Can Webpack still be considered the golden standard of module bundlers?

Research proposal bachelor's thesis 2021-2022

Maxim Vansteenkiste[1]

Samenvatting
The simple times of creating a website with plain HTML, CSS and JS are long gone. Nowadays we want to use a UI library like React or a CSS pre-compiler like Sass. All great technologies that make the life of a web developer easier and allow them to create even more awesome apps. But all these dependencies and different filetypes need to be bundled for a browser to understand them. That's exactly what a module bundler does. There are many module bundlers but by far the most popular is Webpack. Why is this the case? How does it compare to competitors? Should it still be considered as the golden standard? All of these questions will be answered in this paper where I take a deep-dive into the world of module bundlers.

Keywords
Module bundler — Web development — Javascript

Co-promotor
Cedric Vanhaeverbeke[2] (CodeFever)

Contact: [1] maxim.vansteenkiste@student.hogent.be; [2] cedric@codefever.be;

Inhoudsopgave

1. Introduction

A module bundler is one of the most essential technologies in modern web development. Nobody makes webapps with plain HTML, CSS and JS anymore. All the fancy new tools for a webdeveloper to use, like Angular or React, won't work in the browser without being bundled. So the choice of which bundler to use is very important. While Webpack is the most popular there are many others that claim to do the job better. Why is it still the king then? How does Webpack compare to its competitors? Are their claims valid? If so, why is Webpack still the most popular?

Many developers give much thought about which framework to use, which back-end and so forth. The module bundler however often doesn't get that much tought. This is because it's less attractive and in many frameworks comes pre-installed. But as your webapp can't run without it, it's one of the most important choices you make. In this paper, I want to demystify the module bundler and compare the most popular options. Not just in terms of speed and module size, but also its plugins, developer experience, ...

2. State-of-the-art

Much can already be found about module bundlers online. On the website or GitHub repository of the bundler itself, articles written by third-parties and video's on YouTube. While they contain very useful information, I have yet to find an in-depth analysis of the differences between the major players and what that means for the developer and the output product. This is what I want to achieve with this paper.

3. Methodology

First I will compare the most popular bundlers theoretically. How they work and how they may differ. To compare them in practice will require code-bases of many sizes. Luckily there are many open-source code bases on GitHub to chose from. Why many sizes, you ask? Because it could be interesting to see how module bundlers deal with smaller projects and larger ones. To see if the advantages of one diminishes when the project size in- or decreases. It could also be interesting to compare them based on the framework, like Angular or a Create-React-App app. Those frameworks ship with pre-configured bundlers so it remains to be seen if it's even possible to replace those with others. It's something I want to test anyhow.

I will test each codebase with many bundlers and then observe the differences in numerous categories. One important of which is the developer experience: how easy is it to install? How simple or complex is the config file and the plugins. There are a lot of things to consider.

4. Anticipated results

Many module bundlers claim speedier builds and smaller bundle sizes. So that's an easy prediction to make. Wether that remains true with projects of any size remains to be seen. Will those improvements be great enough to warrant the competitor to be used over Webpack. Developer experience and performance in development mode are defining factors as well. Obviously the former will be quite subjective so that's an important factor to note.

5. Anticipated conclusions

I honestly have no idea what I will conclude. That's the most important reason why I want to conduct this research. Webpack has served us well but as the tech world shifts more and more towards applications written with web technologies, the time of newer approaches may have come.