Brandella Journal

FREE SEO AND AEO GROWTH ASSET 14 OF 36

Site Speed Wins Kit: Core Web Vitals Checklist, Fix Planner and LCP Breakdown

Free Core Web Vitals checklist in Excel: rate LCP, INP and CLS per template, find the slow part of LCP, and rank 38 proven fixes by impact and effort.

Updated 24 November 2026
Online version and downloads: https://brandella.in/resources/seo-aeo-assets/site-speed-wins/
Free for personal and client work. By Santaji Gadepatil, Founder of Brandella Journal.

What is in this kit

Item	What it is
Excel workbook, 10 tabs	Templates (100 rows: statuses, speed score, LCP breakdown, first fix and red flags), Speed Wins (500 ready rows with value, priority, quadrant, field data date and result), Fix Library of 38 fixes, Sprint Plan with a capacity check, Dashboard with two charts, Settings, AI Prompts, Checklist and Sources. Opens in Excel and Google Sheets.
Worked demo inside	8 template and device rows, 19 speed wins and 12 sprint tickets for a fictional coffee equipment store, so you see every status, score and result working before you add your own site.
Full guide (PDF)	This complete guide as a printable PDF, to share with developers, designers or clients.
22 AI prompts	Prompts to read PageSpeed Insights reports, explain the LCP breakdown, find slow interactions, plan CLS fixes and write tickets, including a 5 prompt chained workflow.
Printable checklist	The whole Core Web Vitals checklist on one page, to tick off for every template and after every release.

Most site speed advice is a long list of things that might help. This free **core web vitals checklist** is a working kit instead. You add your page templates with real field data, and an Excel workbook rates LCP, INP, CLS and TTFB with Google's thresholds, splits LCP into its four parts to show which one is slow, and suggests the first fix to try. Then every fix gets a value, a priority score and a quadrant, so you know which quick wins to ship this week and which big bets can wait.

It is built for the AI search era. Google says its core ranking systems look to reward a good page experience, and it lists a great page experience among the SEO basics that still matter for AI Overviews and AI Mode. Fast pages that serve their main content in plain HTML are also easier for AI crawlers to read, because the crawlers in one large study did not run JavaScript at all. The kit ends the guessing: it tells you when field data is ready to prove a fix worked, and whether it did.

How to use this site speed wins kit

1. **Set today's date and your capacity.** On Settings, set the date and how many effort points your team can ship each week. Leave Google's thresholds as they are.
2. **List your templates, not single URLs.** Speed problems live in templates. Add one row per template and device (home, product, category, article, checkout) with monthly visits and whether the page earns revenue.
3. **Add p75 field data.** From PageSpeed Insights or the Search Console Core Web Vitals report, add LCP, INP, CLS and TTFB at the 75th percentile. Mark lab only data as Lab only.
4. **Break LCP into its four parts.** Add the LCP element and the four LCP parts from the PageSpeed Insights LCP breakdown or the DevTools Performance panel. The sheet finds the part that is furthest over its ideal share.

5. **Log fixes from the library.** On Speed Wins, add the fixes that match the Biggest LCP gap and Red flags columns. Choose effort and confidence for each.
6. **Ship quick wins first.** Sort by Priority score. Put P1 quick wins on the Sprint Plan with acceptance criteria and a validation method, then copy the ticket text into your tracker.
7. **Prove it with field data.** Add the Shipped on date and the Before value. After 28 days, add the After value from field data. The Result column says Now good, Improved, or No improvement.

Why site speed still matters in the AI search era

Google is clear that speed is not everything: its page experience guide says search always seeks to show the most relevant content, even if the page experience is sub par, and that there is no single page experience signal. But the same page says Google's core ranking systems look to reward content that provides a good page experience, and Google's Core Web Vitals page recommends that site owners achieve good Core Web Vitals for success with Search.

The AI era adds two reasons to care:

- **The basics still decide AI visibility.** Google's guide to AI features and your website says no special optimization is needed for AI Overviews and AI Mode, and lists the SEO fundamentals that still apply, including providing a great page experience and making sure important content is available in textual form.
- **Many AI crawlers read only your HTML.** A Vercel and MERJ study from December 2024 found that none of the major AI crawlers it measured rendered JavaScript, including crawlers from OpenAI, Anthropic and Perplexity. Server rendered pages are usually faster for people too, so one fix helps both.

So a modern Core Web Vitals checklist asks three questions for every template: which metric fails, why it fails, and which fix gives the most improvement for the least work. The workbook answers all three.

What is inside the Core Web Vitals workbook

Tab	What it does	What you type
Templates	Rates LCP, INP, CLS and TTFB, gives a Core Web Vitals verdict and a speed score, splits LCP into four parts, names the biggest gap, suggests the first fix and lists red flags.	Template, device, visits, revenue page, p75 field data, LCP element and parts
Speed Wins	One row per fix. Looks up the metric and the template's status, then works out value, priority score, P1 to P4, a quadrant, the field data date and the result.	Fix ID, task, template, effort, confidence, owner, status, dates, before and after
Fix Library	38 fixes for LCP, INP, CLS and TTFB with the part they fix, how to check them, expected impact and typical effort.	Optional: your own fixes
Sprint Plan	Developer tickets with acceptance criteria and validation, plus a weekly capacity check.	Win ID, week, done when, method
Dashboard	16 key numbers, top 10 open wins, a template table, wins by quadrant and by metric, and two charts.	Nothing
Settings	Google thresholds, ideal LCP shares, impact, effort, confidence, traffic bands, priority rules and lists.	Optional tuning

Yellow cells are inputs. Grey cells under green headers are formulas, so you never break the logic by accident. Every dropdown is fed from Settings, so you can rename owners or add statuses in one place.

Core Web Vitals checklist thresholds

Every metric is judged at the 75th percentile of real visits, the way PageSpeed Insights reports field data from the Chrome UX Report over the previous 28 days.

Metric	What it measures	Good	Needs improvement	Poor
LCP (Largest Contentful Paint)	Loading: when the biggest image or text block shows	2.5 s or less	Over 2.5 s up to 4 s	Over 4 s
INP (Interaction to Next Paint)	Responsiveness: how fast the page reacts to clicks, taps and typing	200 ms or less	Over 200 ms up to 500 ms	Over 500 ms
CLS (Cumulative Layout Shift)	Visual stability: how much content jumps around	0.1 or less	Over 0.1 up to 0.25	Over 0.25
TTFB (Time to First Byte)	Server and network speed. Not a Core Web Vital, and marked experimental in PageSpeed Insights	800 ms or less	Over 800 ms up to 1,800 ms	Over 1,800 ms

A template passes only when every measured Core Web Vital is good. Like PageSpeed Insights, the sheet judges a page with no INP data on LCP and CLS alone. If LCP or CLS has no data, it says

Incomplete data instead of guessing.

The LCP breakdown and why it finds the real problem

LCP is not one number. Google's Optimize LCP guide splits it into four parts and gives a rough ideal share of the total for each. The guide says these are guidelines, not strict rules.

LCP part	What it is	Ideal share	First fix to try
Time to first byte	Until the first byte of HTML arrives	About 40%	Cache HTML on a CDN, remove redirects, speed up server work
Resource load delay	From TTFB until the browser starts loading the LCP image	Under 10%	Put the image in the initial HTML, add `fetchpriority="high"`, never lazy load it
Resource load duration	The time to download the LCP image	About 40%	Compress, use WebP or AVIF, serve responsive sizes from a CDN
Element render delay	From download until the element is painted	Under 10%	Cut render blocking CSS and JavaScript, server render, break up long tasks

For text LCP elements there is no image to load, so the two middle parts are zero and the time sits in TTFB and render delay. The guide also warns that download time is usually not the main bottleneck, which is why compressing images alone often disappoints. The sheet subtracts each ideal share from the real share and names the part that is furthest over.

How the template rating works (worked example)

In the demo, the Product page on mobile has 96,000 of the 303,500 monthly visits in the sheet, a 32 percent traffic share. Its p75 field data is LCP 3.6 s, INP 240 ms, CLS 0.18 and TTFB 900 ms. All three Core Web Vitals are Needs improvement, so the verdict is **Failed**.

Speed score. Each metric scores 100 at the good threshold, falls to 50 at the poor threshold, and keeps falling above it. LCP 3.6 s scores 50 + 50 x (4 minus 3.6) / (4 minus 2.5) = 63.3. INP 240 ms scores 93.3 and CLS 0.18 scores 73.3. The average is a speed score of **77**. The score is a kit assumption to compare templates, not a Google number.

LCP breakdown. The four parts are 900, 1,300, 600 and 800 ms, a total of 3,600 ms:

- Time to first byte: 25% (ideal about 40%), 15 points under
- Resource load delay: 36% (ideal under 10%), 26 points over
- Resource load duration: 17% (ideal about 40%), 23 points under
- Element render delay: 22% (ideal under 10%), 12 points over

The biggest gap is **resource load delay**: the browser waits 1.3 seconds before it even starts loading the main product image. The Red flags column explains why: "LCP image is lazy loaded. LCP image is

not in the initial HTML." The gallery slider builds the image with JavaScript. Compressing the image would barely help; making it discoverable would.

How fixes are ranked (worked example)

Every row on Speed Wins gets a value from 0 to 100 and a priority score:

- **Impact points** from the Fix Library: High 3, Medium 2, Low 1, divided by 3.
- **Status factor** for the metric the fix improves on that template: Poor 1, Needs improvement 0.6, Good 0.2, No data 0.5. A fix for a metric that is already good is worth little.
- **Traffic factor** from the template's share of visits: 30% or more 1, 15% or more 0.8, 5% or more 0.5, otherwise 0.2.
- **Revenue factor** 1.5 for pages that sell or capture leads, and a **confidence factor** of 1, 0.8 or 0.5.

Value = 100 x impact x status x traffic x revenue x confidence, capped at 100. Priority score = value divided by effort points (S 1, M 2, L 3, XL 5).

Worked example. SW-001 renders the first gallery image in HTML instead of the JavaScript slider. It is library fix SP01 with High impact, for LCP, which is Needs improvement on the Product page (Mobile). So value = 100 x 3/3 x 0.6 x 1 x 1.5 x 1 = **90**. Effort is S, so the priority score is also 90: **P1: do now**, and a **Quick win** because value is 30 or more and effort is 2 points or less.

Compare SW-007, rebuilding brew guides on a server rendered template. LCP is Poor there (4.6 s), but brew guides have only 7 percent of visits and confidence is Medium, so value is 100 x 1 x 1 x 0.5 x 1 x 0.8 = 40. Effort XL (5 points) brings the priority score to 8: **P3** and a **Big bet**. It is worth doing, after the quick wins. And SW-008, the Apply coupon fix in checkout, has only 3 percent of visits but the revenue factor lifts its value to 30, enough to make it a quick win.

Every factor lives on Settings. They are assumptions of this kit, not Google numbers, so tune them to your business.

Field data, the 28 day wait and the Result column

Lab tests in Lighthouse are great for debugging, but Google's reports use field data: the Chrome UX Report collects real visits over a rolling 28 day period. That means a fix you ship today shows fully in field data about 28 days later. The Search Console Core Web Vitals report works the same way: when you validate a fix, it starts a 28 day monitoring session for that issue.

So the sheet adds the field data window to the Shipped on date. In the demo, SW-003 added `fetchpriority="high"` on 20 November 2026. Field data is ready on 18 December, so on 24 November the Result column says "Wait for field data: 24 days left". After that date, add the After value and the sheet compares it with the good threshold:

Result	Meaning	What to do
Now good	After value is at or under the good threshold	Mark it Validated. Move to the next metric.
Improved, not good yet	Better than before, still over the threshold	Log the next fix for the same part of the metric.
No improvement: investigate	The same or worse	Check the fix really shipped, then recheck the cause. In the demo, the cookie banner was not the source of the layout shifts; the ads were.
Wait for field data	Less than 28 days since shipping	Use lab tests or your own RUM data meanwhile.

Column by column guide to the Speed Wins tab

Column	What goes in it	Tip
Win ID	Fixed ID such as SW-001.	Keep it: the Sprint Plan uses it.
Fix ID	Dropdown from the Fix Library, or blank for your own task.	A blank Fix ID counts as Medium impact on all metrics.
Fix or task	What will change, in your words.	Name the element: "main product image", not "images".
Template and device	Dropdown from the Templates tab.	Fix the same template on both devices with two rows.
Effort and Confidence	S, M, L or XL; High, Medium or Low.	Ask the developer, not the SEO.
Owner and Status	Who does it; Idea, Planned, In progress, Shipped, Validated, Dropped.	Shipped and Validated leave the priority list.
Shipped on, Before, After	Release date and the p75 value before and 28 days after.	Use the metric's unit: seconds for LCP, milliseconds for INP and TTFB.
Metric, Library fix, Expected impact	Automatic from the Fix Library.	Change impact in the library, not here.
Metric status now and the factors	Automatic from Templates and Settings.	Update field data monthly and every score moves.
Value, Priority score, Priority	Automatic. P1 at 30 or more, P2 at 12, P3 at 5.	Sort by Priority score.
Quadrant	Quick win, Big bet, Small task or Avoid for now.	Quick wins first, then one big bet per month.
Field data ready on, Result	Automatic once Shipped on is filled.	Set a calendar reminder for the ready date.

Quick wins by metric

These are the fixes Google's web.dev team lists in the most effective ways to improve Core Web Vitals and its LCP and INP guides. All of them are in the Fix Library.

- **LCP.** Make the LCP image discoverable in the HTML with `src` or `srcset`, add `fetchpriority="high"`, remove `loading="lazy"` from it, and preload it if CSS or JavaScript references it. Prefer server rendering over client side rendering. Our guide to lazy loading and LCP shows the most common mistake.
- **TTFB.** Use a CDN and cache HTML at the edge, even briefly, remove redirects and stop unique URL parameters from bypassing the cache. See browser caching explained.
- **INP.** Break up long tasks (a long task runs over 50 ms) by yielding to the main thread, remove unused JavaScript and tag manager tags, avoid layout thrashing and keep the DOM small. Our INP guide explains the three parts of an interaction: input delay, processing duration and presentation delay.
- **CLS.** Set width and height on images, reserve space for ads and embeds, and animate `transform` instead of properties like `top` or `margin`. Pages eligible for the back and forward cache restore instantly with no layout shifts.

Your first 30 days with this kit

- **Week 1: measure.** List your top templates per device, add p75 field data and the LCP breakdown. Read the Core Web Vitals verdict, Biggest LCP gap and Red flags for each.
- **Week 2: plan.** Log fixes from the library on Speed Wins, choose effort and confidence with your developer, and put the P1 quick wins on the Sprint Plan.
- **Week 3: ship.** Release the quick wins, fill Shipped on and Before, and check each one in a lab test right away.
- **Week 4: review.** Refresh field data, plan one big bet, and set reminders for each Field data ready on date.

After that, update field data monthly and run the checklist after every big release or theme change. Our guide on how to measure Core Web Vitals explains each data source.

The chained prompt workflow (from report to tickets)

Prompts 18 to 22 below are designed to run in order in one AI chat:

1. **Brief the assistant** on your site, stack and templates.
2. **Read the data** you paste from PageSpeed Insights or Search Console and rate each template.
3. **Find causes** from the LCP breakdown, red flags and long tasks.
4. **Pick fixes** from the library IDs with effort and confidence.
5. **Write tickets** in the same columns as the workbook, ready to paste.

The AI explains and drafts; field data decides. Never accept a speed number from an AI assistant that did not come from your own reports.

Core Web Vitals mistakes to avoid

- **Chasing a 100 lab score.** Lighthouse runs on one simulated device. Google's reports use real visits at the 75th percentile, and Google says good report results do not guarantee top rankings.
- **Testing only the home page.** Product, category and article templates usually carry most visits. Work by template.
- **Compressing images and stopping there.** For many pages the delay before the image starts loading is the real problem, not the download.
- **Lazy loading the hero image.** It delays LCP. Lazy load only images below the fold.
- **Ignoring INP.** Third party tags, chat widgets and heavy filters often make clicks feel slow. Audit tags before adding new ones.
- **Judging a fix after two days.** Field data needs about 28 days. Use the Field data ready on date.
- **Hiding content behind JavaScript.** It slows rendering and can hide content from crawlers that do not run JavaScript.

AI prompts

Replace the parts in [square brackets]. Always check the output against real data.

1. Explain a PageSpeed Insights report

```
Here is a PageSpeed Insights report for [URL] on [mobile or desktop]: [paste field data
and the main diagnostics]
Explain in plain language which Core Web Vitals pass or fail at the 75th percentile,
what is causing each failure,
and the 3 fixes most likely to help. Do not invent numbers that are not in the report.
```

2. Read the LCP breakdown

```
My LCP element is [image, text or video] and the LCP breakdown is: time to first byte
[ms], resource load delay [ms],
resource load duration [ms], element render delay [ms]. Compare each part with the
guideline shares
(about 40%, under 10%, about 40%, under 10%), name the part that is furthest over, and
list the fixes for that part.
```

3. Make the LCP image discoverable

```
Here is the HTML for my page header and hero area: [paste HTML]
Find out whether the LCP image is in the initial HTML, lazy loaded, set as a CSS
background or built by JavaScript.
Rewrite the markup so the image is discoverable early with fetchpriority high, without
lazy loading.
```

4. Find render blocking resources

```
Here is the head section of my page: [paste head]. List every stylesheet and script that blocks rendering,
say which ones can use defer or async, which CSS could be inlined because it is small, and what to load later.
```

5. TTFB and caching plan

```
My site runs on [CMS or framework] with [hosting or CDN]. TTFB at p75 is [ms] on [template].
Suggest a caching plan: which pages can be cached at the edge and for how long, what must never be cached,
and how to stop URL parameters like utm tags from bypassing the cache.
```

6. Find long tasks behind a slow INP

```
This is a DevTools Performance trace summary (or my own notes) for the interaction "[click or tap]" on [page]: [paste]
Split the delay into input delay, processing duration and presentation delay. Point to the long tasks
(over 50 ms) and suggest where to yield to the main thread.
```

7. Break up a long event handler

```
Here is my event handler code: [paste JavaScript]. It runs when a user [action] and makes INP slow.
Rewrite it so the visual update happens first and non urgent work runs after yielding to the main thread.
Keep the behaviour the same and explain each change.
```

8. Tag manager clean up

```
Here is the list of tags in my tag manager with what each one does: [paste list]
Group them into keep, review and remove. Flag tags that fire on every page but are needed only on some pages,
and tags that could load after the page is interactive.
```

9. Third party script audit

```
These third party scripts load on [template]: [paste list with sizes if known]. For each one, say whether it can
be removed, delayed until interaction (a facade), loaded later, or must stay. Note the likely effect on INP and LCP.
```

10. Find layout shift causes

```
CLS on [template] is [value] at p75. The page has [ads, embeds, banners, web fonts, images without sizes].
List the most likely causes of layout shifts in order, how to confirm each in DevTools, and the fix for each.
```

11. Reserve space for ads and embeds

```
My [ad slot, video embed or widget] loads after the page and pushes content down. Its usual sizes are [sizes].
Write the CSS that reserves space for it on mobile and desktop without leaving large empty gaps.
```

12. Image optimization plan

```
Here is a list of the largest images on [template] with file type, size in KB and displayed width: [paste list]
Suggest the right format (WebP or AVIF), target file size, srcset widths and which images should be lazy loaded.
```

13. Web font plan

```
My site loads these fonts: [paste font families, weights and hosts]. Suggest how to cut the number of files,
which font to preload, the font-display value to use and how to reduce layout shifts when the font swaps.
```

14. Back and forward cache check

```
Chrome says my page is not eligible for the back and forward cache for these reasons: [paste reasons].
Explain each reason in plain words and what a developer must change to fix it.
```

15. Server rendering decision

```
My [template] builds its main content with client side JavaScript using [framework]. Explain the trade offs of
server rendering, static generation and prerendering for this page, the likely effect on LCP, and how to make the
main text and links visible in the initial HTML for crawlers that do not run JavaScript.
```

16. Estimate effort for a fix list

```
Here are speed fixes for my site: [paste fixes]. Our stack is [stack] and our team is [team size].
Give each fix an effort of S (under a day), M (a few days), L (one to two weeks) or XL (a project),
and a one line reason. Flag any fix that needs design or business approval.
```

17. Explain speed work to a manager

```
Write a short update for my manager about our site speed work. Data: [paste dashboard numbers].
Explain in plain language what passed, what failed, what we shipped, what we are waiting to measure and what is next.
No jargon without a one line explanation.
```

18. Workflow 1 of 5: brief the assistant

```
You are my web performance assistant. My site: [site and business]. Stack: [CMS,
framework, hosting, CDN].
Main templates: [list]. Goal: pass Core Web Vitals on mobile for the templates with the
most visits.
Confirm you understand, then wait for the next step.
```

19. Workflow 2 of 5: read the data

```
Here is p75 field data per template and device: [paste table with LCP, INP, CLS, TTFB
and visits].
Rate each metric as good, needs improvement or poor (LCP 2.5 s and 4 s, INP 200 and 500
ms, CLS 0.1 and 0.25),
and rank the templates by how much traffic is affected.
```

20. Workflow 3 of 5: find causes

```
For the worst three templates, here are the LCP breakdowns and diagnostics: [paste].
Name the main cause for each failing metric, using the LCP parts, long tasks and layout
shift sources.
```

21. Workflow 4 of 5: pick fixes

```
Pick the fixes for these causes from this list of fix IDs: [paste Fix Library IDs and
names].
For each fix give the template, effort (S, M, L or XL) and confidence (High, Medium or
Low), and say why.
```

22. Workflow 5 of 5: tickets for Excel

```
Turn the fixes into a table I can paste into my Speed Wins tab, with these columns
separated by tabs:
Fix ID, Fix or task, Template and device, Effort, Confidence, Owner. Then write
acceptance criteria for each fix
that a developer can test, and say which validation method to use.
```

Checklist

- [ ] Set today's date and weekly capacity on the Settings tab
- [ ] Listed every main template on mobile and desktop with monthly visits
- [ ] Marked which templates earn revenue or leads
- [ ] Added p75 field data for LCP, INP, CLS and TTFB from PageSpeed Insights or Search Console
- [ ] Marked templates that have lab data only
- [ ] Recorded the LCP element and checked it is in the initial HTML and not lazy loaded
- [ ] Added the four LCP parts and read the Biggest LCP gap for each template

- [ ] Checked render blocking CSS and JavaScript in the head
- [ ] Audited tag manager and third party scripts for INP
- [ ] Checked images, ads and embeds have reserved space for CLS
- [ ] Checked the main content and links are in the initial HTML for crawlers that do not run JavaScript
- [ ] Logged fixes from the Fix Library with effort and confidence
- [ ] Put P1 quick wins on the Sprint Plan with acceptance criteria
- [ ] Filled Shipped on and Before for every released fix
- [ ] Added After values from field data once the 28 day window passed
- [ ] Set a monthly date to refresh field data

Frequently asked questions

Is this Core Web Vitals checklist really free?

Yes. The Excel workbook, PDF guide, prompts and checklist are free to download and use for your own sites and for clients, with no sign up.

Where do I get field data if my pages have little traffic?

When a URL has no data in the Chrome UX Report, PageSpeed Insights falls back to origin level data for the whole site. The Search Console Core Web Vitals report groups URLs with a similar experience. If neither has data, use lab data and mark the row Lab only, or collect your own data with real user monitoring.

Do Core Web Vitals decide my rankings?

No single signal does. Google says its core ranking systems look to reward a good page experience, but it always seeks the most relevant content first, and good Core Web Vitals reports do not guarantee top rankings. Treat speed as a tie breaker and a conversion win.

Why does the sheet wait 28 days before judging a fix?

Google's field data comes from the Chrome UX Report, which covers the previous 28 days of real visits. Until a full window has passed, the numbers still mix visits from before and after your fix.

Does the workbook work in Google Sheets?

Yes. Upload the .xlsx file to Google Drive and open it with Google Sheets. Formulas, dropdowns and colours carry over. The charts may need a quick refresh.

Does site speed matter for AI search?

Google lists a great page experience among the SEO basics that still apply to AI Overviews and AI Mode. Separately, a 2024 study found major AI crawlers did not run JavaScript, so fast, server rendered HTML helps those crawlers read your content.

Sources

- Google Search Central: Understanding Core Web Vitals and Google search results
  https://developers.google.com/search/docs/appearance/core-web-vitals
- Google Search Central: Understanding page experience in Google Search results
  https://developers.google.com/search/docs/appearance/page-experience
- Google for Developers: About PageSpeed Insights (field data and thresholds)
  https://developers.google.com/speed/docs/insights/v5/about
- web.dev: Optimize Largest Contentful Paint
  https://web.dev/articles/optimize-lcp
- web.dev: The most effective ways to improve Core Web Vitals
  https://web.dev/articles/top-cwv
- Vercel: The rise of the AI crawler
  https://vercel.com/blog/the-rise-of-the-ai-crawler

More free templates, checklists and prompts: https://brandella.in/resources/seo-aeo-assets/