Proprietors Assembly / Hausversammlung

Final Year Project

Alexander Gogl

Supervisor: Chris Casey

Course: BsC (Hons) Computing

University of Central Lancashire

25th February 2019

Contents

Abstract

Chapter 1
Literature Review

1.1 History of Vue.js

Vue was created by Evan You whilst working at Google on a UI-heavy project [5]. Prior to the creation of Vue, there were no frameworks for rapidly prototyping UI interfaces. Subsequently, You, who did not find any viable solutions to his problem started working on his own. Angular, already widely used by developers and created by Google itself was too big and bloated of a framework to be sensible for small applications. React, a fairly new framework at the time also proved being too complex just like Backbone.js which was used for large-scale enterprise applications. None of these frameworks were adequate for prototyping [3, p. 10]. You's approach of reactive data-binding and reusable components later named Vue.js, was realeased in February 2014 [5] and helped fill that niche. Gradually improved over the years Vue is now utilized for complex enterprise-grade applications and small prototpyes alike and has since been adopted by many developers and companies around the world. The most noteworthy of which are: Facebook, Netflix, Adobe, Xiaomi, Alibaba and GitLab [12]. With more than 130,000 stars on GitHub at the time of this writing, Vue is more popular than both React (122,000 stars) and Angular (45,000 stars).

1.2 Comparison to other Frameworks

As there is an excessive amount of libraries and frameworks available for creating web based applications this section will be limited to comparing Vue to the other most popular frameworks React and Angular.

Glossary:

Basic Usage: How this particular framework / library is typically used with a bundler / build process

Scaling: scaling in terms of functionality rather than how many developers are actively working on a project or the number of end users using the application.

Flexibility of a framework / library: flexibility in terms of how opinionated it is

1.2.1 Vue 2.6.X

Basic Usage

Vue.js is a JavaScript framework that "makes it easier for developers to create rich, interactive websites" [5] which is based on a virtual DOM [9]. A virtual DOM is a set of instructions that are followed when a user interface is updated [1, p. 81] creating a new layer between the Browsers DOM and Vue, drastically increasing efficiency in the process [1]. By default, any valid HTML can be used to define the basic structure of Vue components, which is also referred to as a "template" in Vue terminology. Component-scoped data and logic are defined within script tags

and are complemented by functional directives such as "v-if" inside the template. Style sheets are defined within style tags, separating style and logic. By optionally using the scoped keyword, CSS is bound to a specific component, reducing the risk of polluting global style sheets. The composition of template, script and style tags is called a "Single File Component", indicating that only a single file is needed to create a functional and styled component. Additionally, props provide a uni-directional way of passing data from a parent component to a child component, whereas events are commonly used to pass data in the opposite direction. "Methods" can be defined to execute repeatable code, "computed properties" are a powerful way of reflecting changes of data and "watchers" behave like computed properties but only execute code upon data changes without returning data themselves.

```
<template>
<div>
<h1 v-if="condition" class="custom-title"> Hello {{ user }} </h1>
<h1 v-else> "Hello Stranger" </h1>
</div>
</template>

</script>
export default {
    data: {
        return() {
            user: "Alex"
        }
    },
    methods: {/*execute code*/},
    computed: {/*return data when data changes*/},
    watch: {/*execute code when data changes*/}
}
</script>

</style scoped>
.custom-title {
    size: 30px;
}
</style>
```

Listing 1.1: Vue Single File Component

Scaling

Vue applications can easily be scaled up or down in terms of application functionality depending on the requirements. In order to create sophisticated and large applications, three parts have to be incorporated in general: The core library, routing and state management, of which all are officially provided by Vue as supporting libraries [9]. This typically goes along with bundling tools such as Webpack or Browserify, which make for a much more powerful development environment. In addition, Vue offers an optional CLI generator interface for scaffolding projects, leaving the choice to the developer which building system and plugins to use. Using this interface, sophisticated applications along with routing and state management can be created. If the goal is to add reactive and interactive elements to an existing webpage, Vue can be used by adding a script tag to to any valid site where necessary [7, 9]. In doing so, no bundler is needed for code to function but at the same time developers are deprived from being able to use plugins, preprocessors, and various other tools and are most commonly left with a larger bundle size [9].

Performance & Build Size

The source size of Vue with the additional dependencies of vuex (state-management) and vue-router is about 30KB [9]. As for performance, Vue resides in the upper category in terms of startup time, memory allocation and rendering time [11], due to the usage of a virtual DOM. Early beta versions of Vue 3.0 even reduce memory allocation and rendering time by about 50%.

Flexibility & Learning Curve

Among other things, Single File Components are easy to use when coming from an HTML background, make transitioning existing applications to Vue smoother, do not have a steep learning curve resulting in faster adoption by beginners and can be further enhanced with various preprocessors [9]. As Vue supports various bundlers and build systems while not enforcing specific ways of usage, you could argue that it is less opinionated than some other technologies.

1.2.2 React 16.8.X

Basic Usage

React describes itself as a "library for building user interface", which does not automatically come with as much functionality a traditional framework would have [1, p. 2]. Like Vue it also utilizes a virtual DOM [1, p. 81]. In React everything is defined in terms of JavaScript, meaning HTML often coupled with CSS are directly embedded into so called render functions. JSX, which is essentially a syntax extension to JavaScript [8] is most commonly used (even though not necessary) to define these render functions and is capable of mixing the full power of a programming language with rendering and UI logic [9]. Styling is most commonly achieved by CSS-in-JS solutions provided by additional libraries, effectively consolidating logic and styling in the same place [4].

```
1 class Welcome extends React.Component {
2     render() {
3         return <h1>Hello, {this.props.name}</h1>;
4     }
5 }
```

Listing 1.2: Usage of JSX Render Function

In addition, JSX render functions provide, full leverage of JavaScript, including temporary variables and direct references to these and good tooling support (linting, type checking, autocompletion) [8, 9].

Scaling

Like Vue React can either be used with a bundler or added to a single site by using a script tag. React outsources routing and state management to the community, fragmenting its ecosysytem in the process [9]. More than eleven well-known routing libraries are available for React with similar statistics for state management and styling which gives developers a vast variety of

available options but also makes choosing the right library for a projects requirements a much more tedious task. Reacts cli "create-react-app" works in a similar fashion like Vues but is much more limited: instead of letting the user choose a variety of options, it assumes that a single page application is to be created with always the same dependencies. However, an existing project can be migrated to a more customized environment with a command provided by React.

Performance & Build Size

The React source itself has about 4.7KB gzipped which makes sense, given that React advertises itself as a library rather than a framework. However, including React DOM (34KB), routing with react-router (6.9KB) and state management with redux (2.4KB) the total size grows considerably to 48KB. Performancewise, React can also be categorized as a library of the upper category [11] due to its usage of a virtual DOM.

Flexibility & Learning Curve

To take advantage of online resources and documentation JSX is almost a requirement. As it is an extension to JavaScript, developers who are familiar with this programming language can easily learn the additional syntax [8]. Without prior knowledge of JavaScript however, the learning curve rises considerably. The availability of various libraries makes React a very flexible library, given that the core of react is so slim and additional functionality is to be added by using those.

1.2.3 Angular 7.2.X

Basic Usage

Even though JavaScript could be used, Angular essentially requires the usage of TypeScript, a typed superset of JavaScript [13]. As opposed to JavaScript, TypeScript comes with static type checking which naturally makes applications less prone to errors [6] and therefore is often used within very large corporate projects. Like Vue Angular also uses a component-based approach for composing user interfaces, but rather than using a single file, multiple files for HTML, CSS and JavaScript logic are typically created.

```
1  //app.component.html
2  <button (click)="show = !show">{{show ? 'hide' : 'show'}}</button> show = {{
       show}}
3  <br>
4  <div *ngIf="show; else elseBlock">Text to show</div>
5  <ng-template #elseBlock>Alternate text while primary text is hidden</
       ng-template>
6
7  // app.component.ts
8  import { Component } from '@angular/core';
9
10 @Component({
11   selector: 'app-root',
12   templateUrl: './app.component.html',
13   styleUrls: ['./app.component.css']
14 })
15
```

```
16  export class SampleComponent {
17    show: boolean = true;
18  }
```

Listing 1.3: Angular Basic Usage Example

Scaling

Angular provides a lot of functionality out of the box. Designed with the purpose of building large and complex applications Angulars API exposes a lot of functionality which is most commonly only necessary for exactly these types of applications. It includes everything from routing, state management, http calls to complete testing suites, however, Angular can also be added to existing sites using a script tag, only providing core functionality.

Performance & Build Size

Angular applications built with the angular project scaffolding interface angular-cli have around 65KB gzipped, double the space as the other two frameworks. As for performance, Angular is also a performant framework [11] but becomes slow under certain circusmstances: For instance if a project uses a lot of watchers and data in the scope changes, all watchers are re-evaluated again [9].

Flexibility & Learning Curve

Angulars very big ecosystem and API provide most needed functionality out of the box but at the same time limit its flexibility. What is more, by prodiving pre-defined ways of interacting with Angular, it inherently becomes opinionated and more difficult to master [9].

1.2.4 Conclusio

All three frameworks solve similar problems but are used in different ways. Vues Single File Components consolidate logic, styles and HTML in the same place, React can and is mosten often used in a very resemblant way if used with a CSS-in-JS approach, whereas the "Angular way" is to split these parts into separate files (altough inlining is possible but not typical).

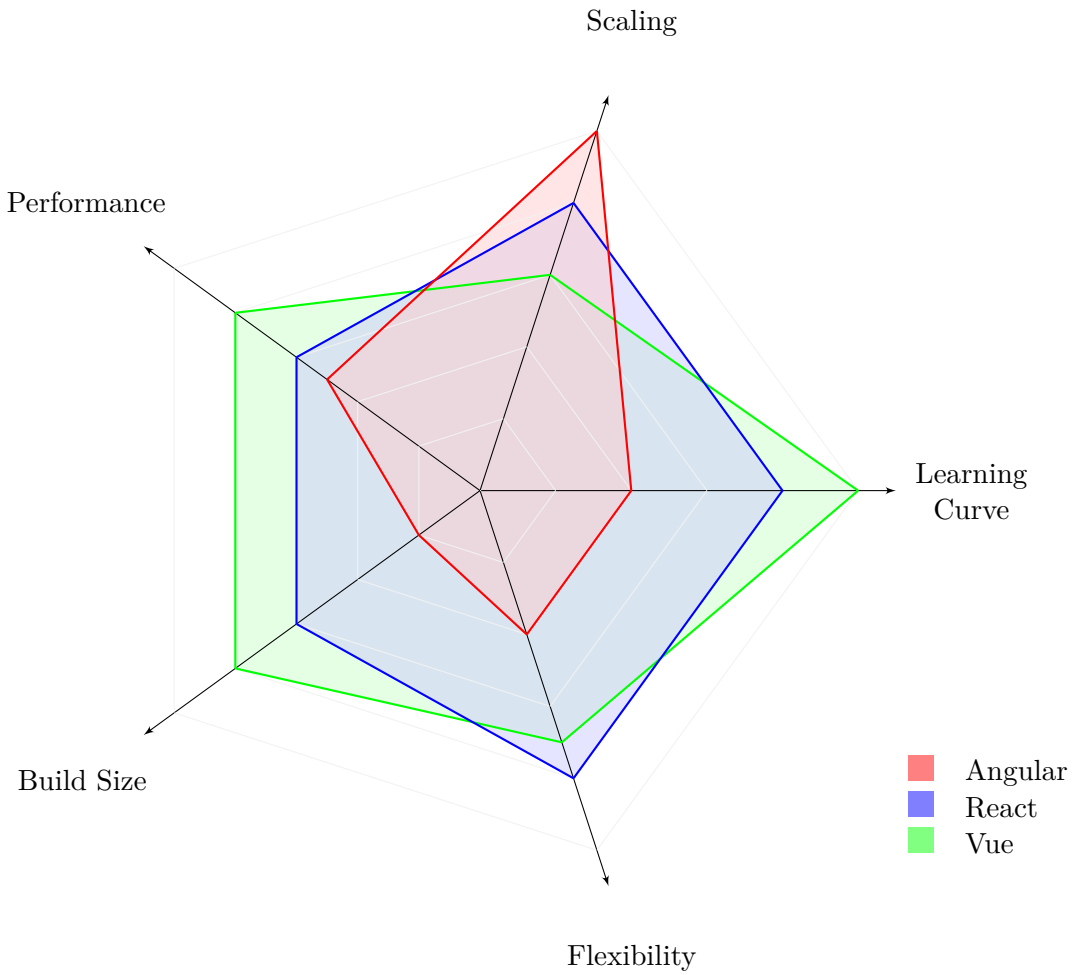
Applications built with React or Vue can be easily scaled up or down depending on the application-requirements. Large applications typically need the core library, routing and state management. While all of these are officially provided by the Vue core team for Vue applications, React leaves routing and state management to the community [9]. Vue and React both offer an optional CLI generator interface to scaffold projects, however, Vue offers more options, leaving the choice to the developer which building system and plugins to use. React's more limiting approach with create-react-app makes it easier to start a project as it only needs a single dependency but limits the user to a given setup, which can however, be moved to a more customized environment with little effort whereas Angular is completely set up for the development of very large projects and provides most functionality out of the box, scaling up is therefore not a big problem because developers can rely on most of the functionality already existing.

All three frameworks can be scaled down by adding script tags to any site [7, 9] which however removes the powerful building layer.

Vue and React both are resemblant in terms of runtime performance, are based on a virtual DOM and are used for similar use cases while Angular is a much more heavy-weight framework regarding performance and size.

JSX as well as TypeScript are additional learning steps and can lead to decreased productivity in smaller projects. Being a dynamic language, JavaScript's ability to address quickly changing requirements makes it especially suitable for rapid prototyping [6, p. 72]. By using very HTML-like components Vue is often easier to master than the other two frameworks as most developers have at least basic knowledge of HTML. Nonetheless, Vue makes it possible to use TypeScript if the need arises, paving the way for bigger projects [10]. Angular is not very flexible in the sense that users can pick whatever supporting library they deem best and is said to be an opinionated framework which enforces the "Angular way". Such frameworks are "pragmatic, with a strong sense of direction" [2] often forcing very specific conventions upon its users, effectively restricting what a developer can to with a framework. This also means there is a steeper learning curve but once overcome, can lead to increased productivity [9].

The following diagram does by no means reflect the properties of the given framework / library 100% accurately as some of them are somewhat subjective (e.g flexibility) but shall rather illustrate the strengths and weaknesses in relation to another when used in a typical manner.



1.3 SEO for Single Page Applications

Chapter 2
Project Planning

2.1 Methodology

2.2 Requirements

2.3 Potential Solutions

2.4 Tools and Techniques

2.5 Legal and Ethical Issues

Chapter 3
System Design

3.1 System Design

3.2 User Interface Design

3.3 Summary

Chapter 4
Implementation

Chapter 5
Test Strategy

5.1 Introduction

5.2 Testing Frontends

Describes which possibilites there are regarding frontend testing: Mainly unit and integration

5.3 Test Setup

Talks about vue-test-utils and jest integration

5.4 Continuous Integration

Gitlab CI / CD

Chapter 6
Deployment with Docker

6.1 Applicability of Containers

6.2 Docker

6.3 Docker Swarm

6.4 Microservices

6.5 Reverse Proxy

Chapter 7
Evaluation

7.1 Project Objectives

7.2 Self Evaluation

7.3 Project Evaluation

7.4 Commercial Applicability

7.5 Conclusions

7.6 Future Work

References

Literature

[1] Alex Banks and Eve Porcello. *Learning React: Functional Web Development with React and Redux*. " O'Reilly Media, Inc.", 2017 (cit. on pp. 1, 3).

[2] Kevin Bedell. 'Opinions on Opinionated Software'. In: *Linux J.* 2006.147 (July 2006), pp. 1–. URL: http://dl.acm.org/citation.cfm?id=1145562.1145563 (cit. on p. 6).

[3] Olga Filipova. *Learning Vue. js 2*. Packt Publishing Ltd, 2016 (cit. on p. 1).

[4] Naimul Islam Naim. 'ReactJS: An Open Source JavaScript Library for Front-end Development'. In: (2017) (cit. on p. 3).

[5] Callum Macrae. *Vue. js: Up and Running: Building Accessible and Performant Web Apps*. " O'Reilly Media, Inc.", 2018 (cit. on p. 1).

[6] Francisco Ortin and Miguel Garcia. 'A Programming Language That Combines the Benefits of Static and Dynamic Typing'. In: *Software and Data Technologies*. Ed. by José Cordeiro, Maria Virvou and Boris Shishkov. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 72–87 (cit. on pp. 4, 6).

Online sources

[7] React contributors. *Add React to a Website – React*. URL: https://reactjs.org/docs/add-react-to-a-website.html (visited on 18/02/2019) (cit. on pp. 2, 6).

[8] React contributors. *Rendering Elements – React*. URL: https://reactjs.org/docs/rendering-elements.html (visited on 18/02/2019) (cit. on pp. 3, 4).

[9] Vue contributors. *Comparison with Other Frameworks — Vue.js*. URL: https://vuejs.org/v2/guide/comparison.html (visited on 18/02/2019) (cit. on pp. 1–3, 5, 6).

[10] Vue contributors. *TypeScript Support — Vue.js*. URL: https://vuejs.org/v2/guide/typescript.html (visited on 18/02/2019) (cit. on p. 6).

[11] Stefan Krause. *Interactive Results*. URL: https://stefankrause.net/js-frameworks-benchmark8/table.html (visited on 20/02/2019) (cit. on pp. 3–5).

[12] Michał Sajnóg. *13 Top Companies That Have Trusted Vue.js – Examples of Applications | Netguru Blog on Vue*. Mar. 2018. URL: https://www.netguru.com/blog/13-top-companies-that-have-trusted-vue.js-examples-of-applications;%20http://gfs.sf.net/gerris.pdf (visited on 18/02/2019) (cit. on p. 1).

[13] *TypeScript - JavaScript that scales.* URL: `https://www.typescriptlang.org/` (visited on 19/02/2019) (cit. on p. 4).

List of Figures

Listings

List of Tables