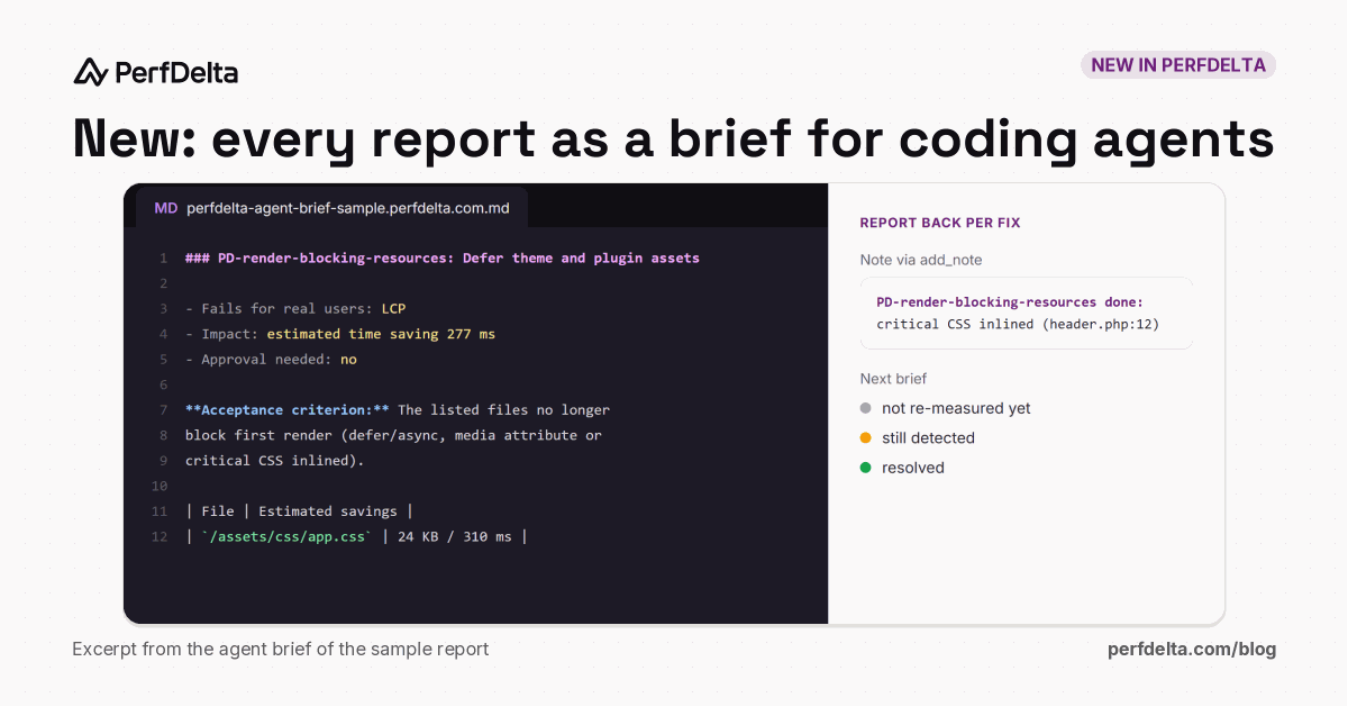
Website Speed Optimization with AI Coding Agents: Every Report as a Markdown Brief

Published November 5, 2026 by Christian Schreiber



The PerfDelta report lists the fixes that improve a website's speed the most, with the affected files, pages and elements. Until now, you had to carry them into the code yourself. Every report is also available as an agent brief, on the PerfDelta free plan for your own scans and from the Solo plan for every measured domain: a Markdown file that a coding agent such as Claude Code, Cursor, Codex or GitHub Copilot works through step by step in the repository of your website. The agent then reports back on every fix, and the next brief knows that status.

Website speed optimization with AI coding agents: the brief as a work order

A coding agent is only as good as its instructions. Tell it to "make the site faster" and you get guesswork. The agent brief gives it the same evidence the report shows, in a form it can read without asking back. The file has twelve sections:

1. **Task:** the goal is good Core Web Vitals for real users, the list is worked through from top to bottom, one fix per commit.
2. **Guardrails:** no visible change to design or content, no new plugins and no change to tracking or consent without your approval.

3. **A note on measured content** (more on that below).
4. **Baseline:** field data at the 75th percentile, lab data per page and device, the LCP element with its subparts, layout-shifting elements, page weight, third parties and script bundles.
5. **Environment:** the CMS, framework, CDN and other technology detected from the outside, explicitly marked "verify in the repository".
6. **Fixes** in the order of the report.
7. **Status from earlier notes**, so the agent does not touch anything twice.
8. **Changes since the previous scan:** what has been fixed since then and what changed in files or third parties.
9. **Performance budgets** of your project as acceptance limits.
10. **How to verify:** a Lighthouse command that the agent points at a local or staging address, and the re-scan in PerfDelta after deployment.
11. **Report-back format.**
12. **Limits of the data.**

The brief therefore covers more than the PageSpeed score and Core Web Vitals. Short cache lifetimes, missing text compression, a blocked back/forward cache or heavy third-party scripts are in it just like a slow LCP image. For an overview of all topics of speed optimization, see our guide to website speed optimization.

Every fix with a stable ID, an acceptance criterion and an approval flag

Every fix carries an ID such as `PD-render-blocking-resources` or `PD-third-party-blocking-youtube`. It is derived from the type of fix and the affected file or vendor. Build hashes in file names are left out, so the ID stays the same after a new build and notes stay attached to the right fix. Each fix also lists:

- **Finding with numbers:** the affected metric, whether it fails for real users, the estimated saving, devices and pages.
- **Files and elements:** the URLs of the affected files with the saving per file, and for layout shifts the HTML of the elements that move.
- **Action tailored to the platform:** for WordPress the text names a suitable plugin, for Next.js the matching framework feature. In the brief, a recommended plugin counts as a suggestion that you approve.
- **Acceptance criterion:** a target state you can check, for example "The LCP image has fetchpriority="high", no loading="lazy" and is referenced directly in the HTML".
- **Approval needed:** the agent only prepares changes to server, CDN, caching or third parties. It may deploy them only after you agree, because they touch infrastructure or business and legal questions.

This is a fix block from the brief of the sample report, shortened:

```
### PD-render-blocking-resources: Defer theme and plugin assets

- Affects: LCP
- Fails for real users: LCP
- Impact: estimated time saving 277 ms
- Approval needed: no

**Acceptance criterion:** The listed files no longer block first render
(defer/async, media attribute or critical CSS inlined); Lighthouse
reports no render-blocking resources for them.

| File | Estimated savings |
|---|---|
| `https://sample.perfdelta.com/assets/css/app.css` | 24 KB / 310 ms ~ |
```

Measured content marked as data: less exposure to prompt injection

A brief contains text that PerfDelta measured on your website. If an HTML snippet contained a sentence like "ignore all previous instructions", an agent could read it as a command. This is called prompt injection. The brief therefore puts HTML snippets, file names, selectors, URLs, vendor names and note texts into code blocks that their content cannot break out of. It also tells the agent to treat this content as data only and to report anything suspicious. This lowers the risk considerably but does not replace reviewing the commits before deployment.

Report back with a note: the next brief knows the status

After the work, the agent records one note per fix in a fixed format, for example `PD-render-blocking-resources done: inlined critical CSS (header.php:12)`. The status is one of done, skipped, needs_approval and in_progress. With the PageSpeed MCP server connected, the agent writes the note itself with the `add_note` tool. Without MCP, it lists one line per fix at the end of its answer, and you add them in the note field of the report history. A note holds at most 280 characters; several lines in one note are evaluated one by one.

The next brief compares the notes of the last 180 days with the date of the scan and distinguishes three cases:

- **Not re-measured yet:** the note is newer than the scan. The agent does not implement the fix again.
- **Still detected:** the fix was reported as done, but a later scan still finds the problem. The brief then asks to check whether the change is deployed or a cache still serves the old version.

- **Resolved:** reported as done, and the later scan no longer reports the fix.

Every fix is ticked off in a traceable way, and the confirmation comes from the measurement rather than from the agent's own account.

Download the agent brief or fetch it via MCP

In the report, the "Agent .md" button sits in the "Share this report" bar next to PDF and CSV. It appears for your own scans on every plan, including the free plan, and from the Solo plan for every measured domain. You can download the brief of the sample report without an account. With the MCP server, an agent fetches the brief itself with the `get_agent_brief` tool, for one of your own scans or the latest scan of a domain that was scanned before. Note status and budgets are only included for domains in your projects. The brief follows the language of the report and is available in English and German.

The instruction to the agent can be as simple as: "Read perfdelta-agent-brief.md and work through the fixes. Follow the guardrails and report every fix in the report-back format." Keep the file outside the repository or exclude it with `.gitignore`, because it describes a single scan and is outdated by the next one. Review the commits as you would a colleague's before you deploy.

Limits of the agent brief

- **Only what is in the repository:** the agent cannot see third-party files or settings in the CMS backend or the CDN. It marks such fixes as skipped and says where they probably need to be changed.
- **Savings are estimates:** they come from Lighthouse or PerfDelta and do not add up when several fixes affect the same file.
- **Local measurements only compare with each other:** Lighthouse on the agent's machine runs with a different CPU and network than PageSpeed Insights. The re-scan in PerfDelta after deployment is what counts.
- **Field data reacts slowly:** the Chrome UX Report covers 28 days. An improvement shows there gradually over up to four weeks, while the lab test shows it right after the re-scan.
- **The agent can be wrong:** guardrails and acceptance criteria reduce the risk but do not replace reviewing the changes before deployment.

All plans with the agent brief are on the pricing page. A free page speed test is enough for a first report.