Example Shop — PERFORMANCE DIAGNOSTIC

Performance & Core Web Vitals

External preliminary evaluation — publicly observable signals

A focused audit of loading, interactivity and visual stability on the mobile storefront — measured with Lighthouse (3-run median), PageSpeed Insights (lab) and CrUX (real-user field data).

Client: Example Shop | **Domain:** example-shop.com

Date: 18 July 2026

Author: Yassin BELHAJ SALAH

E-commerce Solutions Architect

Executive summary

Scope & method. This is an **external, preliminary evaluation** based only on publicly observable signals — no access to your code, admin, infrastructure, analytics or monitoring. It was produced with an AI-assisted workflow and manual validation of the retained findings. Every figure below is a **real measurement** (Lighthouse local 3-run median, PageSpeed Insights lab, CrUX field data) or an attributed benchmark — nothing is estimated by formula. Conclusions requiring internal access are framed as *observation → hypothesis to validate*. The audit covers the **home storefront tested on 18 July 2026 (mobile)**; security, SEO/GEO and accessibility are out of this performance audit's lane.

The headline — read lab and field together

In the **synthetic lab** (throttled mobile), Example Shop's storefront scores **poor (Lighthouse)**: a performance score of **27** (PSI lab / Google), an **LCP** of **21.2 s** and a **TBT** of **2,920 ms**. Taken alone, that looks alarming. But the **field data (CrUX, real users)** tells the honest, more nuanced truth:

PSI LAB SCORE	FIELD LCP (REAL USERS)	FIELD INP (REAL USERS)
27	1.7s	388ms
poor — throttled mobile	FAST — CrUX p75	needs-improvement

FIELD CLS (REAL USERS)
0.09
FAST — a strength

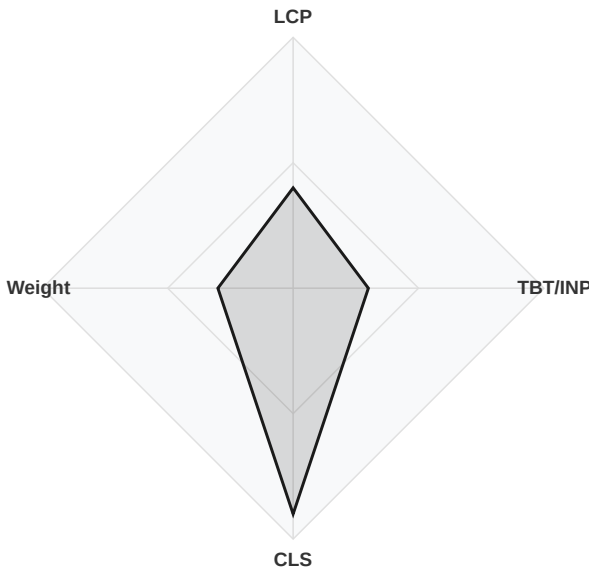
Your real users already experience a **fast first paint** (LCP 1.7 s, well inside Google's 2.5 s "Good" band) and a **stable layout** (CLS 0.09). So the *poor* lab LCP is a **fragility signal, not a lived crisis** — it shows how little headroom you have when a device is slow or the cache is cold. The one place real users **do** feel friction is **interactivity: INP at 388 ms** sits in the *needs-improvement* band (Google's "Good" bar is 200 ms). And every lab axis points at the same root cause — **too much JavaScript on the critical path**.

Performance radar (measured ratings

What's already excellent

- **Visual stability** — CLS 0.09 field (FAST) & ~0.05 lab (good (Lighthouse)).
- **Modern image delivery** — the CDN auto-serves **WebP** (verified).

/10)



CLS is the clear strength; LCP is fragile (poor lab / fast field); interactivity & weight are the work.

- **Fonts done right** — self-hosted, preloaded, `font-display: swap`.
- **Modern transport** — Cloudflare edge, HTTP/2 + HTTP/3.
- **Clean runtime** — no console errors, no duplicated JS.

Where the leverage is

- Critical-path & unused **JavaScript** (~1,780 KiB unused).
- **Third-party** scripts (2,577 KiB, ~3.2 s blocking).
- An entry **redirect** costing ~1.7–1.9 s in the lab.

The opportunity in one line. Your delivery fundamentals (WebP, HTTP/3, stable layout, clean fonts) are already where the market expects them. The single highest-leverage programme is **JavaScript discipline** — reduce and defer what runs before the page is interactive. That directly targets the one metric your buyers actually feel (INP) and removes the fragility behind the *poor* lab LCP. **27.5 man-days** of work is identified (full backlog); **~20.5 man-days** form the prioritised short/mid-term roadmap.

Your architecture — what we detected

Reading your storefront's own signals, Example Shop runs a **modern headless commerce stack** — a sophisticated setup, and the right lens for every finding in this report:

Layer	Detected technology	Evidence (observed)
Storefront (frontend)	**Next.js (React)** — headless	`_next/image`, `_next/static/chunks/framework-*`, `_app-*`
Content / media	**Contentful** CMS	hero image sourced from `images.ctfassets.net`
Commerce & checkout	**Shopify**	`powered-by: Shopify`, `us.checkout.example-shop.com`, open `products.json`
Edge / delivery	**Cloudflare**	`server: cloudflare`, HTTP/2 negotiated, `alt-svc: h3` (HTTP/3)

Asset CDN	First-party `cdn.example-shop.com`	self-hosted fonts & static assets

> **Why this matters for performance.** On a headless Next.js storefront, the browser downloads a JavaScript application and **hydrates** it before the page becomes fully interactive. This is powerful for experience and flexibility — but it puts **JavaScript on the critical path**. So your performance ceiling is set by the *frontend app and the third-party scripts it loads*, not by Shopify's commerce infrastructure. Every recommendation here is aimed at the frontend/JS layer; none of them is "Shopify is slow" — it isn't.

A note on honesty: the initial HTML shell is delivered quickly (TTFB ~0.17 s on the tested request), so **initial HTML-shell delivery is not the measured bottleneck**. That does not clear the whole infrastructure — APIs, hydration payloads and third-party endpoints are not exercised by a shell request — but the dominant observed cost is **client-side rendering / JavaScript execution**, and that is where this report concentrates.

Core Web Vitals — the measured verdict

All values below are real measurements. Lab = throttled mobile emulation; Field = CrUX, aggregated from your actual visitors (75th percentile).

Metric	Lighthouse (lab, local median, 18 Jul 2026)	PSI lab (Google)	Field — CrUX (real users)	Google "Good" bar
Performance score	25	27	overall **AVERAGE**	90+
LCP (loading)	25.6 s — poor (Lighthouse)	21.2 s — poor	**1.72 s — FAST**	≤ 2.5 s
TBT / INP (interactivity)	6,360 ms TBT — poor (Lighthouse)	2,920 ms TBT — poor	**INP 388 ms — needs-improvement**	INP under 200 ms
CLS (stability)	0.05 — good (Lighthouse)	0 — good	**0.09 — FAST**	below 0.1
FCP / Speed Index	FCP 11.4 s · SI 17.7 s	FCP 5.0 s · SI 16.1 s	—	FCP ≤ 1.8 s

Lab vs field — the honest reading. A *poor* lab LCP together with a **FAST** field LCP means your typical visitor (warm cache, capable device, landing on the canonical URL) is served quickly — the lab exposes the **worst case**. Do not dramatise it, and do not dismiss it: it is the margin you lose the moment you add a feature or a network gets slow. **TBT is only a lab proxy for interactivity; INP is the real field metric** — and INP (388 ms) is the one Core Web Vital where your real users sit in *needs-improvement*. We therefore do not claim "all Core Web Vitals are poor": all *measured laboratory* axes except CLS are in the poor band, while real-user INP is the field-confirmed weakness to fix.

CLS	INP 388ms	LCP lab
GENUINE STRENGTH — PROTECT IT ON ANY REDESIGN	THE FIELD-CONFIRMED FRICTION — PRIORITY	FRAGILE — FAST TODAY, NO HEADROOM

LCP — the largest element, phase by phase

Lighthouse names the LCP element for us: it is the **hero image**, rendered by the Next.js image component (`<img data-nimg="fill" … _next/image?url=…ctfassets.net…>`), i.e. a Contentful-sourced banner. Breaking its timing into phases tells us exactly where the time goes:

LCP phase	Share	What it means here
TTFB	3%	Server sends the first byte quickly — not the problem.
Load Delay	**27%**	The hero is **discovered late** — it is not preloaded and carries no priority hint.
Load Time	1%	Once requested it downloads fast (WebP + CDN working well).
Render Delay	**69%**	The image **cannot paint until JavaScript hydration completes** — the dominant cost.

Reading it plainly: the hero image itself is not heavy and downloads fast — the delay is that (a) it is found late and (b) it waits behind JavaScript before it can appear. Two levers follow directly, in order of size: **reduce the JS on the critical path** (the 69% render-delay lever, see CT-01) and **prioritise the hero** (the 27% load-delay lever, CT-05). Both are backed by measurement, and the hierarchy is a judgement grounded in the phase data — not a reflex.

CT-05 : Prioritise the LCP hero image

Observation. On the tested page there is **no image preload and no `fetchpriority`** anywhere in the document (0 occurrences); the hero relies on normal discovery, which the phase data ties to the 27% Load Delay.

Risk. The first meaningful pixel arrives later than it needs to on cold loads, on ad/shared-link landings and on slower devices — precisely the fragility the lab exposes.

Solution (validate against your Next.js pipeline). Mark the above-the-fold hero as high priority and preload it. With the Next.js image component this is typically the `priority` prop, which emits both a preload and a high fetch priority:

```
// Next.js Image — hero above the fold (validate against your
component/version)
<Image src={heroUrl} alt="…" fill sizes="100vw" priority />

// Equivalent raw markup the framework should emit:
<link rel="preload" as="image" href="/_next/image?url=…&w=1200"
fetchpriority="high">
<img … fetchpriority="high" decoding="async">
```

Benchmark (attributed, directional): in Google's own test, adding `fetchpriority="high"` to the LCP image improved load from 2.6 s to 1.9 s (~700 ms) — a single-attribute win. The exact gain on your site is measured after the change, not projected here.

Frontend Confirmed — **1 man-day**

Expected gain: earlier hero paint (attacks the 27% Load Delay); measurable as −X ms on LCP after deployment.

JavaScript & the main thread — the core programme

This is where the report earns its keep. The weight of your page is **JavaScript-dominant**, and the blocking time is concentrated in a handful of named third parties. These are the measurements:

3,389 KiB
SCRIPT TRANSFERRED (32 FILES) — THE #1 WEIGHT

~1,780 KiB
UNUSED JAVASCRIPT (LIGHTHOUSE)

2,577 KiB
THIRD-PARTY (~40% OF THE PAGE)

The largest single measured opportunity on the page is **unused JavaScript: ~1,780 KiB / ~3.75–3.99 s**. Lab main-thread time is dominated by Script Evaluation (~15.6 s under 4× CPU throttling) — the same cost your users feel as the 388 ms INP.

Third-party scripts — named, measured (Lighthouse entities)

Each Lighthouse "entity" is an **aggregate** — e.g. "Google Tag Manager" includes the container *and* the tags it fires. Figures are transfer / main-thread blocking as attributed to the entity.

Third party (entity)	Blocking	Transfer	Role
Google Tag Manager (+ its tags)	1,279 ms	852 KiB	tag management
Intercom	777 ms	289 KiB	live chat
AWS WAF (`awswaf.com`)	757 ms	583 KiB	bot / fraud protection JS

Facebook / Meta pixel	256 ms	166 KiB	marketing tag
mParticle	140 ms	138 KiB	customer data platform

Also present: Shopify (395 KiB), OneTrust/Optanon consent, Google Ads/Doubleclick, Google Analytics, Datadog RUM, Algolia (search), shop.app. Detected in raw HTML but active usage not confirmed: Attentive (SMS), Bazaarvoice (reviews), Riskified (anti-fraud). **Total third party: ~3,209 ms blocking / 2,577 KiB.**

CT-01 : Cut critical-path JavaScript execution Top priority

Constat. Unused JS ~1,780 KiB; script the dominant weight (3,389 KiB); lab TBT 2,920 ms (PSI) / 6,360 ms (local); field **INP 388 ms (needs-improvement)**; LCP render-delay 69%.

Risk. Interactivity friction on real users (taps that feel laggy), and no headroom — the metric Google made a ranking signal in March 2026.

Solution (validate against your build). Route/component-level code-splitting and dynamic imports for below-the-fold and non-critical UI; aggressive tree-shaking; ship a modern build target (drop legacy transpilation — ~53 KiB of legacy JS is currently served to modern browsers); break long tasks so the main thread yields. This is iterative and measured after each pass.

Fullstack Senior — **8 man-days**

Expected gain: the largest measured lever — up to ~1,780 KiB / ~3.75 s of unused JS addressable; the primary route to moving INP toward the 200 ms "Good" band.

CT-02 : Govern third-party tags (GTM) & evaluate server-side tagging

Constat. GTM and the tags attributed to its entity transfer 852 KiB and block ~1,279 ms; Meta, mParticle, Google Ads and Analytics add more.

Solution. Audit the active GTM tags, remove or consolidate stale ones, ensure marketing tags are **consent-gated** (OneTrust is already present), and evaluate **server-side tagging** to move measurement off the browser main thread. Recommend validating scope and feasibility against your analytics governance.

Data / Analytics Senior — **5 man-days**

Expected gain: measured reduction in third-party blocking time and transfer; steadier INP under marketing peaks.

CT-03 : Defer the live-chat widget (Intercom) to post-interaction

Constat. Intercom transfers 289 KiB and blocks ~777 ms (bootup ~1.8 s) — loaded eagerly, though most visitors never open chat.

Solution (validate). Replace the eager embed with a lightweight **facade** (a static button) that loads the real widget only on click or after the page is idle/interactive.

Frontend Confirmed — **1.5 man-days**

Expected gain: removes ~289 KiB and ~777 ms of blocking from the initial load for the majority of sessions.

CT-04 : Validate the always-on bot-protection JS (AWS WAF) scope

Observation. The AWS WAF client SDK (`awswaf.com`) is the single heaviest script by bootup (~7.3 s under throttling; 583 KiB / 757 ms blocking on the tested page).

Solution — a security/performance trade-off to validate, not to cut blindly. Confirm with your security team whether the JavaScript challenge needs to run on **every storefront page** or can be

scoped to sensitive flows (login, checkout, account). This is a hypothesis to validate against your threat model — protection must not be weakened; the goal is to right-size where the SDK executes.

DevOps / Security Senior — **2 man-days**

Expected gain: potentially large main-thread relief on browse pages, subject to your security requirements.

Page weight & waterfall

Total transferred weight on the tested page is **6,505 KiB across 183 requests** (Lighthouse `total-byte-weight`). Unusually for retail, **JavaScript — not images — is the dominant post**:

Resource type	Transfer	Requests	Note
Script	**3,389 KiB**	32	#1 weight — see CT-01/CT-02
Image	2,588 KiB	82	WebP already auto-served — see CT-10
Document	151 KiB	2	—
Font	133 KiB	9	self-hosted, preloaded — good
Stylesheet	121 KiB	5	~106 KiB unused — see CT-07
Other	123 KiB	53	—
of which third-party	**2,577 KiB**	83	~40% of the page

> **Honest note on savings.** Lighthouse opportunities overlap (they often target the same bytes), so we cite the single largest rather than summing them: **unused JavaScript ~1,780 KiB** is the headline. We do not project a post-fix score — the gain is re-measured after each optimisation.

CT-07 : Remove unused CSS / ship critical CSS

Constat. ~106 KiB of unused CSS rules (Lighthouse), ~620–750 ms opportunity on the tested page.

Solution (validate). Inline the critical CSS needed for the first view and defer the rest; purge unused rules at build time. On a Next.js app this is usually a build-config / CSS-strategy change to validate against your styling setup.

Frontend Confirmed — **2 man-days**

Expected gain: ~106 KiB less CSS and a measured reduction in render-blocking time.

CT-08 : Modernise JS delivery & caching Backlog

Constat. ~53 KiB legacy/polyfill JS shipped to modern browsers; `uses-long-cache-ttl` flags 19 resources with short cache lifetimes.

Solution (validate). Target modern browsers in the build (drop unnecessary polyfills), and extend cache lifetimes / fingerprinting on static assets so repeat visits re-use them.

DevOps Confirmed — **2 man-days**

Expected gain: smaller modern bundles and better repeat-view caching (measured after change).

Images, fonts & delivery — mostly strengths

> **Recognise what already works.** These are not gaps — they are things your team got right, and they should be **protected** on any future redesign.

Capability	Status	Evidence (observed)
Next-gen images (WebP)	**Auto-served ✓**	a `.png` URL returns `content-type: image/webp` via `vary: Accept`
Self-hosted fonts	**Yes ✓**	Montserrat + Roboto `woff2` on `cdn.example-shop.com`
Font preload	**Yes ✓**	4 `<link rel="preload" as="font">` observed
`font-display: swap`	**Yes ✓**	observed on the tested document — no invisible-text delay
HTTP/3 (QUIC)	**Available ✓**	`alt-svc: h3` advertised (Cloudflare)
Layout stability (CLS)	**FAST ✓**	0.09 field / ~0.05 lab
LCP hero priority	**Missing**	no image preload / no `fetchpriority` — see CT-05

CT-10 : Extend the image strategy to AVIF + right-sizing Backlog

Constat. WebP is already auto-negotiated — a real strength. Images still total 2,588 KiB over 82 requests, so there is room to go one format further.

Solution (validate against your Next.js / CDN pipeline). Enable **AVIF** as a negotiated format (Next.js image formats or the CDN), and audit `srcset`/`sizes` so no image is delivered larger than its rendered container.

Benchmark (attributed): AVIF is typically ~50% smaller than JPEG and 20–30% smaller than WebP at comparable quality, with ~94.9% browser support in 2026 (2026 web-performance surveys).

Frontend Confirmed — **2 man-days**

Expected gain: lighter images on top of an already-good baseline; measured KiB reduction per image.

Transport, redirects & back/forward cache

Transport (observations)

Delivery is modern: **Cloudflare** edge, **HTTP/2** negotiated with **HTTP/3** advertised, and compression active on the tested document (~13.5 KB on the wire vs ~46.5 KB decompressed). The homepage

document is served `cache-control: private, no-store` with `cf-cache-status: DYNAMIC` — normal for a personalised storefront document (static assets are cached on the CDN). This is an *observation* about the tested response, not a claim that caching is broken.

CT-06 : Collapse the entry redirect chain

Observation. Lighthouse attributes **~1.7–1.9 s** to redirects on the tested mobile navigation; the apex `example-shop.com` routes through at least one redirect hop into the storefront.

Risk. First-touch visits — shared links, ad landings, typed apex — pay a delay that returning users on the canonical URL avoid (which is part of why field LCP stays FAST).

Solution (validate the topology). Map the real redirect path (apex → `www` → market/locale) and minimise it to a single hop; make sure ad and campaign URLs point straight at the canonical localized URL.

DevOps Confirmed — **1.5 man-days**

Expected gain: up to the ~1.7–1.9 s measured redirect cost removed from first-touch navigations.

CT-09 : Restore back/forward cache (bfcache) eligibility

Constat. Lighthouse reports **1 blocking reason** for bfcache; the document's `cache-control: no-store` is a common cause.

Solution (validate). Identify the blocker (a `no-store` document header and/or an `unload` listener) and remove it where safe, so Back/Forward navigations restore instantly from memory.

Why it matters (2026 standard): bfcache turns Back/Forward into near-instant, zero-layout-shift restorations — a high-leverage, low-code interactivity win.

Frontend Confirmed — **1.5 man-days**

Expected gain: near-instant back/forward navigation for repeat browsing journeys (measured via the bfcache audit after change).

> **Out of this audit's lane (one line):** platform security headers are present on the tested response (`strict-transport-security`, a `content-security-policy` with `frame-ancestors 'none'`) — noted only as a passing signal; a security posture review is a separate exercise.

Competitive benchmark — where you stand

We measured competitors with the **same tool as you** (PageSpeed Insights, Google's infrastructure) so the lab comparison is homogeneous, and we read each side's **CrUX field data** too. Direct rivals identified: Alphalete, AYBL, NVGTN; aspirational references: Lululemon, Alo Yoga, Vuori, Nike (WebSearch, 2026).

Layer 1 — your measures vs the living market standard

Google's current "Good" bands (web.dev, 2026) are LCP ≤ 2.5 s, INP under 200 ms, CLS below 0.1. Your lab LCP (21.2 s) and TBT (2,920 ms) are **classed poor (Lighthouse)** — an order of magnitude off the standard — and your field INP (388 ms) is **needs-improvement**. Your field LCP and CLS already meet the bar. The gap the market cares about, and that your buyers feel, is **interactivity**.

Layer 3 — measured comparison (PSI lab + CrUX field, both sides)

Measured axis	Example Shop (you)	Lululemon	Alo Yoga	Alphalete	Source
PSI lab performance	27	31	15	Not measured	PSI lab (Google)
LCP lab (rating)	21.2 s (poor)	18.7 s	75.0 s	Not measured	PSI lab
Field LCP (real users)	FAST	FAST	FAST	—	CrUX
Field INP (real users)	AVERAGE (388 ms)	AVERAGE	AVERAGE	—	CrUX
Field CLS (real users)	FAST	FAST	FAST	—	CrUX

Benchmark on **2 measured peers**: Alphalete could not be measured — PSI returned *NO_FCP* (its page painted no content for the tool). That is a limit of automated external tooling, not evidence about its speed.

Layer 2 — observable delivery capabilities (immune to measurement blocks)

Capability the market expects	You	Segment	Buyer stake
Field LCP in "FAST"	**Yes (1.7 s)**	Yes	fast first paint for real users
Next-gen images (WebP)	**Yes (auto)**	Varies	lighter pages
HTTP/3 delivery	**Yes**	Varies	less connection latency
Stable layout (CLS good)	**Yes**	Yes	no janky shifts
Field INP in "Good"	**No (388 ms)**	No	snappy taps & add-to-cart
Prioritised LCP hero	**No**	—	earlier hero on cold loads

Layer 4 — the honest positioning. The entire premium-activewear DTC segment carries heavy media and JavaScript: all measured peers score *poor* in the synthetic lab (Alo Yoga 15, you 27, Lululemon 31) while every one delivers a FAST field LCP. So this is not "you are uniquely slow." It is that **the whole pack is stuck in *needs-improvement* on interactivity (INP)**, and your lab fragility is real headroom you will spend as you add features. The strategic reading: you already **lead on delivery fundamentals** (WebP, HTTP/3, CLS, fonts) — the **unclaimed advantage** is to become the segment brand whose taps feel instant, by moving INP into "Good" before your rivals do. That is a reason for a shopper to prefer you, and it is available now.

What the market expects in 2026 (dated standards)

Standard (2026)	Where you stand	Source
CWV thresholds: LCP ≤ 2.5 s · INP under 200 ms · CLS below 0.1 (75th pct)	Field LCP/CLS meet it; INP does not	web.dev / Google
INP is the interactivity metric & a ranking signal (since March 2026)	Field INP 388 ms — needs-improvement	web.dev / Google Search
Third-party app/script JS = #1 cause of failing CWV	2,577 KiB / ~3.2 s of third-party — matches	Shopify CWV guidance 2026
`fetchpriority="high"` on the LCP image	Not applied — see CT-05	Google (fetchpriority test)
AVIF image format (≈94.9% support)	WebP yes, AVIF next — see CT-10	2026 web-performance surveys
bfcache for instant back/forward	1 blocker — see CT-09	web.dev
Speculation Rules API (prerender)	Prospective — see CT-11	web.dev / Chromium docs

CT-11 (prospective) : Speculation Rules for near-instant navigation

Idea. The Speculation Rules API lets the browser prerender the most likely next page (e.g. category → product), so navigation feels instant. Production-ready in Chromium in 2026.

Caveat. This is a *prospective* item — its hierarchy is a strategic judgement, not a measurement, so it sits behind the measured-impact work above, never in the top-3.

Frontend Confirmed **— 1 man-day**

Expected gain: perceived-instant next-page loads on predicted journeys (measure engagement after a controlled rollout).

Timing & the business case

We are 18 July 2026 — the summer active season. Looking at the calendar ahead for activewear in your US/UK/global markets, two windows dominate and both are **upcoming**:

Black Friday / Cyber Monday (Fri 27 – Mon 30 November 2026) is the peak apparel-DTC trading window — roughly **four months out**. Then the biggest activewear moment of the year: the **January 2027 "New Year, new me" resolution surge**, ~five to six months out, when acquisition traffic spikes hardest. These are exactly the moments when a JavaScript-heavy page is most exposed — more concurrent visitors, more devices, more marketing tags firing.

> **Why now.** The quick wins (CT-05 hero priority, CT-06 redirects, CT-03 chat defer, CT-09 bfcache) can land before back-to-gym; the structural JavaScript programme (CT-01, CT-02, CT-04) can be in place before BFCM — so your storefront is at its snappiest exactly when the traffic and the stakes are highest.

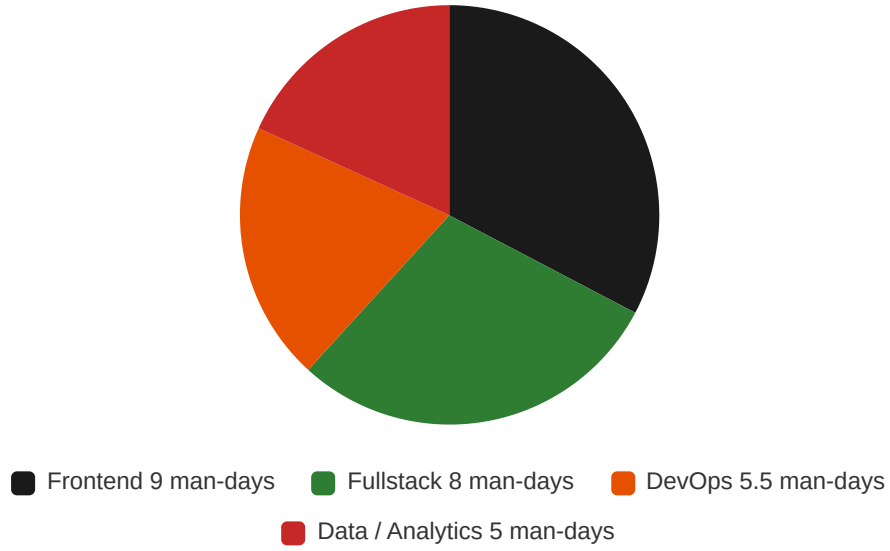
How to value the gain — honestly

We do not invent a revenue percentage — we don't have your analytics. The gain is expressed three legitimate ways:

- **Measured savings (Lighthouse):** e.g. ~1,780 KiB / ~3.75 s of unused JavaScript, ~1.7–1.9 s of redirect cost, ~106 KiB of unused CSS — all real, all re-measured after the fix.
- **Attributed benchmark:** Google/Deloitte's *"Milliseconds Make Millions"* (2020) associates faster mobile load with higher retail conversion — a sector benchmark, not a measurement of your site.
- **Client-variable formula:** monthly gain = traffic × Δ conversion rate × average order value. Your Analytics makes this concrete in a 30-minute scoping workshop.

Effort distribution by profile

The full programme is **27.5 man-days identified** (complete backlog), of which **~20.5 man-days** form the prioritised short/mid-term roadmap. Split by profile:



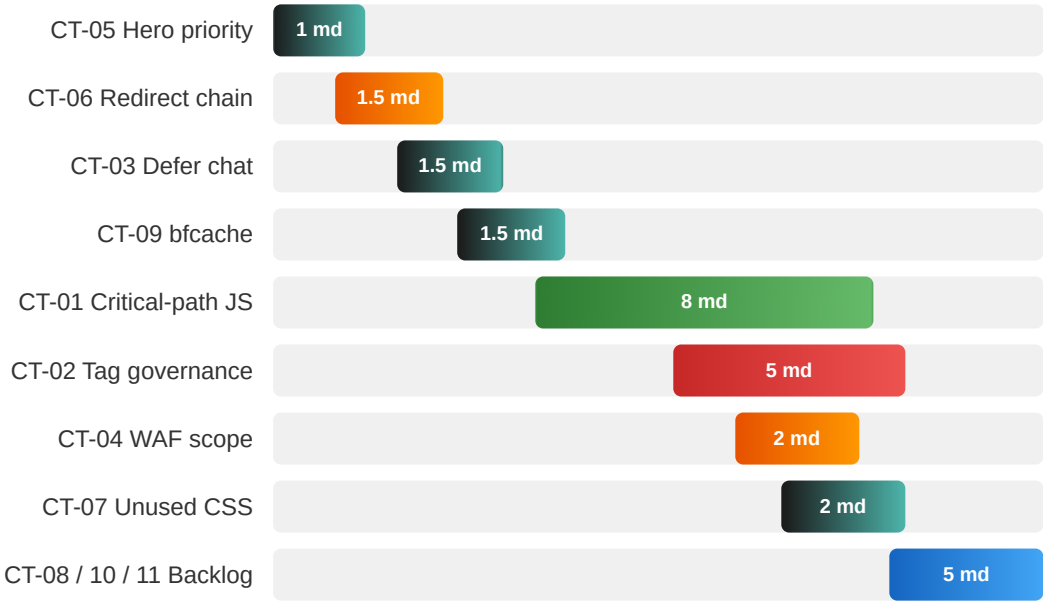
> Two-thirds of the effort is **frontend / fullstack JavaScript work** — consistent with the diagnosis: on a headless Next.js storefront, disciplined JavaScript is the lever that moves both interactivity (INP) and the fragile LCP.

Prioritised roadmap

#	Action	Profile	Seniority	Effort	Phase
CT-01	Cut critical-path JavaScript execution (INP + LCP render)	Fullstack	Senior	8 md	Mid
CT-02	Govern third-party tags (GTM) + server-side tagging	Data/Analytics	Senior	5 md	Mid
CT-03	Defer live-chat (Intercom) to post-interaction	Frontend	Confirmed	1.5 md	Short
CT-04	Validate always-on bot-protection JS (AWS WAF) scope	DevOps/Security	Senior	2 md	Mid
CT-05	Prioritise the LCP hero (fetchpriority + preload)	Frontend	Confirmed	1 md	Short
CT-06	Collapse the entry redirect chain	DevOps	Confirmed	1.5 md	Short

CT-07	Remove unused CSS / ship critical CSS	Frontend	Confirmed	2 md	Mid
CT-09	Restore bfcache eligibility	Frontend	Confirmed	1.5 md	Short
Backlog (valuable, scheduled after the priority set)					
CT-08	Modernise JS delivery & caching (legacy JS, cache TTL)	DevOps	Confirmed	2 md	Long
CT-10	Extend images to AVIF + right-sizing	Frontend	Confirmed	2 md	Long
CT-11	Speculation Rules API (prospective)	Frontend	Confirmed	1 md	Long
		Identified backlog / Prioritised roadmap		**27.5 md / ~20.5 md**	

Sequencing



Left→right ≈ relative sequence, not fixed dates. Quick wins first (before back-to-gym), the JavaScript programme in place before BFCM.

Closing. Example Shop's storefront is a well-built headless stack with genuine strengths — stable layout, WebP, HTTP/3, clean fonts, and a real-user LCP that is already fast. The work ahead is focused, not a rebuild: **discipline the JavaScript**. Do that, and you move the one metric your buyers feel (INP) out of *needs-improvement*, remove the fragility behind the *poor* lab scores, and walk into BFCM and the January surge with headroom to spare.