PRACTITIONER RESEARCH WHITE PAPER · NOVEMBER 2026

Do AI Crawlers Render JavaScript in Late 2026?

What 225,276 Verified Origin Server Log Requests Reveal About Headless Queues, Live Prompt-Time Fetchers, and How Site Owners Can Audit Execution on Their Own Logs

Tygart Media · tygartmedia.com · Anonymized Multi-Property Telemetry Edition

225,276	0.0%	< 1.0%	6
Verified Origin Requests	**Live-Agent JS Execution**	**OpenAI Deferred Render**	**JS-Capable Bot Families**
8 anonymized properties	0 of 623 prompt-time hits	0.98% OAI-SearchBot queue	Google, Bing, Apple, Meta, Baidu, OAI

Executive Summary

A widely shared 2024 Vercel and MERJ benchmark concluded that none of the major AI crawlers execute JavaScript (excluding Googlebot), leaving client-side rendered pricing pages and React/Vue shells as blank grey skeleton screens. To test whether that rule still holds in late 2026 — and whether site owners can measure JavaScript execution reliably from their own server logs — we audited **225,276** CIDR- and ASN-verified non-operator HTTP requests across eight independent WordPress content properties, paired with **1,220** client-side frontend analytics sessions.

Our empirical answer is split cleanly in two: first, the 'Googlebot is the only exception' shorthand is outdated. Across our verified logs, **six distinct crawler ecosystems** executed JavaScript at runtime: **Google-NotebookLM**, **Applebot** (Apple Intelligence), **Bingbot** (Microsoft Copilot), **Googlebot / GoogleOther**, **Meta-WebIndexer / Meta-ExternalAgent** (selective `Chrome/145` passes), and **OpenAI's OAI-SearchBot/1.4 and GPTBot/1.4**, which now route a small deferred queue through headless Linux `Chrome/148`. Second, the practical warning against client-side skeleton screens remains **100% urgent**: every live prompt-time citation fetcher (**ChatGPT-User**, **Claude-User**, **Perplexity-User**, **Claude-Code-CLI**, and **ExaUser**) logged **0 runtime JavaScript executions across 623 verified HTML fetches**, and even OpenAI's deferred headless queue touches **less than 1%** of fetched pages.

1. Why the 2024 'Only Googlebot' Rule Is Half Outdated — Yet Still Dangerous

Web builders often confuse two very different bots under the same vendor brand: **indexers** that crawl on a background schedule (like `OAI-SearchBot`, `GPTBot`, `Bingbot`, or `Googlebot`), and **live prompt-time fetchers** that retrieve a single page while a user waits for a chat answer (`ChatGPT-User`, `Claude-User`, `Perplexity-User`). Spinning up a headless Chromium container consumes roughly an order of magnitude more CPU, RAM, and wall-clock latency than a raw HTTP GET.

When a user asks an assistant to verify current plan tiers and seat rates, a live fetcher operating inside a 1-to-3-second latency budget simply cannot wait for a client-side JavaScript bundle to hydrate skeleton boxes. Meanwhile, background search ecosystems (`Bingbot`, `Applebot`, `Googlebot`) already run full rendering pipelines, and OpenAI has quietly begun selective deferred rendering for roughly `0.3%` to `1.0%` of pages — not enough to rescue a JavaScript-only site, but unmistakably visible in origin logs.

KEY TAKEAWAY FOR PUBLISHERS & SAAS TEAMS

If your pricing, product specifications, or comparison tables render only after client-side JavaScript runs, 100% of live answer-time AI fetchers (and ~99%+ of OpenAI, Anthropic, and Perplexity index crawls) see empty skeleton placeholders. Server-side rendering (SSR), static HTML, and clean Markdown endpoints remain mandatory.

2. Log-Forensic Methodology: Separating .js Downloads from Real Execution

Standard web access logs (`combined` format) record every HTTP request, status code, referrer, and User-Agent string — they do not record CPU instructions inside a visitor's browser. That creates a classic measurement trap: seeing `GET /assets/app.js` in a server log only proves a crawler **downloaded** the script file. As the 2024 Vercel/MERJ study documented, bots frequently fetch static `<script src>` URLs listed in HTML and store them as plain text without ever executing them.

To prove genuine V8 or WebKit JavaScript execution from origin logs alone, we tracked **three runtime-only forensic signals** across eight anonymized WordPress properties (one 7-day full capture and seven 24-hour snapshots totaling `225,276` non-operator lines):

- **Signal A — Runtime DOM Script Injection (`wp-emoji-release.min.js`):** On all eight properties, `wp-emoji-release.min.js` never appears in any static `<script src>` tag. It lives only inside an inline JSON configuration object. A client requests that file **only** when a JavaScript engine parses the inline bootstrap, runs a `<canvas>` 2D emoji rendering test, and dynamically injects a new `<script>` node into the DOM.
- **Signal B — Runtime Telemetry XHR (`clientId` UUID + `navigator` JSON):** A third-party storefront/analytics script executes inside the browser runtime, generates a fresh UUIDv4, reads `navigator.userAgent`, `os`, and `window.location.href` from the live DOM, and fires a relative XHR `GET` request back to the origin carrying the rendered browser fingerprint in the query string.
- **Signal C — Frontend Analytics Session Beacons (`1,220` sessions):** Across 7 days of client-side behavior tracking on seven properties, **1,220 of 1,220 recorded sessions (100%)** came from human desktop/mobile browsers (`0` AI bots). Even headless-capable bots either block analytics POSTs or tear down the page before session beacons flush.

FIGURE 1: THREE SERVER-LOG SIGNATURES THAT SEPARATE HTML SCRAPERS FROM REAL JS EXECUTION

TIER 1: RAW HTML ONLY	TIER 2: STATIC .JS DOWNLOAD	TIER 3: RUNTIME JS EXECUTION
0% JS Visibility	False-Positive Risk in Logs	Confirmed Browser Engine
Fetches document HTML/Markdown	Fetches static <script src> URLs	Executes inline script -> injects DOM
Zero .js, .css, or font subrequests	Treats .js files as plain text	Triggers wp-emoji-release.min.js
Blind to grey skeleton placeholders	Does NOT execute V8 / WebKit DOM	Fires UUIDv4 + navigator XHR call
Examples: ChatGPT-User, Claude-User, PerplexityBot, Perplexity-User, Exa	No dynamic scripts or runtime XHR	Leaks container engine (Blink 148)
	Seen in periodic scraper sweeps	Google, Bing, Apple, NotebookLM, Meta*

The Spoofed-User-Agent Quarantine Gate (Why Raw Greps Lie)

Raw User-Agent log greps without IP verification produce false positives. Across our dataset, three `.js` requests carrying `ChatGPT-User` (2 hits on `/__env.js` and `/config.json.js`) and `PerplexityBot` (1 hit on `/runtime-config.js`) actually came from unverified cloud scanner IPs outside OpenAI and Perplexity JSON IP feeds. Every hit in this study passed official vendor CIDR matching (`openai.com/*.json`, `claude.com/crawling/bots.json`, `perplexity.ai/perplexitybot.json`, Google crawler JSONs) or ASN/rDNS verification before counting.

3. Empirical Matrix Across 14 Verified Search & AI Bot Families

Every row below reflects CIDR- or ASN-verified traffic after excluding operator tests and quarantined exploit scanners. Notice the stark architectural divide between background search engines and live prompt-time citation tools:

Verified Bot Family	Primary Role	HTML	.js Hits	Runtime Canaries	Observed JS Execution Verdict
Google-NotebookLM	Live prompt / URL add	19	66	16 (XHR)	Full headless render (~84%+)
Applebot	Search / Apple Intel.	674	124	18 (emoji)	Regular WebKit browser render
Bingbot	Bing / Copilot	4,042	191	19 (emoji)	Two-wave Chrome/136 render
Googlebot & GoogleOther	Search / Gemini / AI	1,509	53	5 (mixed)	Evergreen Chrome/157 WRS
Meta-WebIndexer / ExtAgent	Meta AI / training	6,589	50	11 (+11 woff2)	Selective Chrome/145 (~0.2-0.5%)
Baiduspider-render/2.0	Baidu Search	1,276	99	11 (emoji)	Dedicated render crawler
OAI-SearchBot/1.4	ChatGPT Search index	820	0	8 (XHR)	Deferred Linux Chrome/148 (0.98%)
GPTBot/1.4	OpenAI model training	2,568	6	7 (4 em + 3 XHR)	Deferred Linux Chrome/148 (0.27%)
AhrefsBot/7.0	SEO backlink audit	1,665	7	3 (emoji)	Tiny diagnostic sample (0.18%)
PerplexityBot & -User	Perplexity AI	3,790	0	0	0.0% JS (HTML, MD, images only)
ClaudeBot & -SearchBot	Anthropic indexers	1,340	1 (304)	0	0.0% JS execution (static only)
ChatGPT-User	Live ChatGPT fetch	358	0	0	0.0% JS (1-3s latency budget)
Claude-User & Code-CLI	Live Claude fetch/CLI	265	0	0	0.0% cloud JS (HTML/MD only)
Amazon / AzureAI / Exa*	Search / RAG indexers	4,840	0	0	0.0% JS (*plus Duck/Yandex/Semrush)

4. Inside OpenAI's Two-Wave Queue vs. Answer-Time Latency Budgets

The clearest late-2026 surprise in our logs comes from OpenAI. Across `820` `OAI-SearchBot/1.4` HTML fetches, `99%` were single raw HTTP requests with zero CSS, image, or script loads. Yet on **8 pages** (`0.98%`), a two-wave pattern appeared across verified `openai.com/searchbot.json` IPs (`74.7.243.x` -> `74.7.229.x`):

- **Wave 1 (`T+0m`):** An `OAI-SearchBot` node fetches the raw server HTML (`200 OK`) with zero subresources.
- **Wave 2 (`T+91m`):** A second `OAI-SearchBot` node sends an HTTP `User-Agent` claiming macOS `Chrome/131.0.0.0`, but the executed telemetry XHR reveals the true container runtime inside V8: **Linux x86_64 headless** `Chrome/148.0.0.0` **(Blink 148)** reporting the exact `window.location.href` from the live DOM. `GPTBot/1.4` showed the same `Chrome/148` signature at `0.27%` (7 canary hits across `2,568` HTML fetches).

Crucially, **neither OpenAI, Anthropic, nor Perplexity renders JavaScript during live user-prompt browsing**. When a user triggers `ChatGPT-User` (358 HTML hits), `Claude-User` (162 HTML hits), `Claude-Code-CLI` (103 HTML hits), or `Perplexity-User` (all `0` JS), the agent fetches raw HTML or Markdown twins only. Meanwhile, **Google-NotebookLM** acts like an interactive cloud browser when a user explicitly adds a source URL (16 runtime XHR chains across 19 HTML fetches), while local **Claude Desktop** apps on residential networks occasionally render via their embedded Chromium frame (3 hits).

5. Five-Step Self-Audit Checklist for Engineering & Content Teams

You do not need expensive third-party crawler simulators to verify what AI engines see on your own site. Run these five concrete checks against your live URLs and origin access logs:

1. **Run the `curl` / Disable-JS Skeleton Test on Revenue Pages:** Fetch your pricing, documentation, and comparison pages with plain `curl` (or Chrome DevTools `Disable JavaScript`). If prices, feature matrices, or product tables turn into blank grey skeletons, `100%` of live answer-time AI fetchers and `~99%` of AI training/search crawlers cannot read them.
2. **Grep Logs by Verified Vendor IP Ranges, Never Raw User-Agents:** Download official JSON CIDR feeds (`openai.com/searchbot.json`, `claude.com/crawling/bots.json`, `perplexity.ai/perplexitybot.json`, Google crawler JSONs) before measuring bot behavior. Unfiltered UA greps misclassify exploit scanners probing `/__env.js` or `/runtime-config.js` as real AI crawlers.
3. **Instrument a Runtime-Only Execution Canary:** Do not count static `<script src>` hits as JavaScript rendering. Watch for scripts injected dynamically by inline JS, CSS font loads (`.woff2`) referenced inside external stylesheets, or lightweight first-party XHR beacons that carry a runtime UUID and `window.location.href`.
4. **Ship Server-Rendered HTML Tables & Semantic Markup First:** Move core pricing tiers, specifications, FAQs, and JSON-LD directly into the initial HTTP `200` HTML payload via SSR, SSG, or ISR so zero hydration step is required for text extraction.
5. **Pair Critical Pages with Clean Markdown Twins & `/llms.txt`:** Live prompt fetchers and coding CLIs (`Claude-User`, `Claude-Code-CLI`, `ChatGPT-User`) heavily favor compact, token-efficient text. Serving clean `.md` twins and structured HTML tables ensures fast, accurate retrieval even on strict 1-second latency budgets.

Primary Verification Sources

- **Vercel & MERJ (2024 AI Crawler Benchmark):** vercel.com/blog/the-rise-of-the-ai-crawler — Established that AI crawlers download `.js` files as text without executing them.
- **Google Search Central (JavaScript SEO & Crawlers):** developers.google.com/search/docs/crawling-indexing/javascript/javascript-seo-basics and google-user-triggered-fetchers.
- **Applebot Documentation (WebKit Rendering):** support.apple.com/en-us/119829 — Documents browser rendering and requirement for unblocked JS/CSS.
- **Microsoft Bingbot Verification:** bing.com/webmasters/help/how-to-verify-bingbot-3905dc26.
- **Official AI Crawler CIDR JSON Feeds:** openai.com/searchbot.json, openai.com/gptbot.json, openai.com/chatgpt-user.json, claude.com/crawling/bots.json, and perplexity.ai/perplexitybot.json.

ABOUT THIS RESEARCH & REPRODUCIBILITY NOTE

Published by Tygart Media (tygartmedia.com) on November 23, 2026. All traffic counts reflect anonymized server access logs across eight independent WordPress properties audited with strict vendor IP/ASN verification. Client identities, internal hostnames, and full IP addresses have been omitted.