Optimizing Performance for a Better User Experience

Jad Joubran
WEB CONSULTANT
@joubranjad www.jadjoubran.io

Overview

Set a Performance Budget

Code Splitting

Dynamic Imports

Continuous Performance Auditing

Source Map Analyzer

Any website can be made into a PWA.

This doesn't mean your users are going to love it or even use it.

Performance Budget

It's a limit on the size of resources

- Size of JavaScript assets
- Size of CSS assets
- Maximum weight of images

It could also be a limit on metrics

- Page load time
- Time to Interactive

Benefits of a Performance Budget

Team-wide responsibility (technical & non-technical)

Strive to remain under the limit

Prioritize performance optimization based on budget

Performance Budget in Webpack

```
const performanceConfig = {
  hints: "warning",
  maxEntrypointSize: 50000,
  maxAssetSize: 100000
};
```

Enable for Production Mode Only

```
const mode = env.mode ? env.mode : "production";

return {
  input: {/*...*/},
  output: {/*...*/},
  plugins: [/*...*/],
  performance: mode === "production" ? performanceConfig : {}
};
```

Demo

Setting a Performance Budget in Webpack

Code Splitting

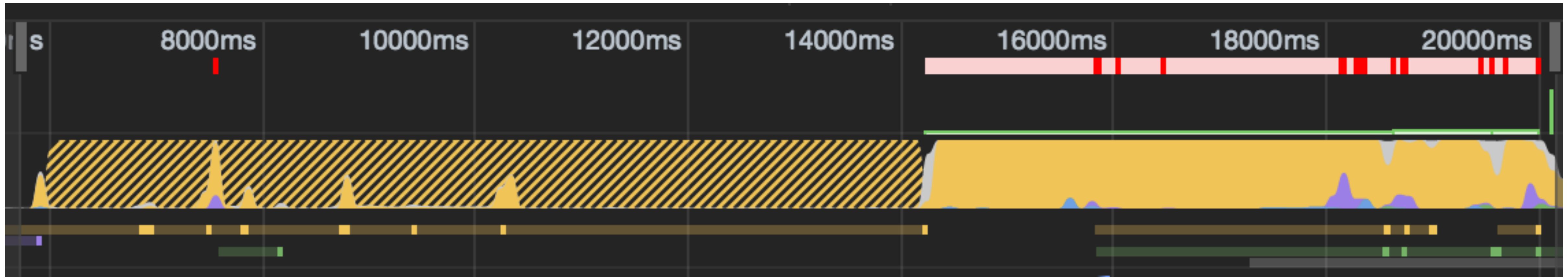
App Shell & Performance Budget

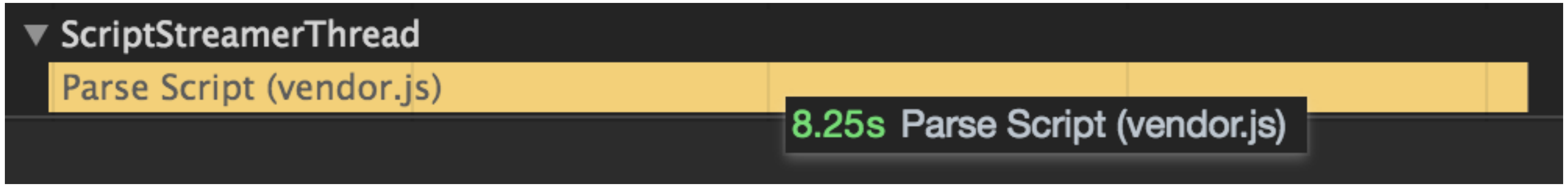
- Allow us to start fast

Use code splitting

- Start fast and remain fast

Motivation


s
8000ms
10000ms
12000ms
14000ms
16000ms
18000ms
20000ms


ScriptStreamerThread
Parse Script (vendor.js)
8.25s Parse Script (vendor.js)

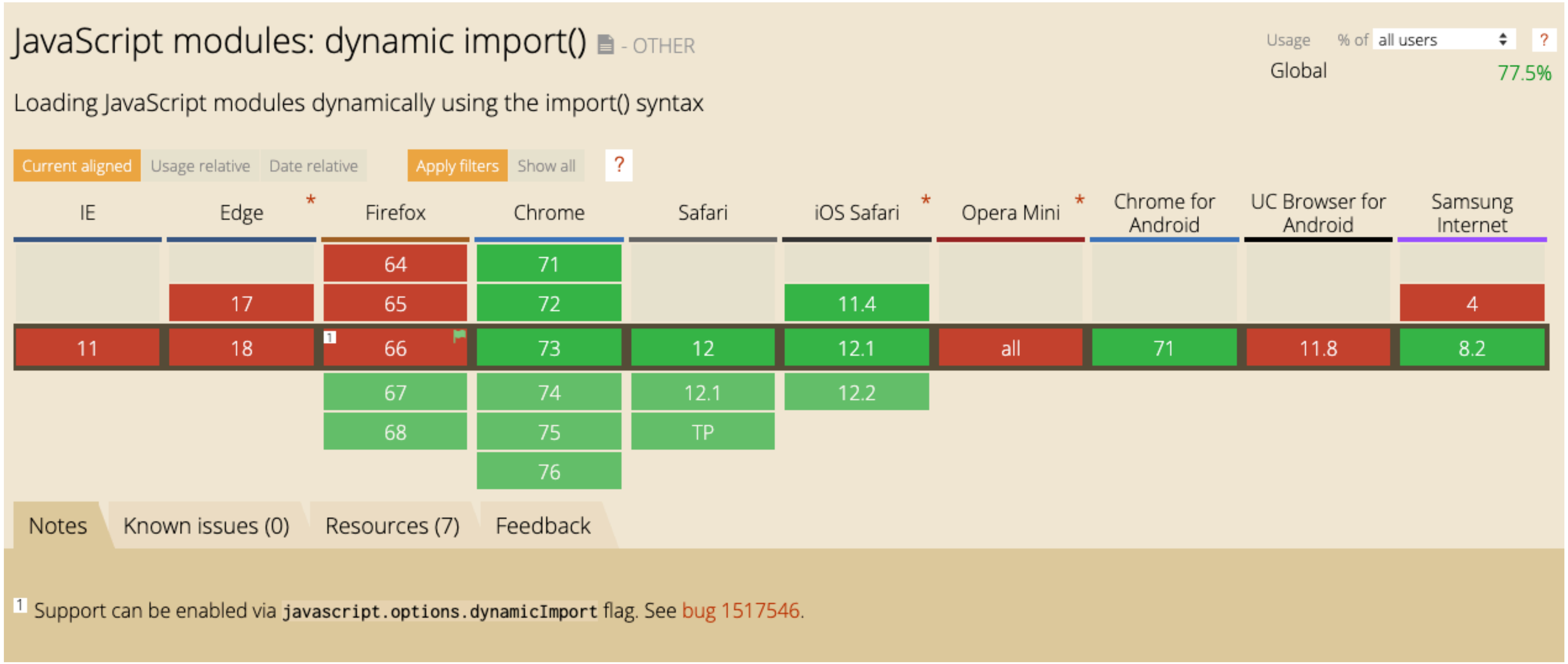
Benefits of Code Splitting

Start fast and remain fast

Add features without slowing down your web app

Conditionally load features that are used

Browser Support



JavaScript modules: dynamic import() - OTHER

Usage % of all users ?

Global 77.5%

Loading JavaScript modules dynamically using the import() syntax

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
		🟥 64	🟩 71						
	🟥 17	🟥 65	🟩 72		🟩 11.4				🟥 4
🟥 11	🟥 18	🟥 66 [1]	🟩 73	🟩 12	🟩 12.1	🟥 all	🟩 71	🟥 11.8	🟩 8.2
		🟩 67	🟩 74	🟩 12.1	🟩 12.2				
		🟩 68	🟩 75	🟩 TP					
			🟩 76						

[1] Support can be enabled via `javascript.options.dynamicImport` flag. See bug 1517546.

Dynamic Imports with Webpack

```
import("./help.js");

// => 0.84b15dc547719c822f71.bundle.js
```

Dynamic Imports with Webpack

```
import(/*webpackChunkName: "help"*/"./help.js");

// => help.84b15dc547719c822f71.bundle.js
```

Conditional Loading

```
together.addEventListener("click", () => {
    import("./src/logout.js");
});
```

Dynamic Imports for ES Modules

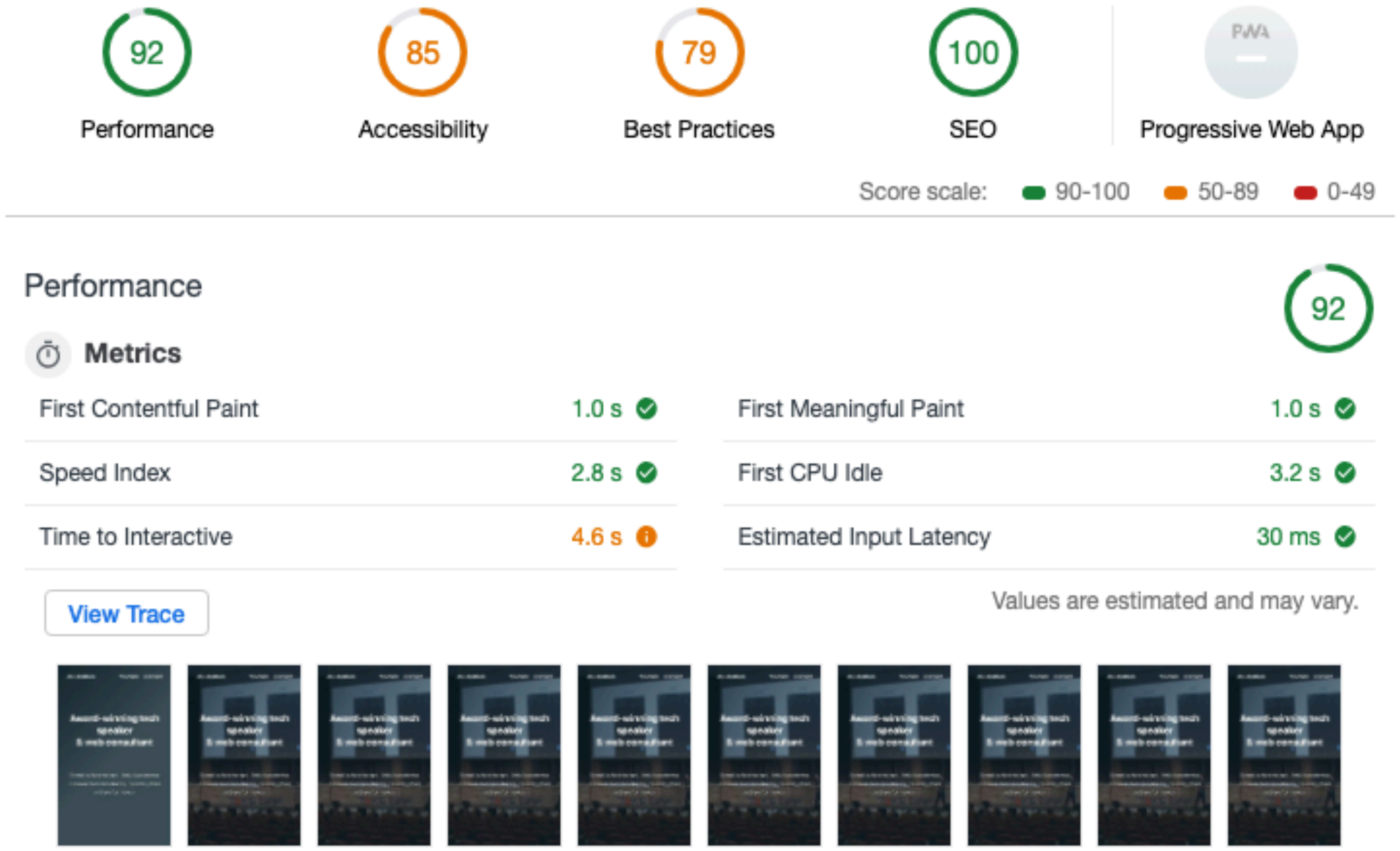
```
import("./src/modal.js").then(module => {
    module.default();
    module.otherFunction();
});
```

Continuous Performance Audits

Stay on top of your Performance Budget

Get notified as soon as things break

Lighthouse



92	85	79	100	PWA –
Performance	Accessibility	Best Practices	SEO	Progressive Web App

Score scale: 🟩 90-100 🟧 50-89 🟥 0-49

Performance

92

Metrics

First Contentful Paint	1.0 s ✓	First Meaningful Paint	1.0 s ✓
Speed Index	2.8 s ✓	First CPU Idle	3.2 s ✓
Time to Interactive	4.6 s ●	Estimated Input Latency	30 ms ✓

View Trace

Values are estimated and may vary.

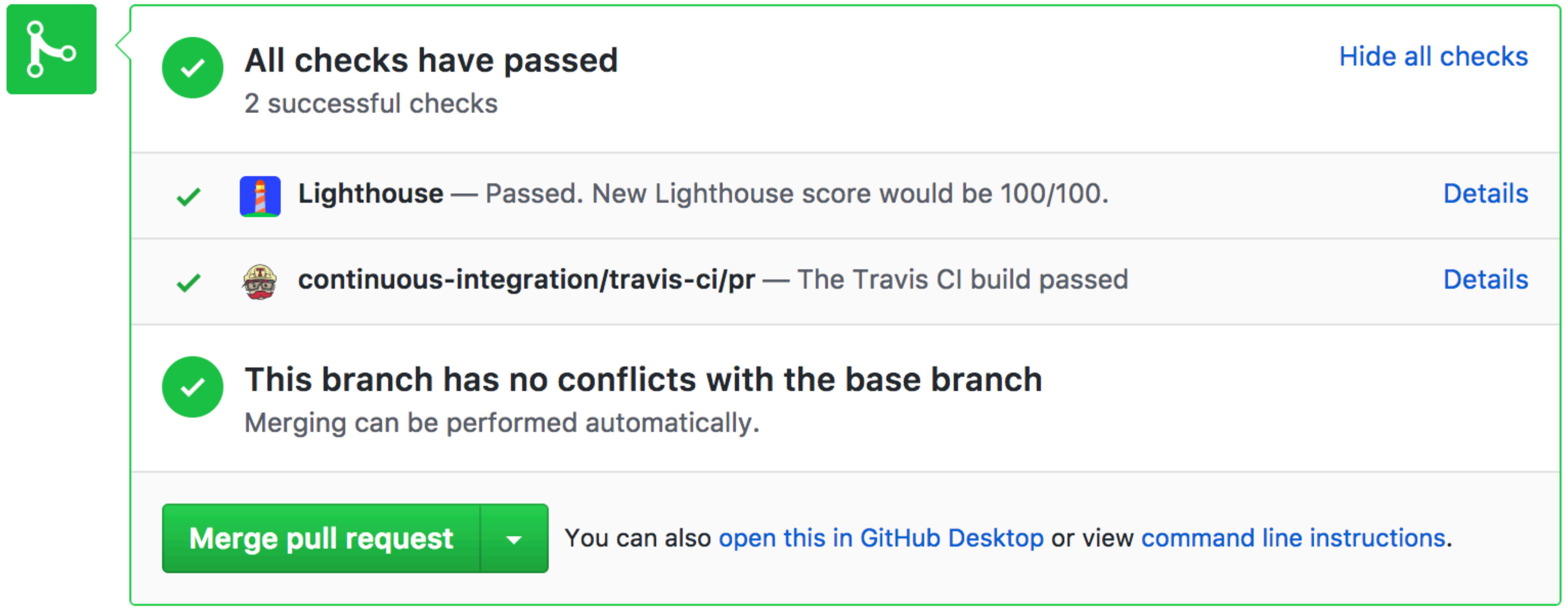
Continuous Integration

lighthouse (node module)

Github integration

- Lighthousebot

Lighthousebot



All checks have passed
2 successful checks

Hide all checks

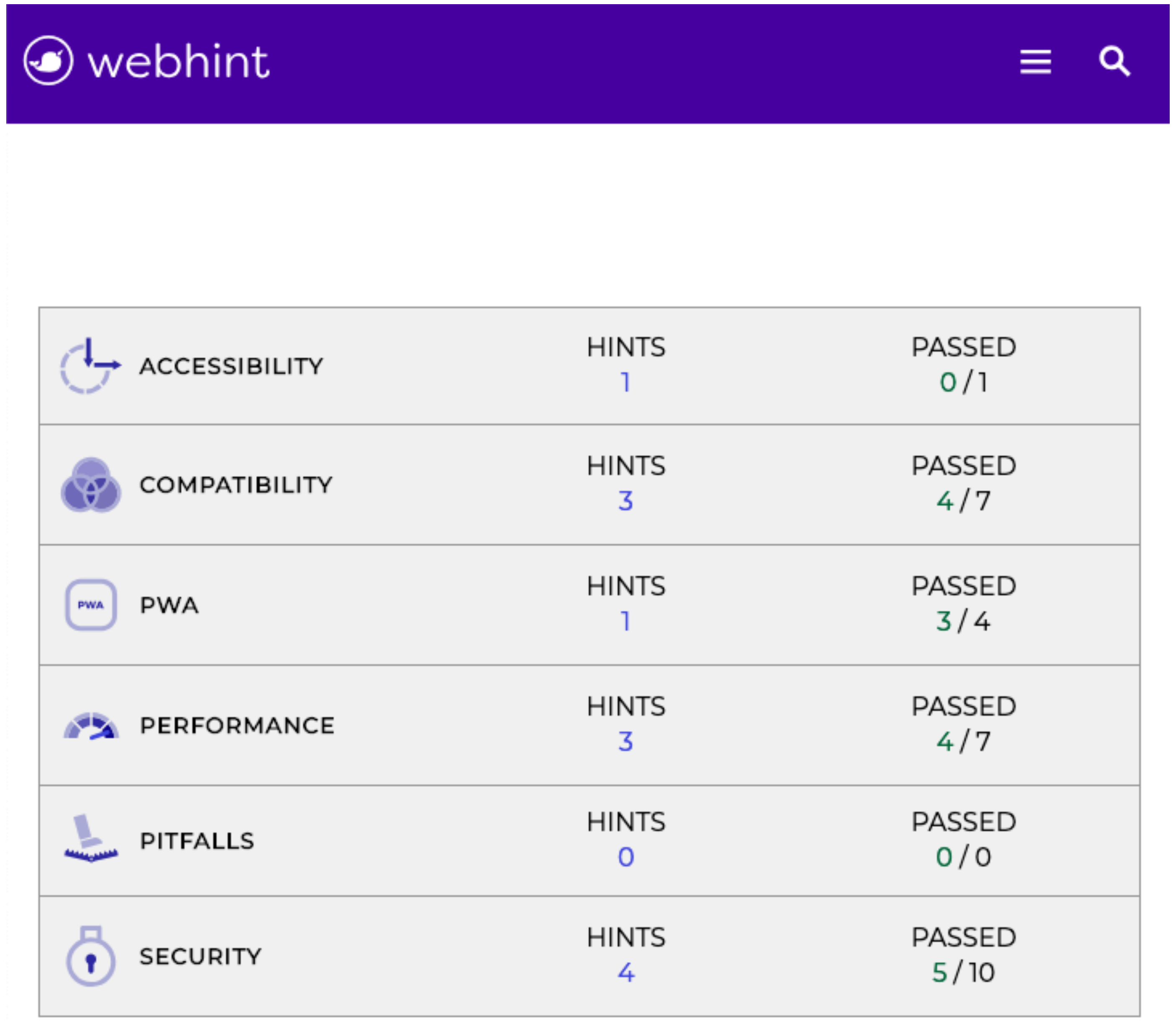
✓ **Lighthouse** — Passed. New Lighthouse score would be 100/100. Details

✓ **continuous-integration/travis-ci/pr** — The Travis CI build passed Details

This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

Webhint



ACCESSIBILITY	HINTS 1	PASSED 0 / 1
COMPATIBILITY	HINTS 3	PASSED 4 / 7
PWA	HINTS 1	PASSED 3 / 4
PERFORMANCE	HINTS 3	PASSED 4 / 7
PITFALLS	HINTS 0	PASSED 0 / 0
SECURITY	HINTS 4	PASSED 5 / 10

Benefits of Continuous Performance Audits

Automation

Blame PR

Blame Features

Integrate Performance with the Product

Demo

Performance Auditing

- Lighthouse

Source Map Analyzer

Analyze your JavaScript bundles

Understand & visualize what you're shipping

Source Map Analyzer

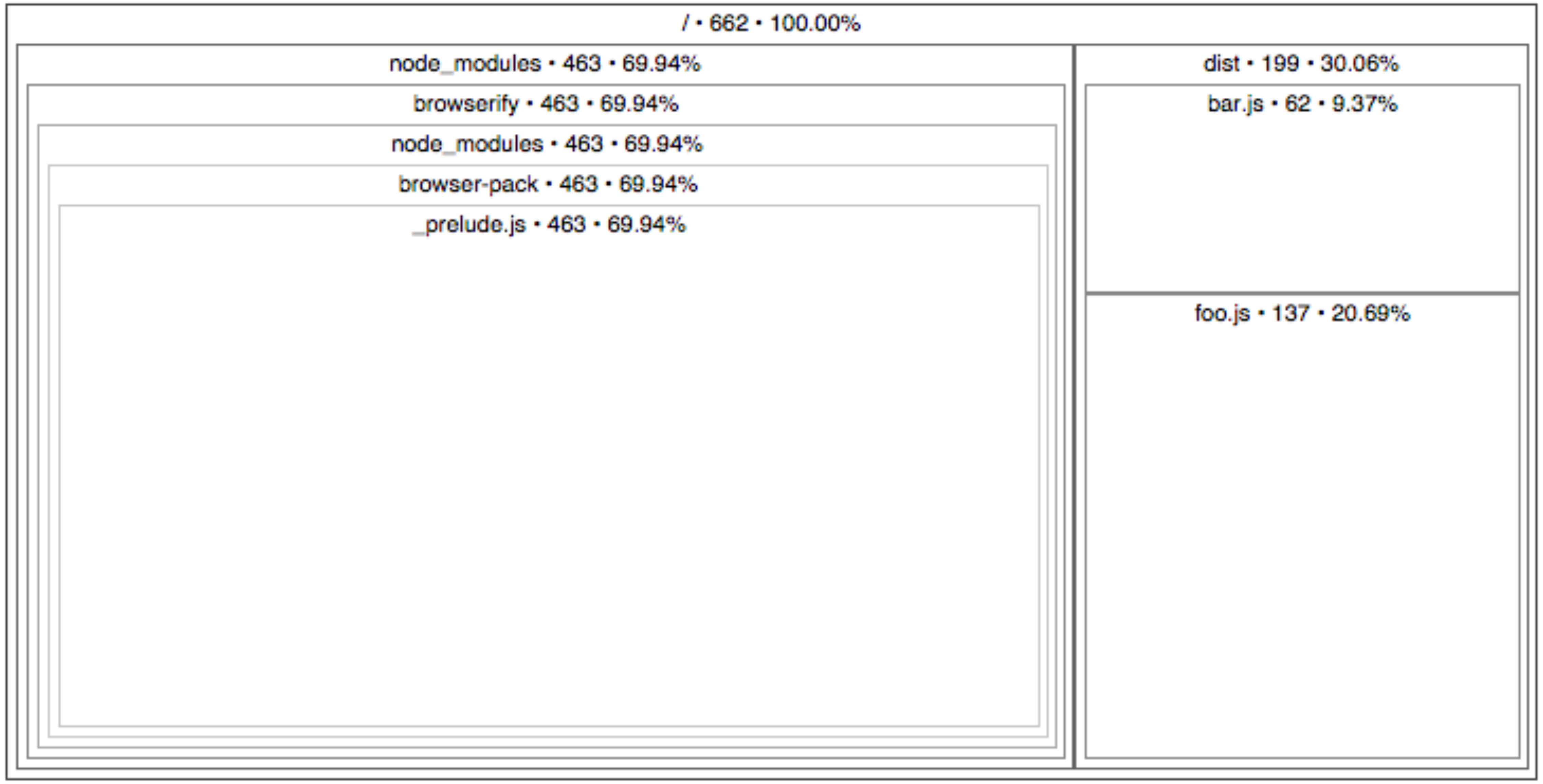
Detect anomalies

- Duplicate libraries
- Backend module included in frontend

Understand Performance bottlenecks

- Large library for minor functionality
- Large chunk for admin feature
- Large chunk that can be lazy loaded

Source Map Analyzer



Setup

```
npm install --save-dev webpack-bundle-analyzer
```

Webpack Setup

```
const BundleAnalyzer = require("webpack-bundle-analyzer");

return {
  plugins: [
     new BundleAnalyzer.BundleAnalyzerPlugin({
        analyzerMode: "static",
        openAnalyzer: false,
        reportFilename: "bundle-analyzer.html"
     })
  ]
}
```

Demo

Source Map Analyzer

Summary

Importance of Performance Budgets

Code Splitting using dynamic Imports

Continuous Performance Auditing

Source Map Analyzer