Tech Review

CS 461 Fall 2018

Group 19 BrewHops: Ninkasi Brewing, Automating the brewing process

Dan Van Horn
vanhornd@oregonstate.edu

October 23, 2018

Abstract

The front-end of this project will be built using the most modern web-development practices. The frameworks which best represent that fact are reviewed as they pertain to the project needs. Special consideration is given to technologies that will reduce developer overhead, maximize the chances of completing our project goals ahead of schedule, and increase the maintainability of the application.

Contents

1 Introduction 1

2 Front End 1

2.1 React 1

2.2 Vue 1

2.3 Preact 1

3 Style 1

3.1 CSS 2

3.2 Less 2

3.3 SCSS 2

4 Bundling 2

4.1 Webpack 2

4.2 Parcel 2

4.3 Babel 3

5 Conclusion 3

References 3

1 INTRODUCTION

We are continuing development on an existing solution from a previous capstone group. It includes a small set of applications designed to manage brewing data automatically, allow for manual user editing, and generate graphic visualizations. This system will manage brewing data quickly, efficiently and correctly as it will reduce the possibility for human error. The applications will be hosted on a cloud provider and be both desktop and mobile compatible. Each developer role in the project is to be adaptable and familiar with the entire technology stack we are implementing.

2 FRONT END

The state of web application development has moved past serving static html pages, style sheets, and JavaScript. With the demand for dynamic content and more user features, there has been significant advancement in front-end frameworks that can be developed in a more readable language and allows for the creation of dynamic web applications. These frameworks use a process called transpilation to transform the source code into a bundled application that can be executed by all modern browsers. This is one of the criteria for a Progressive Web Application which would be a benefit for Ninkasi because they plan to use this application cross platform with possibly unreliable internet connection. From a project perspective, the use of these frameworks has the benefit of quicker development, superior code modularity, and testability. The existing front-end application is written in Vue.js, one of the frameworks we review for its practical applicability among other potential candidates.

2.1 React

React is a front-end framework developed by Facebook and one of the most popular libraries used in industry today. It uses a quicker virtual DOM instead of the traditional DOM which increases performance and can be rendered on clients and on servers. It encourages the developer to utilize reusable UI components that are dynamic. Data flows unidirectionally through these components, which makes the application easier to reason about and avoids unintended side effects during runtime.

You can write your front-end logic in JavaScript and use a templating language called JSX to build the app views. There is a slight learning curve to the React style of programming, but many engineers clearly prefer this technology as it is the top-ranking technology in terms of engineers that would use it and use it again [1].

2.2 Vue

Vue is a similar technology to React in that it is primarily concerned with the front-end aspect of the application. It was developed and is maintained by an international community of software engineers. Like React it also utilizes a virtual DOM, promotes the use of composable reusable components and allows for other libraries to be integrated with ease. One main difference is that in Vue, you do not have to use the JSX templating language and can simply use html. JSX is an option with Vue as well, so in some ways that is a major advantage.

There are no major performance advantages between the two however, so the real decision come down to the preference of the coding syntax. As huge benefit to Vue is the concept of single file components. These components contain the template, logic and style for a section of the application [2]. This simple organization is a huge advantage and is no doubt one of the reasons why the previous group that worked on this project chose it as their front-end framework. Although it is not as widely used, many engineers in the industry are excited about Vue and among those who have not used it, they have expressed interest in learning it [1].

2.3 Preact

Yet another similar framework to the two previously described is Preact; a lighter library (just over 5kb) and in some ways more simplified version of React. The programming paradigm used is the same as React so experienced developers would find this an easy transition. As a smaller library, it excludes pieces of React that are not commonly used and simplifies other aspects. The benefit for this project is that we could guarantee that our app was the smallest size possible and that any additional functionality that Preact needs for is completely modular and attachable [3].

A great benefit of Preact, although not in our use case, is that it can be substituted for an existing React app very easily, reducing the application size while performance stays the same. There are many large companies and open source organizations that use this technology, so it is surely vetted and reliable.

3 STYLE

Style governs how the content is displayed to the user and is arguably the most important part of user experience. This area should be considered with special care because it must meet the needs of the user and be intuitive. There have already been some issues raised around how things are being displayed on the current application and those issues are more pressing because they are the most visible.

3.1 CSS

Cascading Style sheets, or CSS, is natively understood by modern web browsers and was first developed in 1994. This technology governs how html elements are laid out and styled on the display. No matter which technology we use, it will all end up turning into CSS in the end [4].

One major benefit to this technology is how well is integrates with Vue. Vue can take the CSS written in single file components and apply those style scoped to those components only. With a cursory understanding of CSS, the task of styling an application becomes much easier. This ability is not limited to Vue as React has this ability as well, but it is not built into the library by default. Special care will need to be given to ensure browser and mobile compatibility but that is no different than any other modern web application.

3.2 Less

Less stands for Leaner Style Sheets and is a preprocessor language that is compiled into CSS. A big advantage of Less is the ability the create variables that are reusable and the ability to scope style definitions in a much easier way than CSS provides with its selectors. Sheets can be imported so that style definitions may be further reused [5].

This technology would be more of a benefit to use if there was not an already established set of CSS stylesheets for this app that simply need to be refactored. We are not ruling out the possibility of using this technology, because it may be easier to implement the changes we need if we include this technology with our implementation.

3.3 SCSS

SCSS stands simply for Sassy CSS and is a superset of the existing CSS language which means that every CSS stylesheet is already compatible with it. The syntax is terser than that of CSS and it uses indentation to indicate code blocks [6]. All the same features the Less has are present here as well and the benefit of refactoring a potentially complex set of stylesheets are present with this preprocessor.

4 BUNDLING

The end goal is to have an application that will run on any browser or mobile device and perform quickly. To do that we will have to make use of module bundlers or transpilers, because many of the newest technology features are not natively supported by browsers yet. They will take our code and transform it into a minified version of a languages that the browsers can understand. Gone are the days where you had to include script tags in order of dependencies, which had to make multiple requests back to the server. Now that process is simplified and much easier for browsers to load. There are many applicable technologies and they all have varying degrees of configuration overhead.

4.1 Webpack

Webpack is the most commonly used open-source module bundlers and is shipped as a dependency with many of the application bootstrapper programs for front-end frameworks. This means that often no configuration is necessary, and the application just runs. Other times, specific actions are needed during the build process or a certain library needs to be used that is not compatible. It is during these times that the technical overhead of webpack configuration becomes daunting.

Of course, with a widely used framework, documentation, examples, and boilerplate code are widely available online and depending on the complexity of the app, webpack simply could be left alone to do its job. The more time that can be spent on feature development and not bogged down by configuration, the better.

4.2 Parcel

Parcel aims to avoid the complex configuration known to webpack. Its syntax is much simpler, and it has support for all the other technologies mentioned in this review. Parcel takes an entry point file of type html or JavaScript and analyzes all the dependencies that are imported. It then does the same for each dependency until it has constructed an asset tree. The assets are then placed into a bundle tree which is optimized by joining common file dependencies. Finally, Parcel writes the bundled content to minified files with a packager specific to the file type [7].

Parcel takes care of the entire transpilation on its own, while webpack requires the proper configuration to do so. That alone, is a huge advantage in using parcel. We could count on a reduction in development time because everything works out of the box.

4.3 Babel

Babel is an open source transpiler for JavaScript. Webpack can use babel as a plugin to do its JavaScript transpilation. With this approach we lose the automated style pre-processing but that can be easily added as a separate build pipeline. The configuration for babel is very small because it only concerns itself with one language.

The singular use of babel would increase the configuration overhead in this project but since the app already uses babel as a plugin for webpack, we wont have to worry about piecing together a build process for the project. Babel is much better served in our project as a webpack plugin.

5 Conclusion

It's fortunate that we are working with a project that has been implemented already because it makes our choice of technologies somewhat easier. Our selection is more limited, but we agree with most of the technologies chosen by the previous team. There are a few changes that we would like to make, although the front-end technologies will likely not be replaced. We're favoring developing new features within the technologies we agree with rather than spend time radically changing the front-end.

[1] "State of javascript survey results: Front-end frameworks." [Online]. Available: https://2017.stateofjs.com/2017/front-end/results
[2] "Comparison with other frameworks - vue.js." [Online]. Available: https://vuejs.org/v2/guide/comparison.html
[3] "Home." [Online]. Available: https://preactjs.com/guide/differences-to-react
[4] [Online]. Available: https://www.w3.org/Style/CSS20/history.html
[5] T. C. L. Team, "Overview." [Online]. Available: http://lesscss.org/features/
[6] "Sass basics." [Online]. Available: https://sass-lang.com/guide
[7] [Online]. Available: https://parceljs.org/getting_started.html
[8] tutorialspoint.com, "Reactjs overview." [Online]. Available: https://www.tutorialspoint.com/reactjs/reactjs_overview.htm