Инженерный вестник Дона, №12 (2023)
ivdon.ru/ru/magazine/archive/n12y2023/8889

Исследование эффективности Tree-Shaking в современных инструментах сборки веб-приложений

А.С. Лабоскин

Acquire, San Francisco, USA

Аннотация: В данной работе проводится анализ эффективности механизма Tree-Shaking, который является ключевым способом оптимизации размера клиентских веб-приложений. Сравнивается его реализация в пяти популярных инструментах для сборки проектов: Webpack, Rollup, Parcel, Vite и Esbuild. Результаты тестирования демонстрируют различия в их поведении и общей эффективности при удалении избыточного кода, подчеркивая актуальность применения Tree-Shaking в веб-разработке.
Ключевые слова: tree-shaking, javascript, front end, веб-приложения, оптимизация, скорость загрузки.

Введение

В современной веб-разработке, где скорость загрузки и производительность приложений играют важную роль в удержании пользователей [1], оптимизация размера загружаемых ресурсов является одной из ключевых задач для разработчиков. Эта проблема становится особенно актуальной в связи с ростом сложности и размера клиентских веб-приложений, которые часто включают большое количество внешних библиотек и фреймворков [2]. Большой объем кода ухудшает интерактивность и увеличивает нагрузку на клиентские устройства, что имеет критическое значение для мобильных устройств, составляющих 64.1% мирового веб-трафика [3]. Tree-Shaking представляет собой современный подход к оптимизации, позволяющий удалять избыточный код из конечных сборок JavaScript-приложений, улучшая их производительность и скорость загрузки.

Tree-Shaking

Tree-Shaking — процесс удаления неиспользуемого кода при сборке клиентских веб-приложений, который позволяет сократить размер финальной сборки (бандла) приложения. Принцип его работы основывается

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

на статическом анализе кода, который подразумевает комплексное исследование структуры кода без его выполнения [4]. На этапе сборки проекта анализатор строит граф зависимостей и определяет неиспользуемые модули. После этого из финальной сборки приложения удаляются неиспользуемые участки кода. Пример работы Tree-Shaking представлен на рис. 1.

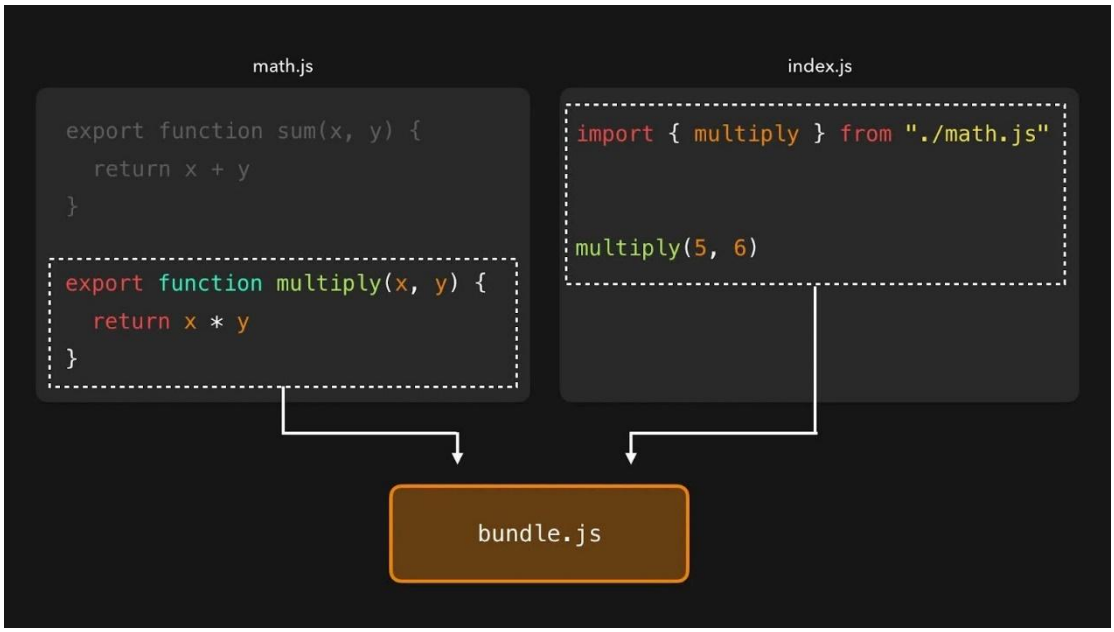

Рис. 1. – Пример работы Tree-Shaking

Важным аспектом эффективности Tree-Shaking является модульность кода. Современные принципы веб-разработки подразумевают разделение всего кода проекта на мелкие независимые модули, что позволяет более точно идентифицировать и исключать неиспользуемые компоненты. При этом для правильного определения избыточного кода необходимо использование модулей ECMAScript с ключевыми словами import и export [5]. Более ранние форматы модулей в JavaScript, включая CommonJS, не поддерживают статический анализ зависимостей на том же уровне [6].

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

Стоит отметить, что не все сценарии удаления кода являются тривиальными, в частности когда речь идет о побочных эффектах (side-effects) — действиях программы, изменяющих среду ее выполнения [7]. Удаление кода, результат работы которого не используется напрямую, но изменяет глобальное состояние или взаимодействует с внешними системами, может привести к нежелательным изменениям в поведении приложения. Данная проблема, как правило, решается при помощи специальных аннотаций, предотвращающий удаление фрагмента с побочными эффектами в процессе Tree-Shaking [8].

Эффективность Tree-Shaking может значительно варьироваться в зависимости от используемого инструмента для сборки, каждый из которых использует уникальные алгоритмы для определения неиспользуемого кода. Разница может заключаться как в обработке отдельных участков кода, так и в скорости работы и конечном размере сборки. Поэтому правильный выбор инструмента для сборки является ключевым фактором успешной оптимизации веб-приложений.

Методика тестирования

В рамках данного исследования было проведено сравнение эффективности механизма Tree-Shaking в 5 популярных инструментах для сборки клиентских веб-приложений: Webpack (версия 5.88.2), Rollup (версия 3.28.1), Parcel (версия 2.9.3), Vite (версия 4.4.9) и Esbuild (версия 0.19.2) [9]. В качестве оборудования для тестирования был использован ноутбук Apple Macbook Pro 16 с процессором Apple M1 Pro и 16 Гб ОЗУ, работающий на операционной системе macOS 14.0 Sonoma.

Для получения объективных данных о поведении выбранных инструментов при удалении кода были определены и имплементированы 5 экспериментов, отражающих различные сценарии присутствия неиспользуемых фрагментов в исходном коде. Описание и исходный код

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

экспериментов представлены в таблице №1. Критерием успешного выполнения в экспериментах 1, 2, 4 и 5 считалось отсутствие кода функции bar в конечной сборке проекта, в эксперименте 3 — ее наличие.

Таблица № 1

Сценарии тестирования

№	Описание	Исходный код

1	Неиспользуемый импорт	// foobar.js
		export const bar = () => console.log(‘bar’);
		// index.js
		import * as helpers from './helpers';

2	Частично используемый импорт	// foobar.js
		export const foo = () => console.log(‘foo’);
		export const bar = () => console.log(‘bar’);
		// index.js
		import { foo } from './helpers';
		foo();

3	Побочный эффект	// foobar.js
		export const foo = () => console.log(‘foo’);
		export const bar = () => console.log(‘bar’);
		bar();
		// index.js
		import { foo } from './helpers';
		foo();

4	Динамический импорт	// foobar.js
		export const foo = () => console.log(‘foo’);
		export const bar = () => console.log(‘bar’);
		// index.js
		import('./helpers').then(({ foo }) => foo());

5	Неиспользуемый метод объекта	// foobar.js
		export const helpers = {
		  foo: () => console.log(‘foo’),
		  bar: () => console.log(‘bar’)
		}
		// index.js
		import { helpers } from './helpers';
		helpers.foo();

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

Для оценки общей эффективности удаления избыточного кода в выбранных инструментах были проведены замеры времени сборки и размера конечных файлов до и после включения Tree-Shaking. В качестве тестового проекта был использован файл, в котором импортируются и используются функции из 10 популярных JavaScript библиотек. Код файла представлен на рис. 2. Включение Tree-Shaking в конфигурации сборщиков должно удалить неиспользуемый код внешних библиотек и значительно сократить конечный размер сборки.

```
import { merge } from 'lodash'
import { useField } from 'formik'
import { t } from 'i18next'
import { clone } from 'underscore'
import { createElement } from 'react'
import { useNavigate } from 'react-router'
import { createStore } from 'redux'
import { scan } from 'rxjs'
import { css } from 'styled-components'
import { render } from 'vue'

console.log(merge);
console.log(useField);
console.log(t);
console.log(clone);
console.log(createElement);
console.log(useNavigate);
console.log(createStore);
console.log(scan);
console.log(css);
console.log(render);
```

Рис. 2. – Код тестового файла

Результаты тестирования

Полученные данные демонстрируют одинаковое поведение всех инструментов в базовых сценариях присутствия неиспользуемого кода, представленных в экспериментах 1–3. При этом, в случае с динамическим импортом модуля в эксперименте 4, избыточный код был удален только в Rollup и Parcel. В то же время в эксперименте 5 только Webpack определил неиспользуемый метод импортируемого объекта и удалил его. Ни один из

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

инструментов не справился со всеми сценариями, однако можно судить о более эффективной реализации Tree-Shaking в Rollup, Parcel и Webpack.

Итоги тестирования представлены в таблице № 2 и таблице № 3.

Таблица № 2

Результаты тестирования поведения в различных сценариях

	№ эксперимента				
	1	2	3	4	5
Webpack	●	●	●	○	●
Rollup	●	●	●	●	○
Parcel	●	●	●	●	○
Vite	●	●	●	○	○
Esbuild	●	●	●	○	○

Таблица № 3

Результаты замеров времени сборки и размера файлов до и после включения Tree-Shaking

	Время сборки, с		Размер файлов, кб	
	До	После	До	После
Webpack	3.9	3.2	534	267
Rollup	3.8	2.95	368	310
Parcel	1.44	1.6	1855	318
Vite	-	1.35	-	305
Esbuild	0.2	0.21	419	278

Результаты замеров времени сборки и размера конечных файлов позволяют судить и том, что включение Tree-Shaking позволяет значительно сократить конечный размер приложения после сборки по всех инструментах. Наименьший размер файлов был продемонстрирован при использовании Webpack и Esbuild.

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

Скорость сборки приложения не является критическим параметром при выборе инструмента разработчиками. Однако, важно отметить, что дополнительный анализ и удаление избыточного кода при включении Tree-Shaking оказывают незначительное влияние на этот показатель. При этом в зависимости от конкретного инструмента длительность сборки может как увеличиться, так и уменьшиться. С высокой долей вероятности это обусловлено различиями в наборе и последовательности прочих преобразований, происходящих в процессе сборки приложения [10].

Заключение

Проведенное исследование демонстрирует важность применения Tree-Shaking для оптимизации размера клиентских веб-приложений. По результатам тестирования было установлено, что использование этого механизма позволяет значительно сократить размер конечного кода, улучшая время загрузки и снижая потребление ресурсов устройства. Полученные данные о различиях в реализации и общей эффективности Tree-Shaking в популярных инструментах сборки могут быть полезны разработчикам при выборе подходящих средств для сборки проектов.

Литература

1. Arapakis I., Bai X., Cambazoglu B.B. Impact of response latency on user behavior in web search // Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval, 2014. P. 103–112. URL: iarapakis.github.io/papers/SIGIR14.pdf

2. Лабоскин А.С. Исследование и сравнительный анализ производительности AssemblyScript // Инженерный вестник Дона, 2023, №11. URL: ivdon.ru/uploads/article/pdf/IVD_95__11_laboskin.pdf_f72a537da2.pdf

3. Mobile vs. Desktop Traffic Market Share [October 2023] | Similarweb. URL: similarweb.com/platforms

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

4. What is tree shaking in JavaScript - Bugpilot Technical Guides. URL: bugpilot.io/guides/en/tree-shaking-in-javascript-0461

5. Huang S. Load time optimization of JavaScript web applications, 2019. URL: diva-portal.org/smash/get/diva2:1318692/FULLTEXT01.pdf

6. How CommonJS is making your bundles larger | Articles | web.dev. URL: web.dev/articles/commonjs-larger-bundles?hl=en

7. Nicolay J, Stiévenart Q, De Meuter W, De Roover C. Purity analysis for JavaScript through abstract interpretation // Journal of Software: Evolution and Process, 2017. URL: soft.vub.ac.be/Publications/2017/vub-soft-tr-17-11.pdf

8. Tree Shaking | webpack. URL: webpack.js.org/guides/tree-shaking/#mark-the-file-as-side-effect-free

9. The Most Popular Build Tools for Front-end Developers (2023). URL: stackdiary.com/build-tools-for-web-development

10. What Does a Bundler Actually Do? – INNOQ. URL: innoq.com/en/articles/2021/12/what-does-a-bundler-actually-do

References

1. Arapakis I., Bai X., Cambazoglu B.B. Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval, 2014. P. 103–112. URL: iarapakis.github.io/papers/SIGIR14.pdf

2. Laboskin A.S. Inzhenernyj vestnik Dona, 2023, №11. URL: ivdon.ru/uploads/article/pdf/IVD_95__11_laboskin.pdf_f72a537da2.pdf

3. Mobile vs. Desktop Traffic Market Share [October 2023] | Similarweb. URL: similarweb.com/platforms

4. What is tree shaking in JavaScript - Bugpilot Technical Guides. URL: bugpilot.io/guides/en/tree-shaking-in-javascript-0461

5. Huang S. Load time optimization of JavaScript web applications, 2019. URL: diva-portal.org/smash/get/diva2:1318692/FULLTEXT01.pdf

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023

6. How CommonJS is making your bundles larger | Articles | web.dev. URL: web.dev/articles/commonjs-larger-bundles?hl=en

7. Nicolay J, Stiévenart Q, De Meuter W, De Roover C. Journal of Software: Evolution and Process, 2017. URL: soft.vub.ac.be/Publications/2017/vub-soft-tr-17-11.pdf

8. Tree Shaking | webpack. URL: webpack.js.org/guides/tree-shaking/#mark-the-file-as-side-effect-free

9. The Most Popular Build Tools for Front-end Developers (2023). URL: stackdiary.com/build-tools-for-web-development

10. What Does a Bundler Actually Do? – INNOQ. URL: innoq.com/en/articles/2021/12/what-does-a-bundler-actually-do

Дата поступления: 6.11.2023

Дата публикации: 15.12.2023

© Электронный научный журнал «Инженерный вестник Дона», 2007–2023