Risk & Liability Brief — The 2026 Client-Side Cliff

Audience: owners, finance, legal, and engineering leads who carry the downside. **As of:** 2026-06-21 · **Companion to:** `FORWARD-DEPLOY-AGENTIC-GRAPHQL.md` (the fix).

> **One line:** Shopify is removing the client-side / Script-Editor era on **published dates**. Storefronts that keep business logic in **Liquid + browser JavaScript** face **silent checkout failure, uptime exposure, and compliance liability** — and the largest deadline is **2026-06-30**.

---

1. Site-down vulnerability — what goes dark, and when

Date	Event	Failure mode if unmigrated
2026-06-30	**Shopify Scripts removed**	Payment / shipping / line-item logic **silently stops**. Checkout still loads — it just stops applying your rules: wrong methods shown, discounts not applied, surcharges dropped. **No error, no alert — just wrong orders and lost margin.**
Rolling, quarterly	**API versions sunset (~12 mo)**	Calls on an expired version start returning errors. Anything pinned to an old version **breaks without a code change on your side.**
2026-01-01 *(passed)*	**No new legacy custom apps**	New integrations can't use the old install path; bolt-ons stall.

Why client-side JS + Liquid is inherently fragile at checkout:

- Business logic runs on the **customer's device and network** — ad-blockers, slow mobile, script errors, and third-party outages all execute *at the moment of payment*, the worst possible place.
- Logic is **scattered across theme files and inline scripts** with no single contract — a small theme edit can silently disable a pricing or eligibility rule.
- There is **no server-side source of truth and no failover** — when a client script or a REST dependency fails, the order is simply wrong, and you find out from a customer or a chargeback.

Net: the failure is not a loud "site down" page — it is a **quietly broken checkout** that keeps taking orders incorrectly. That is harder to detect and more expensive than an outright outage.

---

2. Liability exposure

Domain	Exposure if logic stays client-side / on deprecated APIs
Privacy / consent	Consent enforced only in the browser (cookies/JS) is **bypassable and unauditable**. Consent Mode v2 / CPRA / GDPR expect a **server-side, logged** consent signal. Gaps invite regulator and class-action risk.
Pricing transparency	**EU Omnibus** requires a verifiable 30-day prior-lowest reference price. Client-side price display has **no durable record** to prove compliance.
Security	Keys/tokens used client-side, or pasted into theme/app config, are **exposable and hard to rotate**. Deprecated APIs stop receiving **security fixes**. Both expand breach liability.
Auditability	No server-side event log = **no defensible record** of what the store charged, showed, or consented to at time of sale.
Accessibility / contract	Fragile client logic that misprices or misrepresents at checkout can breach merchant terms and consumer-protection rules.

The common root: **decisions are made where you can neither control nor prove them** — the browser.

---

3. Risk matrix

Risk	Likelihood	Impact	Trigger date
Checkout rules silently stop (Scripts)	**High** (automatic)	**Severe** (revenue/margin)	**2026-06-30**
Deprecated API version errors	**Medium**	High (feature outages)	quarterly
Consent/privacy non-compliance	Medium	**Severe** (fines/litigation)	ongoing
Secret/key exposure or breach	Low–Med	**Severe** (breach cost)	ongoing
Inability to prove pricing/consent	Medium	High (regulatory)	ongoing

Severity concentrates on a **known, dated, unavoidable** event: **2026-06-30**.

---

4. The window is closing

The dates are **published and fixed** — this is not a "maybe." Every week of delay shortens the runway to migrate **payment, shipping, and discount logic to server-side Functions**, stand up a **server-side source of truth (GraphQL)**, and put **consent, pricing, and secrets where they can be controlled and proven.**

Do-nothing is a decision to accept a quietly broken checkout and an undefendable compliance posture on a date you already know.

---

5. The mitigation (one move, many risks retired)

Forward-deploy to **server-side GraphQL + Shopify Functions behind an agentic Tool Runner** — the browser renders, the **server decides, logs, and proves**. This single architectural move retires the checkout-failure, uptime, consent, pricing, security, and auditability risks above at once.

→ See `FORWARD-DEPLOY-AGENTIC-GRAPHQL.md` for the migration plan, the nine pillars, and the per-deadline schedule.

> **Sources (Shopify-published):** Scripts → Functions transition (removal **2026-06-30**); REST Admin API legacy (2024-10) / GraphQL-only for new public apps (2025-04); legacy custom apps end (2026-01-01); API version sunset cadence (~12 months, quarterly).