



А.Д. Глазов А.Д. Румбенс
Пензенский государственный технологический университет

СРАВНИТЕЛЬНЫЙ АНАЛИЗ СБОРЩИКОВ МОДУЛЕЙ ДЛЯ JAVASCRIPT-ПРИЛОЖЕНИЙ

В статье представлен обзор современных сборщиков модулей для веб-приложений, написанных на JavaScript. Выделены основные достоинства и недостатки каждого из них. Проведен анализ, а также сравнение сборщиков по актуальным критериям разработки.

Сборщик, модуль, Webpack, Parcel, Vite, конфигурация.

JavaScript – один из самых популярных языков программирования в настоящее время. Большинство клиентских приложений, и не только, пишутся на нем. Если вы разрабатываете приложения в текущей экосистеме JavaScript, то вы наверняка знаете, что это практически невозможно без использования одного из многих инструментов сборки JavaScript. В современных реалиях у вас может быть несколько сборщиков под каждое приложение, в зависимости от типа и предназначения проекта. Важно понимать, какие существуют инструменты, и в чем различие между ними. В этой статье будет представлен обзор самых популярных инструментов сборки JavaScript, будут выделены основные плюсы и минусы каждого из них, а так же различия между ними.

Для начала стоит разобраться, что вообще представляет из себя инструмент сборки. Инструмент сборки JavaScript – инструмент, который позволяет автоматизировать, путем написания различного рода скриптов, выполнение широкого спектра задач в отношении к разрабатываемому ПО. Такие инструменты автоматически выполняют задачи, которые в противном случае мы выполняли бы вручную. Как правило, выделяют следующие задачи:

- облегчение, сжатие файлов;
- оптимизация изображений и шрифтов;
- объединение (конкатенация) файлов;
- компиляция или транспиляция кода;
- генерация сервера разработки;
- перезагрузка модулей в режиме реального времени без полного обновления страницы;
- просмотр файлов и автоматическое создание сборок при обнаружении изменений;
- управление и обновление зависимостей;
- автоматизированный анализ и поиск ошибок в коде;
- деплоинг кода (процесс развертывания и запуск веб-приложения или сайта на сервере или хостинге);
- запуск модульных тестов;
- автоматическая генерация документации;

– полифиллинг (процесс добавления недостающих методов, свойств или API к старым браузерам путем предоставления собственной версии недостающего родного кода).

Видно, что большинство функций одинаковы для каждого сборщика, однако между ними, по-прежнему, можно выделить различия. У каждого инструмента реализация этих методов выполняется по-разному, а также существует набор своего уникального функционала.

В этой статье будут рассмотрены 3 самых популярных на данный момент сборщика модулей с точки зрения практического применения и удобства пользования.

Webpack

Пожалуй, это самый известный инструмент сборки. В процессе работы Webpack анализирует модули приложения, создает граф зависимостей, затем собирает модули в правильном порядке в один или несколько бандлов (так называемый узел или пучок), на который впоследствии может ссылаться файл index.html (рис. 1).

```
App.js -> |  
Dashboard.js -> | Bundler | -> bundle.js  
About.js -> |
```

Рис. 1. Сборка модулей в Webpack

Parcel

Parcel – это еще один популярный инструмент сборки, написанный на языке Rust. Уже спустя несколько дней после своего релиза он собрал большое количество положительных отзывов и много звездочек на github. Данный сборщик позволяет быстро собирать маленькие проекты без длительной настройки.

Vite

Vite был создан не так давно, среди прочих сборщиков он является самым молодым. Его разработчик –

Эван Ю, который до этого создал Vue.js – популярный фреймворк для JavaScript. Именно под этот фреймворк и был разработан Vite, встроенная поддержка позволяет использовать его в большинстве фронтенд-проектов, написанных на других языках. В своей основе данный сборщик использует Rollup и esbuild.

Преимущества и недостатки

Каждый из этих инструментов имеет как свои преимущества, так и недостатки. Если говорить про Webpack, то можно выделить следующие достоинства:

- расширенное разделение кода. Webpack известен тем, что позволяет загружать определенные скрипты по необходимости, а не целый бандл на каждой странице;
- плагины и загрузчики. Webpack включает в себя большую экосистему плагинов и загрузчиков, что способствует значительному расширению его функциональности;
- возможности настройки. Вы можете использовать данный сборщик вообще без настройки, но получить максимальную мощность от него вы сможете, если будете настраивать его под каждый проект отдельно;
- поддержка множества различных форматов вывода.

Из общих плюсов также можно выделить широкую распространенность и популярность, что делает процесс поиска различной информации и документации простым.

Недостатки следующие:

- сложность освоения. Webpack – это объемный и многофункциональный инструмент, что делает его изучение весьма длительным и тяжелым;
- актуальность информации. Часть документации устарела, так как обновления привносят большое количество изменений;
- процесс разработки. Из-за объемности и сложности настройки конфигурация Webpack становится слишком нагроможденной, что приводит к замедлению процесса разработки.

Выделим достоинства Parcel:

- нулевая конфигурация. Для работы с Parcel вам вообще не нужно ничего настраивать, не нужен опыт работы с инструментами сборки, чтобы начать работу;
- диагностика ошибок. В случае ошибки, в терминале и в браузере будет отображаться удобная и интуитивно понятная диагностика;
- скорость сборки. Среди всех инструментов сборки, Parcel является одним из быстрых;
- поддержка различных типов файлов. Вам не нужно настраивать Parcel для работы с определенным типом файлов, функции настройки поддерживают практически любой фронтенд-проект;
- встроенная поддержка HMR (Hot Module Replacement). Такое нововведение позволяет отображать изменения в кратчайшие сроки.

К недостаткам же можно отнести следующие моменты:

- работа с кэшем. Плохо оптимизирована, особенно заметно при работе с моно-репозиторием;
- задержка HMR. Потребуется много времени, чтобы отразить изменения в браузере, что может быть связано с кэшем. Браузер может попасть в состояние загрузки, из которого уже никогда не выйдет. В таком случае остается только перезапустить скрипт сборки или закрыть вкладку.

Рассмотрим Vite. Из достоинств данного инструмента можно однозначно выделить следующее:

- время сборки. Оно достаточно быстрое, особенно если это локальная сборка;
- Rollup и esbuild. Как уже упоминалось выше, в своей основе Vite использует esbuild для предварительной сборки и Rollup для производственной сборки;
- улучшение работы с медленными серверами. Vite сокращает время обслуживания приложения за счет предварительной сборки (с использованием esbuild) и обслуживания кода собственными модулями ES (ecmascript);
- HMR. Vite может очень быстро заменять модули даже для самых больших и тяжелых приложений.

К недостаткам можно отнести следующие пункты:

- плохая поддержка typescript. Typescript – это, можно сказать, уже золотой стандарт, мало приложений обходятся без него;
- конфигурация. Ее нужно настраивать, хотя бы на минимальном уровне;
- переменные среды. Работа с ними в данном сборщике отличается от общепризнанных стандартов, что может повлечь некоторые неудобства. Например: обращение переменных `.env` происходит по другому пути.

Теперь, когда были рассмотрены плюсы и минусы каждого, можно приступать к непосредственному сравнению сборщиков.

Анализ инструментов сборки

Для начала нужно выделить критерии сравнения. Сборщик можно назвать эффективным, если соблюдены следующие условия:

- быстрота сборки;
- удобство пользования;
- многофункциональность.

По этим критериям и будем сравнивать.

Webpack в вопросе скорости сборки является рекордсменом с конца. По мере того как приложение будет развиваться и расти, будет расти и время сборки, вплоть до нескольких минут. Большинство современных сборщиков обгоняют его в десятки, а то и в сотни раз. Конечно, можно ускорить время сборки различными плагинами, но это лишняя нагрузка, к тому же это все равно не сделает его конкурентоспособным. У Parcel и Vite ситуация намного лучше, они собирают проект гораздо быстрее, особенно Parcel.

При этом ситуация усложняется тем, что файлы конфигурации имеют различный «вес». Возьмем, к примеру, следующий скрипт:

```
const path = require('path');

module.exports = {
  entry: './src/script.js',
  output: {
    filename: 'main.js',
    path: path.resolve(__dirname, 'dist'),
  },
};
```

Рис. 2. Простейший скрипт на Webpack

Простейший пример, где нет ничего, кроме указания – взять один файл как входную точку и собрать из него бандл (узел, конечный файл конфигурации, создаваемый специальным методом bundler).

Сборка файла занимает 992 мс, выходной файл конфигурации весит 71 кб. Если такую же сборку запустить на Vite, то сборка займет всего 15 мс, а вес файла конфигурации составит всего лишь 86 байт. У Parcel значения схожи с Vite, даже немного лучше.

По удобству пользования фаворитом будет Parcel. Вы достаточно просто можете начать работать, даже если до этого вообще не работали со сборщиками. Сам по себе инструмент достаточно прост в использовании и интуитивно понятен. При работе с Vite уже нужны будут некоторые знания и опыт разработки. Webpack же является далеко не самым удобным в плане использования, как уже упоминалось выше, он требует гораздо больших усилий, чтобы начать писать на нем хотя бы небольшой конфиг.

Однако Webpack до сих пор является неоспоримым лидером в плане функциональности и разнообра-

зия. Огромное количество библиотек и плагинов позволяют настроить самую разнообразную конфигурацию под самые разнообразные проекты. Parcel таким разнообразием похвастаться не может, этот инструмент не обладает такой же популярностью в среде разработчиков, как Webpack, следовательно, и различных дополнений будет мало. В своей основе данный сборщик предназначен в первую очередь для небольших приложений. Если попробовать написать что-то покрупнее, то Parcel – явно не лучший выбор. Vite находится посередине. Он мощнее Parcel, и явно эффективнее в работе с большими приложениями, однако он не обладает настолько большим разнообразием, как Webpack.

Подведем итоги. Webpack – это до сих пор монолитная структура, имеющая большой функционал и разнообразие. Несмотря на некоторую устарелость, много разработчиков не спешат менять его. Parcel же во многом уступает как Webpack, так и Vite. В среде разработчиков его часто называют «Webpack на минималках». Это обусловлено тем, что многие функции Webpack в том или ином виде есть и у Parcel, и он идеально подходит для написания небольших приложений. Для таких приложений он подходит гораздо лучше, чем Webpack. Vite, несмотря на свой недавний релиз, составляет хорошую конкуренцию Webpack. Много разработчиков уже перешли на Vite. И, хотя он не так хорошо проработан в некоторых аспектах, виден большой потенциал. Данный сборщик хорошо поддерживается и до сих пор развивается. Основными достоинствами на фоне того же Webpack, являются легковесность и быстрота. Исходя из этого, можно сделать вывод, что Vite на данный момент – один из лучших и самых перспективных сборщиков для современных проектов.

A.D. Glazov, A.D. Rumbens
Penza State Technological University

COMPARATIVE ANALYSIS OF MODULE BUILDERS FOR JAVASCRIPT APPLICATIONS

The article provides an overview of modern module builders for web applications written in JavaScript. The main advantages and disadvantages of each of them are highlighted. The analysis, as well as a comparison of assemblers according to current development criteria, was carried out.

Builder, module, Webpack, Parcel, Vite, configuration.