
	Polymer

Table 5.7: Ranking of GUI/UI

5.2.8 CSS

All three frameworks do support CSS either as an external file or only included in the Web Component file. The need for CSS support is that a custom style can be added to a Web Component.

All three frameworks spare Rank 1 because they support the CSS standard.

Rank 1	Stencil
	Polymer
	Vue

Table 5.8: Ranking of CSS

5.2.9 CSS Preprocessing

In **Vue** Some preprocessors like Sass/Less or Stylus can be selected during the initialization of a project, they are already included. For **Stencil** is an official plugin available which can be included as on their website described. The only available preprocessing language which is supported is limited to SASS in this case. The **Polymer** framework only allows nested CSS styling and does not support any preprocessing CSS frameworks by default. Although it is possible to build a setup with it could work, Polymer gets the latest rank because of the lack of its integration.

Rank 1	Vue
Rank 2	Stencil
Rank 3	Polymer

Table 5.9: Ranking of CSS Preprocessor support

5.3 AHP

The analytical hierarchy process is an advanced multi criteria decision process which was developed by Thomas Thomas L. Saaty. Saaty tries to reduce the complexity of the problem by setting up a hierarchical structure. This proceed can be divided into four steps according to Saaty's paper [12]:

1. Defining the problem.
2. Structure the problem from the top (goal) over the criteria of different levels to the alternatives at the bottom.
3. Pairwise comparison between the elements of the same level.
4. Use the priorities obtained from the comparison to weigh the priorities below.

The problem gets well-structured because the tree hierarchy can have multiple levels. Each criterion of the same category and level are evaluated by choosing which criterion is more important to reach the goal. To do so the criteria are compared against each other and one has to decide how much more he likes criterion a compared to criterion b or if they are equal. That the results are better comparable the scale is between 1 and 9. The exact meaning of each value is shown in Table 5.10.

Intensity of Importance	Definition	Explanation
1	Equal Importance	Two activities contribute equally to the objective
2	Weak or slight	
3	Moderate Importance	Experience and judgment slightly favour one activity over another.
4	Moderate plus	
5	Strong importance	Experience and judgment strong favour one activity over another.
6	Strong plus	
7	Very strong or demonstrated importance	An activity is favoured very strongly over another; its dominance demonstrated in practice
8	Very, very strong	
9	Extreme importance	The evidence favouring one activity over another is of the highest possible order of affirmation
Reciprocals of above	If activity i has one of the above non-zero numbers assigned to it when compared with activity j, then j has the reciprocal value when compared with i	

Table 5.10: Fundamental scale of absolute numbers [12]

5.3.1 Criteria Weights

The goal of this decision is the *Framework Selection*. To get a better structure, the criteria for the evaluation are divided into the categories *Features*, *Styling* and *User Experience*.

The hierarchy tree for the weight of the strong criteria of Section 5.1.5 is shown in Figure 5.1. To reduce the complexity of this figure, the strong criteria are included in the categories.

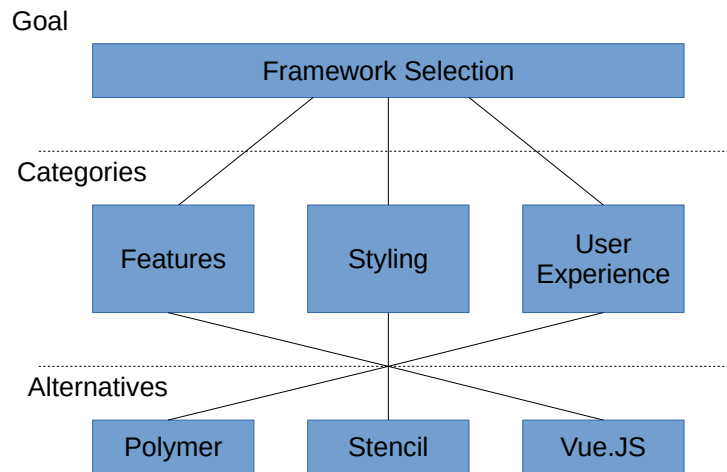


Figure 5.2: Tree Hierarchy of the Framework Selection without criteria

After the criteria of the evaluation are chosen, they must be associated with some weights. This step is mandatory because each criterion should only have an impact based on its importance and not an equal impact as all the other criteria. The weight of the criteria represents a specific scenario and can be different if this changes. This can be seen in the evaluation of use cases in section 5.4.

5.4 Use Cases

In this Section are some use cases which are evaluated and as a result the one which fits best of the three alternatives (Stencil, Polymer, Vue) is proposed.

To get started with AHP the categories and criteria needs priorities. This can be done with the online tool BPMSG⁵. This online tool checks if the choices of which criteria or alternative is better or equal than another, are consistent and displays the CR (consistency ratio) in % and only accepts accurate votes.

Depending on the project size the requirements can be different. In the following sections are some sample projects with different priorities on their criteria for the decision making.

5.4.1 Small Project

An example for a small project could be a static telephone book such as in Figure 5.3. Because a telephone book can be reused several times, it is a good idea to write it as a Web Component with a framework. In this case three frameworks are used for this telephone book. To find the best one, one has to decide which criteria are important. The importance of the different criteria is a bit subjective so it could be that this vary from person to person. One possible resulting decision hierarchy and its priorities are shown in Figure A.4. In each Level are the local priorities respective to its level in the hierarchy listed with a green background color. On the right side are the global priorities for each criterion. The darker the green background of a priority is, the more important is this criterion.

The next step is to combine the Decision History with the already discussed ranking of the frameworks for each criterion (subsection 5.1.5). The result is close, but Stencil is on rank 1 with 35.4%, Polymer on Rank 2 with 32.9% and Vue ends on rank 3 with 31.6% as on Figure 5.4.

⁵<https://bpmsg.com/academic/>, last viewed 15 Mai 2019

5 Evaluation

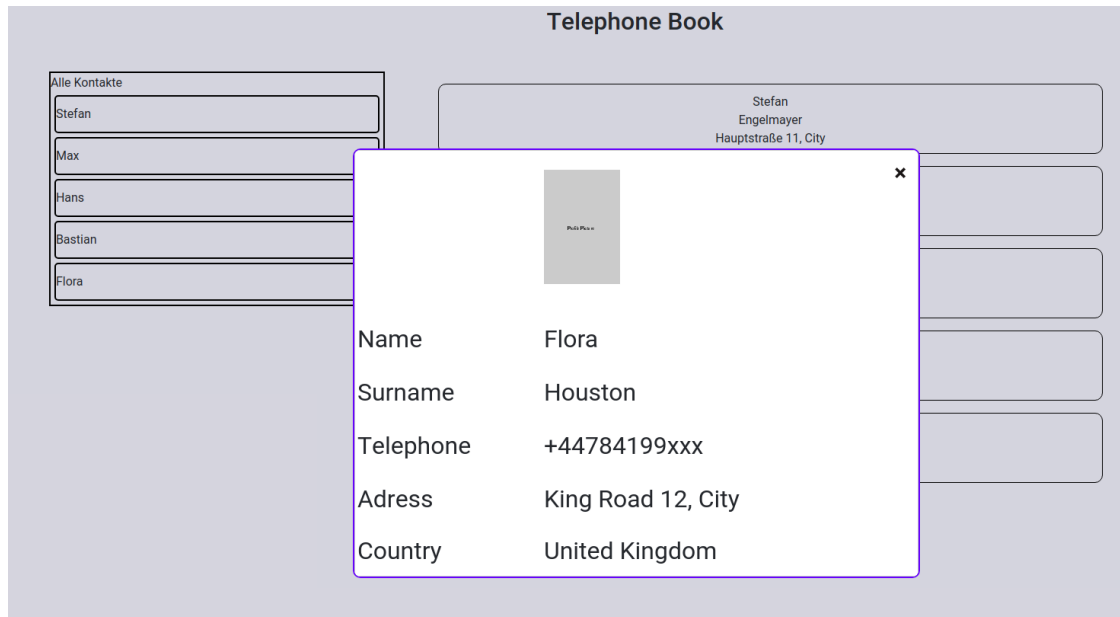


Figure 5.3: An Example Project of a Telephone Book using Web Components.

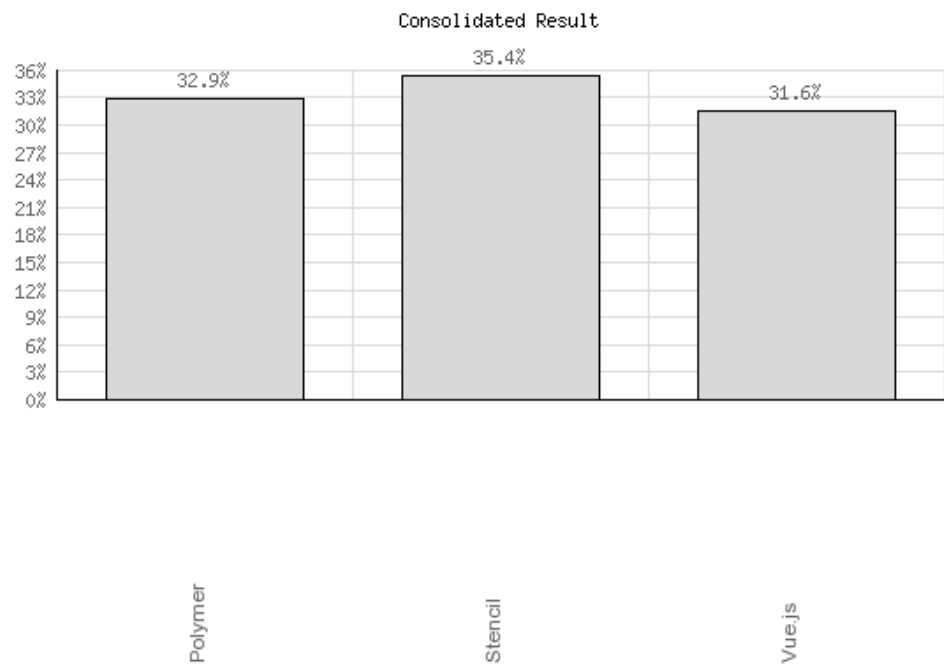


Figure 5.4: Rank of the Alternatives for this small project

5.4.2 Medium Project

A further use case could be a calendar which allows to add, modify and delete some events. Therefore, more functionality has to be implemented into the Web Component. The requirements change a little bit different and the priorities changes.

The estimated priorities for such an project are shown in Figure A.5. As seen in this figure, not only the priorities of the criteria itself has changed but also the priorities of the categories (Features, Styling, User Experience).

Compared to the small project the priorities do not significantly change. As a result, the Stencil Framework in the first choice with 34.7%, Polymer with 24.8% on rank 2 and Vue is on rank 3 with 32.4% as shown in Figure 5.5.

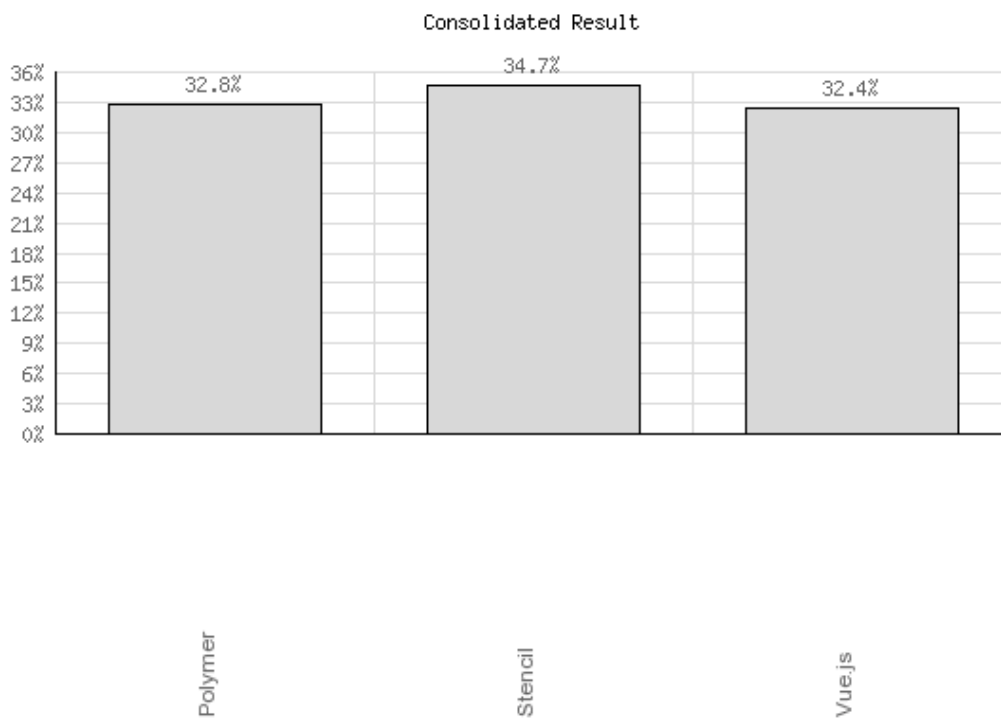


Figure 5.5: Possible Decision Hierarchy for a large project

5.4.3 Large Project

For the large project the priorities must be calculated again because the importance of the criteria changes. As an example, the large project is about writing a Web Component which is an editor for Java code with syntax highlighting and the ability to load and exchange the code between the editor and a backend for further processing. Because the size is increasing the importance of a formatter, CSS Preprocessing and Auto Recompile are increasing. Other criteria are noticeable lower in importance such as the Doc generation, Induction and GUI/UI.

As shown in Figure A.6 the Formatter, CSS Preprocessing and Auto Recompile criteria have a higher priority and as a result the importance of the Doc generation, Induction and GUI/UI is lower.

The next step is to combine the Decision History with the already discussed ranking of the frameworks for each criterion (subsection 5.1.5). The result is not as close as the small project. For this large project the decision is quite clear. Vue is on rank 1 with 40.7%, Polymer on Rank 2 with 30.3% and Stencil ends on rank 3 with 31.6% as shown in Figure 5.6. Vue offers the most desired functionality and is with a minimum of 10% ahead of the other frameworks.

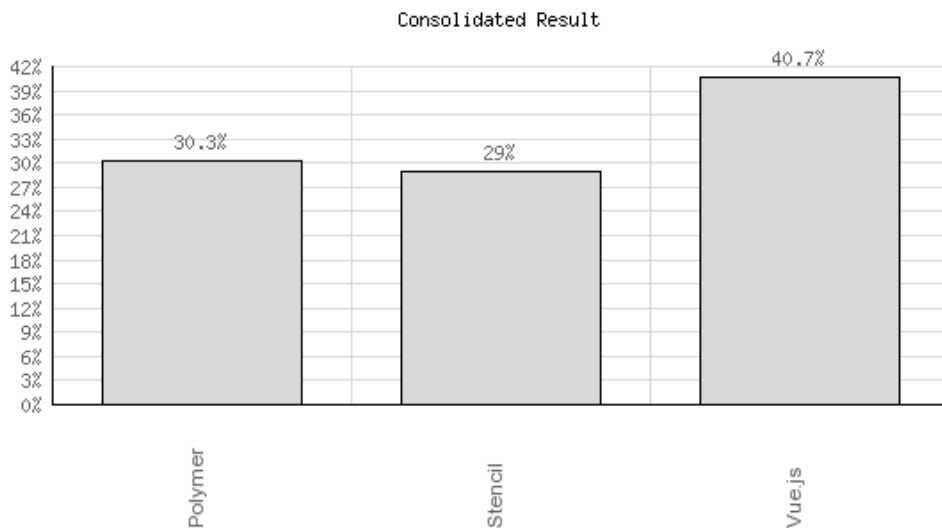


Figure 5.6: Rank of the Alternatives for the sample large project

5.5 Conclusion

After this evaluation it is possible to say that there is no overall framework which the best for all use cases is. For the small and medium project Stencil ranked first with Polymer and Vue only close behind, Vue dominated the ranking for the large project.

Although for small and medium projects Stencil is on the first rank, all three frameworks are a good choice because they cover the required functionality. As a result, if the priorities only get changed a bit, the result changes that another framework is on the first rank. Referring to the result of the large project, the decision is quite clear. Vue has the most features which makes it a good choice. Especially if some criteria are changed, the decision should be Vue because it is far ahead of Stencil and Polymer.

6

Summary

Web Component frameworks are a good tool to help developers with the creation of Web Components. There are many various frameworks available and if a developer wants to start a project, he first needs to check which framework should be chosen for his project. With the result of this paper the decision should be easier. The introduced frameworks are all good candidates for the development if the project size is small or medium. Only for bigger projects is a significant difference in the selection of the framework notable.

As in the field of computer science everything changes fast, these decisions are not final. To make further evaluations easier, it is also possible to take the used criteria as a pattern for another evaluation of Web Component frameworks. Not all frameworks are introduced in this thesis because only selected frameworks are introduced and considered for the evaluation. It is possible to do the research again under different criteria for the framework selection or further use cases.

In this thesis are the importance of the criteria estimated. So it is also possible to make these decisions more accurate and evaluate the medium and large sized projects again with projects as a reference. Based on this thesis, the accuracy of the estimated weights of the criteria can be increased if multiple people submit their choice on the online calculator BPMSG ¹. Then the average of the weights can be used for the evaluation. Another approach of the evaluation can be considered. Although the analytical hierarchy process is a well-structured evaluation method, it is possible that other approaches can lead to a different result.

¹<https://bpmsg.com/academic/>, last viewed 15 Mai 2019

Bibliography

- [1] chrisdavidmills, bminard, b.o.M.s.s.j.a.E.B.s.e.d.S.T.S.a.n.d.j.z.i.S.J.c.B.D.f.O.w.j.m.r.t.t.S.m.k.: Web components | mdn. https://developer.mozilla.org/en-US/docs/Web/Web_Components (2019) last viewed 15 Mai 2019.
- [2] mdnwebdocs bot, bminard, m.s.v.n.a.c.E.B.b.S.D.a.j.u.s.: Using shadow dom - web components | mdn. https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_shadow_DOM (2019) last viewed 15 Mai 2019.
- [3] Hixie, I.H.G.U.: Html standard - custom elements. <https://html.spec.whatwg.org/multipage/custom-elements.html#custom-elements> (2019) last viewed 15 Mai 2019.
- [4] Bidelman, E.: Custom elements - rules by developers.google.com. <https://developers.google.com/web/fundamentals/web-components/customelements> (2019) last viewed 15 Mai 2019.
- [5] Schlueter, I.Z.: Npm docs. <https://docs.npmjs.com/about-npm/> (2019) last viewed 15 Mai 2019.
- [6] Ayağ, Z., Özdemir, R.G.: A fuzzy ahp approach to evaluating machine tool alternatives. *Journal of Intelligent Manufacturing* **17** (2006) 179–190
- [7] Schwer, D.: Strukturierter vergleich aktueller frameworks zur entwicklung mobiler anwendungen. Bachelor's thesis, Universität Ulm, <http://dbis.eprints.uni-ulm.de/1586/> (2019) last viewed 15 Mai 2019.
- [8] Keller, R.: Vergleich von MV* Frameworks der Skriptsprache JavaScript. Bachelor's thesis, Hochschule Mittweida, https://monami.hs-mittweida.de/frontdoor/deliver/index/docId/8480/file/Bachelorarbeit_Normal_Farbe_RobertKeller_M31567.pdf (2016) last viewed 15 Mai 2019.

Bibliography

- [9] Kornher, M.: Analyse und Vergleich von WebGLFrameworks. Bachelor's thesis, Eberhard Karls Universität Tübingen, https://moritzkornher.de/files/kornher_analyse_und_vergleich_von_webgl-frameworks.pdf (2013) last viewed 15 Mai 2019.
- [10] katejeffreys, G.U.: Polymer docs. <https://polymer-library.polymer-project.org/3.0/docs/> (2019) last viewed 15 Mai 2019.
- [11] Josh Thomas, Jakub Mikulas, K.P.A.B.: Stencil website - getting started. <https://stenciljs.com/docs/my-first-component/> (2019) last viewed 15 Mai 2019.
- [12] Saaty, T.L.: Decision making with the analytic hierarchy process. *International journal of services sciences* **1** (2008) 83–98
- [13] katejeffreys, A.E.: Polymer devguide custom elements. <https://polymer-library.polymer-project.org/3.0/docs/devguide/custom-elements> (2019) last viewed 15 Mai 2019.

A

Sources

In this chapter are the inputs of the online AHP application.

After the decision tree without weights, for each framework are two screenshots included. The first has the Weights for each criteria included, the second has the local weights for each alternative.

```
1 Framework Selection: Features, Styling, User Experience;  
2 Features: Formatter, Inheritance, Testing Framework, Doc generation;  
3 Styling: CSS, CSS Preprocessing;  
4 User Experience: Induction, Auto Recompile, GUI/UI;
```

Listing A.1: Decision Tree without Weights

```
1 Framework Selection: Features=0.45454545, Styling=0.09090909,  
2 User Experience=0.45454545;  
3 Features: Formatter=0.10960877, Inheritance=0.38365255,  
4 Testing Framework=0.07715541, Doc generation=0.42958326;  
5 Styling: CSS=0.5, CSS Preprocessing=0.5;  
6 User Experience: Induction=0.41260224, Auto Recompile=0.32747652,  
7 GUI/UI=0.25992124;
```

Listing A.2: Decision Tree with Weights for the Small Project

A Sources

1. Alternatives with local weights (Weighted sum method)					
Crit/Alt	pGlb	Polymer	Stencil	Vue.js	
Formatter	0.049822	0.352205	0.088744	0.55905	
Inheritance	0.174388	0.55905	0.088744	0.352205	
Testing Framework	0.035071	0.683325	0.199817	0.116858	
Doc generation	0.195265	0.344559	0.546919	0.108523	
CSS	0.045455	0.333333	0.333333	0.333333	
CSS Preprocessing	0.045455	0.09739	0.333072	0.569539	
Induction	0.187546	0.333072	0.569539	0.09739	
Auto Recompile	0.148853	0.142857	0.428571	0.428571	
GUI/UI	0.118146	0.166667	0.166667	0.666667	
Group Result		0.329284	0.35429	0.316425	
2. Alternatives by participant					
	Name	Polymer	Stencil	Vue.js	CR max
	Group	0.329284	0.35429	0.316425	0.055958
	Author	0.329284	0.35429	0.316425	0.055958

Figure A.1: Snippet of the .csv of the Small Project with its Weights and Priorities

```

1 Framework Selection: Features=0.44444444, Styling=0.11111111,
2 User Experience=0.44444444;
3 Features: Formatter=0.17515158, Inheritance=0.30677294,
4 Testing Framework=0.13458121, Doc generation=0.38349427;
5 Styling: CSS=0.5, CSS Preprocessing=0.5;
6 User Experience: Induction=0.41260224, Auto Recompile=0.32747652,
7 GUI/UI=0.25992124;

```

Listing A.3: Decision Tree with Weights for the Medium Project

1. Alternatives with local weights (Weighted sum method)					
Crit/Alt	pGlb	Polymer	Stencil	Vue.js	
Formatter	0.077845	0.352205	0.088744	0.55905	
Inheritance	0.136344	0.55905	0.088744	0.352205	
Testing Framework	0.059814	0.683325	0.199817	0.116858	
Doc generation	0.170442	0.344559	0.546919	0.108523	
CSS	0.055556	0.333333	0.333333	0.333333	
CSS Preprocessing	0.055556	0.09739	0.333072	0.569539	
Induction	0.183379	0.333072	0.569539	0.09739	
Auto Recompile	0.145545	0.142857	0.428571	0.428571	
GUI/UI	0.115521	0.166667	0.166667	0.666667	
Group Result		0.328293	0.347271	0.324436	
2. Alternatives by participant					
	Name	Polymer	Stencil	Vue.js	CR max
	Group	0.328293	0.347271	0.324436	0.055958
	Author	0.328293	0.347271	0.324436	0.055958

Figure A.2: Snippet of the .csv of the Medium Project with its Weights and Priorities

```

1 Framework Selection: Features=0.42857143, Styling=0.14285714,
2 User Experience=0.42857143;
3 Features: Formatter=0.34751743, Inheritance=0.38284545,
4 Testing Framework=0.14202197, Doc generation=0.12761515;
5 Styling: CSS=0.2, CSS Preprocessing=0.8;
6 User Experience: Induction=0.1743726, Auto Recompile=0.63370595,
7 GUI/UI=0.19192145;

```

Listing A.4: Decision Tree with Weights for the Large Project

1. Alternatives with local weights (Weighted sum method)					
Crit/Alt	pGlb	Polymer	Stencil	Vue.js	
Formatter	0.148936	0.352205	0.088744	0.55905	
Inheritance	0.164077	0.55905	0.088744	0.352205	
Testing Framework	0.060867	0.683325	0.199817	0.116858	
Doc generation	0.054692	0.344559	0.546919	0.108523	
CSS	0.028571	0.333333	0.333333	0.333333	
CSS Preprocessing	0.114286	0.09739	0.333072	0.569539	
Induction	0.074731	0.333072	0.569539	0.09739	
Auto Recompile	0.271588	0.142857	0.428571	0.428571	
GUI/UI	0.082252	0.166667	0.166667	0.666667	
Group Result		0.302671	0.290108	0.407221	
2. Alternatives by participant					
	Name	Polymer	Stencil	Vue.js	CR max
	Group	0.302671	0.290108	0.407221	0.055958
	Author	0.302671	0.290108	0.407221	0.055958

Figure A.3: Snippet of the .csv of the Large Project with its Weights and Priorities

Decision Hierarchy			
Level 0	Level 1	Level 2	Glb Prio.
Framework Selection	Features 0.455	Formatter 0.110	5.0%
		Inheritance 0.384	17.4%
		Testing Framework 0.077	3.5%
		Doc generation 0.430	19.5%
	Styling 0.091	CSS 0.500	4.5%
		CSS Preprocessing 0.500	4.5%
	User Experience 0.455	Induction 0.413	18.8%
		Auto Recompile 0.327	14.9%
		GUI/UI 0.260	11.8%
			1.0

Figure A.4: Decision Hierarchy for the sample project 'Telephone Book'

Decision Hierarchy			
Level 0	Level 1	Level 2	Glb Prio.
Framework Selection	Features 0.444	Formatter 0.175	7.8%
		Inheritance 0.307	13.6%
		Testing Framework 0.135	6.0%
		Doc generation 0.383	17.0%
	Styling 0.111	CSS 0.500	5.6%
		CSS Preprocessing 0.500	5.6%
	User Experience 0.444	Induction 0.413	18.3%
		Auto Recompile 0.327	14.6%
		GUI/UI 0.260	11.6%
			1.0

Figure A.5: Possible Decision Hierarchy for a large project

Decision Hierarchy			
Level 0	Level 1	Level 2	Glb Prio.
Framework Selection	Features 0.429	Formatter 0.348	14.9%
		Inheritance 0.383	16.4%
		Testing Framework 0.142	6.1%
		Doc generation 0.128	5.5%
	Styling 0.143	CSS 0.200	2.9%
		CSS Preprocessing 0.800	11.4%
	User Experience 0.429	Induction 0.174	7.5%
		Auto Recompile 0.634	27.2%
		GUI/UI 0.192	8.2%

Figure A.6: Possible Decision Hierarchy for a large project

List of Figures

2.1	Redawn Shadow Tree inside the DOM from [2]	6
2.2	Clean <audio> Tag which has to be written by the developer	7
2.3	Audio element with a part of it's Shadow DOM expanded	7
2.4	HTML template for a list of users	9
2.5	Code to use the HTML template and fill it with data	9
2.6	The rendered page	9
4.1	Generated doc for the properties.	16
4.2	Generated doc for some methods.	17
4.3	Structure of a simple Polymer-3 application	18
4.4	Auto Generated Readme.md with a List of Used Properties.	20
4.5	UI Shows Serving a Project	22
4.6	Feature selection with the Vue CLI.	23
5.1	Criteria with their categories. The gray soft criteria are not evaluated	31
5.2	Tree Hierarchy of the Framework Selection without criteria	40
5.3	An Example Project of a Telephone Book using Web Components.	42
5.4	Rank of the Alternatives for this small project	42
5.5	Possible Decision Hierarchy for a large project	43
5.6	Rank of the Alternatives for the sample large project	44
A.1	Snippet of the .csv of the Small Project with its Weights and Priorities	52
A.2	Snippet of the .csv of the Medium Project with its Weights and Priorities	52
A.3	Snippet of the .csv of the Large Project with its Weights and Priorities	53
A.4	Decision Hierarchy for the sample project 'Telephone Book'	54
A.5	Possible Decision Hierarchy for a large project	54
A.6	Possible Decision Hierarchy for a large project	55

List of Tables

5.1	Ranking of Inheritance	33
5.2	Ranking of Linter/Formatter	34
5.3	Ranking of Documentation Generation	34
5.4	Ranking of integrated Testing Frameworks	35
5.5	Ranking of Induction difficulty	36
5.6	Ranking of Auto Recompile	36
5.7	Ranking of GUI/UI	37
5.8	Ranking of CSS	37
5.9	Ranking of CSS Preprocessor support	37
5.10	Fundamental scale of absolute numbers [12]	39

Name: Stefan Engelmayer

Matriculation number: 869566

Honesty disclaimer

I hereby affirm that I wrote this thesis independently and that I did not use any other sources or tools than the ones specified.

Ulm,

Stefan Engelmayer