7-Step Website Leak Fix

Konkrete Schritte, um den Umsatzverlust durch langsame Ladezeiten zu stoppen. Sofort umsetzbar.

Format: 7 Schritte mit Checklisten	**Zeitaufwand:** 2-4 Stunden
Schwierigkeit: Anfänger bis Fortgeschrittene	**Wirkung:** 20-60% schnellere Ladezeit

Erstellt von **Web Skyline** | web-skyline.com

Basierend auf Google PageSpeed Insights Best Practices und Core Web Vitals.

Bevor Sie anfangen

Dieser Guide ist kein Marketing-Material. Es sind die gleichen Schritte, die wir bei jedem Kundenprojekt als Erstes durchgehen. Jeder Schritt hat eine Checkliste am Ende — arbeiten Sie diese von oben nach unten ab. Die Reihenfolge ist bewusst gewählt: Schritt 1 bringt den größten Effekt mit dem geringsten Aufwand, Schritt 7 ist für Fortgeschrittene.

Was Sie brauchen: Zugang zu Ihrem Hosting-Panel oder FTP, grundlegende Kenntnisse im Umgang mit Ihrem CMS (Shopify, WooCommerce, Shopware etc.). Für Schritte 5-7 ist ein Entwickler hilfreich, aber nicht zwingend nötig.

Wichtig: Machen Sie vor jeder Änderung ein Backup. Testen Sie jede Änderung einzeln und messen Sie den Effekt mit Google PageSpeed Insights (pagespeed.web.dev) bevor Sie den nächsten Schritt angehen.

01

Bilder optimieren

Der mit Abstand größte Hebel. 60-80% aller Shops laden Bilder, die 5-10x größer sind als nötig.

Bilder sind fast immer der Hauptgrund für langsame Ladezeiten. Ein typischer Online-Shop hat Produktbilder mit 2-5 MB pro Stück, obwohl 80-150 KB völlig ausreichen. Das Problem: Die meisten Shop-Betreiber laden Bilder direkt von der Kamera oder vom Hersteller hoch — ohne Komprimierung, ohne Größenanpassung, im falschen Format.

Was genau zu tun ist:

Format umstellen auf WebP. JPEG und PNG sind veraltet. WebP liefert die gleiche Qualität bei 25-35% kleinerer Dateigröße. Jeder moderne Browser unterstützt WebP. Für Shopify: nutzen Sie die eingebaute automatische Konvertierung (Settings → Files). Für WooCommerce: installieren Sie das Plugin 'ShortPixel' oder 'Imagify' — beide konvertieren automatisch. Für Shopware: das Plugin 'FroshPlatformThumbnailProcessor' macht das automatisch.

Maximale Breite auf 1200px begrenzen. Kein Produktbild braucht mehr als 1200px Breite. Die meisten Produktkarten auf der Kategorieseite zeigen Bilder bei 300-400px an. Laden Sie keine 4000px-Bilder hoch und lassen den Browser skalieren — das verschwendet Bandbreite. Nutzen Sie 'srcset' im HTML, damit der Browser die passende Größe lädt.

Lazy Loading aktivieren. Bilder die nicht im sichtbaren Bereich sind, sollen erst laden wenn der Nutzer dorthin scrollt. In HTML: fügen Sie 'loading="lazy"' zu jedem img-Tag hinzu (außer dem Hero-Bild oben — das soll sofort laden). Die meisten CMS haben dafür eine Einstellung.

Komprimierung auf Qualität 75-80. Der Unterschied zwischen Qualität 80 und 100 ist mit bloßem Auge nicht erkennbar, aber die Dateigröße halbiert sich. Nutzen Sie squoosh.app für einzelne Bilder oder ShortPixel für automatische Batch-Komprimierung.

Profi-Tipp: Prüfen Sie Ihre 10 meistbesuchten Seiten in Google Analytics. Optimieren Sie die Bilder auf genau diesen Seiten zuerst — das gibt den größten sofortigen Effekt.

Checkliste Schritt 1:

- ■ Alle Produktbilder auf WebP konvertiert
- ■ Maximale Bildbreite auf 1200px begrenzt
- ■ Lazy Loading für alle Bilder aktiviert (außer Hero)
- ■ Komprimierung auf Qualität 75-80 eingestellt
- ■ PageSpeed-Score vorher/nachher gemessen und notiert

02

Server und Hosting prüfen

Wenn der Server langsam antwortet, hilft die beste Optimierung nichts. TTFB unter 200ms ist das Ziel.

Time to First Byte (TTFB) misst, wie schnell Ihr Server auf eine Anfrage reagiert. Wenn TTFB über 600ms liegt, ist Ihr Hosting das Problem — nicht Ihr Code. Bei Shared Hosting (1&1, Strato Basis-Pakete) ist TTFB oft 800ms+. Das allein kann 2-3 Punkte PageSpeed kosten.

Konkretes Vorgehen:

TTFB messen: Öffnen Sie pagespeed.web.dev, testen Sie Ihre Seite. Im Ergebnis finden Sie unter 'Diagnostics' den Wert 'Initial server response time was...' — das ist Ihr TTFB. Unter 200ms ist gut, 200-600ms ist akzeptabel, über 600ms ist ein Problem.

Hosting-Empfehlungen nach Shop-System: Shopify — Hosting ist inklusive und schnell, hier liegt das Problem selten am Server. WooCommerce — wechseln Sie von Shared auf Managed WordPress Hosting (Raidboxes, Cloudways, Kinsta). Preis: ca. 20-30 EUR/Monat statt 5 EUR, aber der Geschwindigkeitsgewinn ist dramatisch. Shopware — ein VPS mit mindestens 4GB RAM und SSD. Hetzner Cloud CX21 (ca. 6 EUR/Monat) reicht für kleine Shops.

CDN aktivieren: Ein Content Delivery Network liefert Ihre statischen Dateien (Bilder, CSS, JS) von Servern aus, die geografisch nah am Besucher sind. Cloudflare Free reicht für den Anfang — 5 Minuten Einrichtung, 0 EUR. DNS umstellen, fertig. Effekt: 50-200ms weniger Ladezeit.

PHP-Version aktualisieren (für WooCommerce/Shopware): PHP 8.1+ ist 20-30% schneller als PHP 7.4. Prüfen Sie im Hosting-Panel welche Version läuft und aktualisieren Sie. Testen Sie danach ob der Shop noch funktioniert — in seltenen Fällen gibt es Inkompatibilitäten mit alten Plugins.

Checkliste Schritt 2:

- ■ TTFB gemessen und notiert: _____ ms
- ■ Wenn TTFB > 600ms: Hosting-Upgrade geplant
- ■ CDN (Cloudflare) aktiviert
- ■ PHP-Version geprüft und ggf. aktualisiert

03

Caching richtig einrichten

Wiederkehrende Besucher laden Ihre Seite bis zu 80% schneller — wenn Caching aktiviert ist.

Browser-Caching bedeutet: Dateien, die sich nicht oft ändern (Bilder, CSS, Schriftarten), werden lokal beim Besucher gespeichert. Beim zweiten Besuch müssen sie nicht erneut heruntergeladen werden. Server-Caching bedeutet: Statt jede Seite bei jedem Aufruf neu zu generieren, wird eine fertige Version zwischengespeichert.

Umsetzung:

Browser-Caching via .htaccess (Apache): Fügen Sie in Ihre .htaccess-Datei im Root-Verzeichnis folgende Regeln ein. Bilder werden 1 Jahr gecacht, CSS/JS 1 Monat. Wenn Sie Cloudflare nutzen, wird das automatisch übernommen.

```
<IfModule mod_expires.c>
  ExpiresActive On
  ExpiresByType image/webp "access plus 1 year"
  ExpiresByType image/jpeg "access plus 1 year"
  ExpiresByType text/css "access plus 1 month"
  ExpiresByType application/javascript "access plus 1 month"
</IfModule>
```

Server-seitiges Caching: WooCommerce — installieren Sie 'WP Super Cache' oder 'LiteSpeed Cache' (wenn Ihr Server LiteSpeed nutzt). Aktivieren, Standardeinstellungen belassen, fertig. Shopware — eingebautes HTTP-Cache aktivieren unter Settings → System → Caches. Shopify — kein Handlungsbedarf, Shopify cacht automatisch.

GZIP/Brotli-Komprimierung: Textbasierte Dateien (HTML, CSS, JS) werden vor dem Senden komprimiert. Reduziert die Übertragungsgröße um 60-70%. Bei Cloudflare automatisch aktiv. Ohne Cloudflare: in .htaccess 'mod_deflate' aktivieren.

Checkliste Schritt 3:

- ■ Browser-Caching-Regeln in .htaccess / CDN aktiv
- ■ Server-Caching-Plugin installiert und aktiv
- ■ GZIP oder Brotli-Komprimierung aktiv
- ■ PageSpeed-Warnung 'Serve static assets with cache policy' verschwunden

04

Schriften richtig laden

Google Fonts verursachen bis zu 500ms Extra-Ladezeit wenn sie falsch eingebunden sind.

Jede externe Schriftart, die von fonts.googleapis.com geladen wird, braucht mindestens 2 DNS-Lookups, 2 HTTP-Requests und blockiert das Rendering. Wenn Sie 3 Schriftarten mit je 2 Varianten laden (regular + bold), sind das 6 Dateien und 300-600ms zusätzliche Ladezeit.

Konkrete Lösung:

Schriften lokal hosten statt von Google laden. Gehen Sie auf google-webfonts-helper.herokuapp.com, suchen Sie Ihre Schriftart, laden Sie die WOFF2-Dateien herunter, legen Sie sie in Ihren /fonts/-Ordner und binden Sie sie per @font-face in Ihrem CSS ein. Der Vorteil: kein externer DNS-Lookup, kein Datenschutz-Problem (DSGVO!), und die Dateien werden über Ihr CDN gecacht.

Nur die Varianten laden die Sie brauchen. Laden Sie nicht 'Montserrat 100-900' wenn Sie nur 400 und 700 verwenden. Jede unnötige Variante ist 15-30 KB die der Browser herunterladen muss.

font-display: swap im @font-face setzen. Das bedeutet: Der Browser zeigt sofort Text in einer Systemschrift an und tauscht sie aus sobald die eigentliche Schrift geladen ist. Ohne 'swap' sieht der Nutzer weißen Text bis die Schrift da ist — das kostet LCP-Punkte.

DSGVO-Hinweis: Google Fonts direkt von Google zu laden ist in Deutschland nach aktueller Rechtsprechung (LG München, Jan 2022) ohne Einwilligung nicht zulässig. Lokales Hosting löst dieses Problem und macht Ihren Shop gleichzeitig schneller.

Checkliste Schritt 4:

- ■ Schriften lokal gehostet (keine externen Google-Requests)
- ■ Nur benötigte Varianten geladen
- ■ font-display: swap gesetzt
- ■ DSGVO-konform (kein Laden von Google-Servern)

05

Ungenutztes CSS und JavaScript entfernen

Die meisten Shops laden 200-500 KB CSS/JS die auf der aktuellen Seite gar nicht gebraucht werden.

Jedes Mal wenn ein Browser ein CSS- oder JavaScript-File lädt, muss er es parsen und ausführen bevor die Seite fertig gerendert wird. Render-blockierende Ressourcen sind der häufigste Grund für schlechte LCP-Werte. Das Problem bei Shop-Systemen: Plugins und Themes laden oft globale CSS/JS-Dateien auf jeder Seite — auch wenn sie nur auf einer einzigen Seite gebraucht werden.

Vorgehen:

Ungenutztes CSS finden: Öffnen Sie Chrome DevTools (F12), gehen Sie zu 'Coverage' (Ctrl+Shift+P → 'Coverage'), laden Sie die Seite neu. Sie sehen rot markiert welcher Anteil des CSS ungenutzt ist. Bei den meisten Shops sind das 70-90%.

Nicht benötigte Plugins deaktivieren. Jedes WooCommerce/Shopware-Plugin lädt eigenes CSS/JS. Listen Sie alle aktiven Plugins auf und fragen Sie sich: 'Brauche ich das wirklich?' Slider-Plugins (Revolution Slider etc.) sind besonders schlimm — sie laden oft 300 KB+ JavaScript. Ersetzen Sie Slider durch ein statisches Hero-Bild: bessere Conversion UND bessere Ladezeit.

JavaScript defer/async setzen. Skripte die nicht sofort gebraucht werden, sollen das Rendering nicht blockieren. Fügen Sie 'defer' zu Script-Tags hinzu die nicht kritisch für den First Paint sind. Bei WordPress: das Plugin 'Autoptimize' oder 'Flying Scripts' macht das automatisch.

Critical CSS inline einbinden. Das CSS das für den sichtbaren Bereich (above the fold) nötig ist, wird direkt im HTML eingebettet. Der Rest wird asynchron nachgeladen. Tools: 'Critical' von Addy Osmani (npm-Paket), oder in WooCommerce das Plugin 'Autoptimize' mit der Option 'Inline and Defer CSS'.

Checkliste Schritt 5:

- ■ Coverage-Report in DevTools erstellt
- ■ Nicht benötigte Plugins deaktiviert (insb. Slider)
- ■ Script-Tags mit defer/async versehen
- ■ Critical CSS inlined oder Plugin aktiviert

06

Drittanbieter-Skripte kontrollieren

Facebook Pixel, Google Tag Manager, Chat-Widgets, Hotjar — jedes Skript kostet 100-500ms.

Third-Party-Skripte sind der versteckte Performance-Killer. Jedes externe Skript macht einen DNS-Lookup, einen TLS-Handshake, lädt JavaScript herunter und führt es aus. Ein typischer Shop hat 5-15 externe Skripte: Analytics, Facebook Pixel, Google Ads Conversion, Chat-Widget, Cookie-Banner, Hotjar, Trustpilot-Widget, Newsletter-Popup. Zusammen kosten die oft 1-3 Sekunden.

Was konkret zu tun ist:

Inventur machen. Öffnen Sie Chrome DevTools → Network → filtern Sie nach '3rd-party'. Listen Sie jedes externe Skript auf. Fragen Sie sich bei jedem: 'Bringt mir das messbar Geld?' Wenn nein → entfernen. Hotjar auf einem Shop mit 100 Besuchern/Monat? Raus. Newsletter-Popup das niemand nutzt? Raus. Social Media Share Buttons? Raus.

Google Tag Manager nutzen statt einzelne Skripte. Laden Sie ein einziges GTM-Script und steuern Sie alle Tags darüber. Vorteil: ein statt zehn separate Requests. Plus: Sie können Tags verzögert laden (Trigger: 'Window Loaded' oder '5 Seconds Timer') sodass sie den First Paint nicht blockieren.

Cookie-Banner optimieren. Cookie-Banner-Skripte (Cookiebot, Usercentrics) blockieren oft das gesamte Rendering weil sie vor allem anderen laden müssen. Wechseln Sie zu einem leichtgewichtigen Consent-Tool oder laden Sie das Banner asynchron nach dem DOM-Ready.

Chat-Widget lazy loaden. Kein Besucher braucht das Chat-Widget in den ersten 5 Sekunden. Laden Sie es erst nach 5 Sekunden oder wenn der Nutzer bis zum Footer scrollt. In den meisten Chat-Tools (Tidio, LiveChat, Intercom) gibt es dafür eine Einstellung oder Sie wrappen den Script-Tag in ein setTimeout.

Checkliste Schritt 6:

- ■ Alle Third-Party-Skripte aufgelistet
- ■ Nicht benötigte Skripte entfernt
- ■ Verbleibende Skripte über GTM mit Verzögerung geladen
- ■ Cookie-Banner-Performance geprüft
- ■ Chat-Widget verzögert geladen

07

Core Web Vitals gezielt verbessern

Feintuning für die drei Metriken die Google direkt im Ranking berücksichtigt.

LCP (Largest Contentful Paint) optimieren:

LCP misst wann das größte Element im sichtbaren Bereich fertig geladen ist — meistens ein Hero-Bild oder eine große Überschrift. Ziel: unter 2,5 Sekunden. Konkrete Maßnahmen: (1) Das Hero-Bild mit 'fetchpriority="high"' und 'loading="eager"' versehen. (2) Preload-Link im Head setzen: '<link rel="preload" as="image" href="hero.webp">'. (3) Kein Lazy Loading für das LCP-Element. (4) Wenn das LCP-Element ein Text-Block ist: Schriftart preloaden damit der Text sofort sichtbar wird.

CLS (Cumulative Layout Shift) optimieren:

CLS misst ob Elemente auf der Seite 'springen' während sie lädt. Typisches Problem: Bilder ohne width/height-Attribute, Werbebanner die nach 2 Sekunden erscheinen, Cookie-Banner die den Content nach unten schieben. Lösung: (1) Jedem img-Tag explizite width und height geben (das reserviert den Platz). (2) Für dynamische Inhalte (Ads, Banner) feste Höhe reservieren per CSS 'min-height'. (3) Schriften mit font-display: swap laden (Schritt 4).

INP (Interaction to Next Paint) optimieren:

INP misst wie schnell die Seite auf Klicks und Tippen reagiert. Ziel: unter 200ms. Typisches Problem: schweres JavaScript blockiert den Main Thread. Lösung: (1) Große JavaScript-Bundles aufteilen (Code Splitting). (2) Schwere Berechnungen in einen Web Worker auslagern. (3) Event-Handler entprellen (debounce Scroll- und Resize-Handler). (4) Unnötige Re-Renders in JavaScript-Frameworks vermeiden.

Profi-Tipp: Testen Sie nicht nur im Lab (PageSpeed Insights), sondern auch mit echten Nutzerdaten. In Google Search Console unter 'Core Web Vitals' sehen Sie die Werte Ihrer echten Besucher. Lab-Daten und Felddaten können stark abweichen.

Checkliste Schritt 7:

- ■ LCP-Element identifiziert und mit preload/fetchpriority versehen
- ■ Alle Bilder haben explizite width/height
- ■ CLS-Wert unter 0.1
- ■ INP-Wert unter 200ms
- ■ Ergebnisse in Search Console verifiziert

Was kommt nach den 7 Schritten?

Wenn Sie alle 7 Schritte umgesetzt haben, sollte Ihr PageSpeed-Score um 20-40 Punkte gestiegen sein und Ihre Ladezeit um 1-3 Sekunden schneller. Das bedeutet konkret: weniger Absprünge, höhere Conversion-Rate und mehr Umsatz.

Manche dieser Schritte sind technisch anspruchsvoll — besonders Schritt 5, 6 und 7. Das ist normal. Nicht jeder Shop-Betreiber muss (oder sollte) das selbst machen.

Option A: Sie setzen alles selbst um

Perfekt. Nutzen Sie die Checklisten, messen Sie nach jedem Schritt den PageSpeed-Score, und arbeiten Sie sich durch. Bei Fragen können Sie uns jederzeit schreiben — eine kurze Einschätzung per E-Mail ist kostenlos.

Option B: Wir optimieren Ihren Shop für Sie

Wir führen einen vollständigen Performance-Audit durch, setzen alle 7 Schritte um und messen den Erfolg. Pauschalpreis, keine versteckten Kosten, Ergebnis in 5-7 Werktagen.

Was enthalten ist:	
Vollständiger PageSpeed-Audit mit Bericht	✓
Bildoptimierung aller Produkt- und Kategorieseiten	✓
Server/Caching/CDN-Konfiguration	✓
Schriften lokal hosten (DSGVO-konform)	✓
CSS/JS-Optimierung und Third-Party-Cleanup	✓
Core Web Vitals Feintuning (LCP, CLS, INP)	✓
Vorher/Nachher-Bericht mit Messwerten	✓

So erreichen Sie uns:

Kostenlose Erstanalyse anfragen

Schreiben Sie uns — wir schauen uns Ihren Shop an und sagen Ihnen ehrlich, ob sich eine Optimierung lohnt.

Website: web-skyline.com

E-Mail: webskyline12@gmail.com

Antwort innerhalb von 24 Stunden. Keine Verpflichtung.