Technical SEO

Master Guide

For Developers & Advanced Practitioners

- Core Web Vitals
- LCP · CLS · INP
- Schema & Structured Data
- Crawlability & Indexing
- Site Architecture & Performance

Table of Contents

1. Core Web Vitals

- 1.1 Largest Contentful Paint (LCP)
- 1.2 Cumulative Layout Shift (CLS)
- 1.3 Interaction to Next Paint (INP)
- 1.4 Measurement & Tooling

2. Structured Data & Schema Markup

- 2.1 JSON-LD vs Microdata vs RDFa
- 2.2 Key Schema Types
- 2.3 Rich Results & SERP Features
- 2.4 Validation & Testing

3. Crawlability & Indexing

- 3.1 Robots.txt
- 3.2 XML Sitemaps
- 3.3 Canonical Tags
- 3.4 Crawl Budget Optimisation
- 3.5 Indexing Directives

4. Site Architecture & Performance

- 4.1 URL Structure
- 4.2 Internal Linking
- 4.3 Page Speed & Core Infrastructure
- 4.4 CDN & Caching Strategy
- 4.5 JavaScript SEO

Introduction

Technical SEO is the foundation on which all other optimisation efforts rest. Without a technically sound site, even the most compelling content and the strongest backlink profile will fail to reach their potential in organic search. This guide is written for developers and senior SEO practitioners who want implementation-level depth — not just theory.

Google's ranking systems evaluate hundreds of signals, but three pillars consistently separate high-performing sites from the rest: **page experience** (measured by Core Web Vitals), **content clarity** (communicated through structured data), and **crawl efficiency** (enabled by a clean architecture). This guide covers all three in detail, with real code, real thresholds, and real trade-offs.

> This guide targets practitioners already familiar with HTTP, HTML, and basic SEO principles. Code examples are in JavaScript (browser), Python (tooling), and shell commands where applicable.

1. Core Web Vitals

Core Web Vitals (CWV) are a set of real-world, user-centric metrics that Google uses as a page-experience ranking signal. They measure three distinct dimensions of the loading and interaction experience. Since the May 2021 Page Experience update, CWV have been an explicit ranking factor in both mobile and desktop search. In March 2024, **Interaction to Next Paint (INP)** permanently replaced First Input Delay (FID).

Metric	Good	Needs Work
LCP – Loading	≤ 2.5 s	> 4.0 s
CLS – Stability	≤ 0.1	> 0.25
INP – Interactivity	≤ 200 ms	> 500 ms

1.1 Largest Contentful Paint (LCP)

LCP measures the render time of the largest image or text block visible in the viewport from when the page first started loading. It is the best proxy for *perceived load speed* — the moment the user feels the page has arrived.

What counts as an LCP element?

- **<img>** elements (including those inside <picture>)
- **<image>** elements inside an SVG
- **<video>** elements (poster image used)
- Elements with a CSS **background-image**
- Block-level elements containing **text nodes**

Root causes & fixes

LCP is almost always hurt by one of four things: a slow server, render-blocking resources, slow resource load times, or client-side rendering. Below are the most impactful interventions:

1. Preload the LCP image

If your LCP candidate is a hero image, ensure the browser discovers it as early as possible:

```
<!-- In <head> — rel=preload with fetchpriority=high --> <link rel="preload" as="image"
href="/hero.webp" imagesrcset="/hero-480.webp 480w, /hero-960.webp 960w, /hero-1440.webp 1440w"
imagesizes="100vw" fetchpriority="high">
```

■ *Never preload an image that isn't actually the LCP candidate — you'll waste bandwidth on non-critical resources.*

2. Optimise server response time (TTFB)

Google's guidance is TTFB < 800 ms. Poor TTFB cascades into every subsequent metric. Audit with:

```
# curl to isolate TTFB curl -o /dev/null -s -w "TTFB: %{time_starttransfer}s\n"
https://example.com/ # Lighthouse CLI npx lighthouse https://example.com
--only-categories=performance --output=json
```

3. Eliminate render-blocking resources

```
<!-- BAD: blocking stylesheet in <head> --> <link rel="stylesheet" href="/styles.css"> <!--
GOOD: inline critical CSS + async load the rest --> <style>/* critical above-the-fold CSS here
```

```
*/</style> <link rel="preload" href="/styles.css" as="style"
onload="this.onload=null;this.rel='stylesheet'"> <noscript><link rel="stylesheet"
href="/styles.css"></noscript>
```

4. Use modern image formats & responsive images

```
<picture> <source srcset="/hero.avif" type="image/avif"> <source srcset="/hero.webp"
type="image/webp"> <img src="/hero.jpg" alt="Hero image" width="1440" height="600"
loading="eager" fetchpriority="high"> </picture>
```

Always specify explicit **width** and **height** attributes on images. This lets the browser reserve the correct space before the image loads, preventing layout shift.

1.2 Cumulative Layout Shift (CLS)

CLS quantifies visual instability — how much page content unexpectedly moves during loading. Each shift is scored as *impact fraction* × *distance fraction*. CLS accumulates over the **entire session** using a windowed approach: shifts within 1 s of each other and within a 5 s window are grouped; only the largest window contributes to the final score.

Common CLS culprits

- **Images without dimensions** — browser doesn't reserve space
- **Ads, embeds, iframes** — late-injected content pushes content down
- **Web fonts causing FOUT/FOIT** — text reflow on font swap
- **Dynamically injected content** — banners, cookie notices, chat widgets
- **CSS animations** using non-composited properties (top, left, margin)

Fix 1: Reserve space for dynamic content

```
/* Reserve space for an ad slot */ .ad-slot { min-height: 250px; /* exact ad height */ contain:
layout; /* isolation boundary */ } /* Reserve space for a lazy-loaded image */ img {
aspect-ratio: 16 / 9; width: 100%; height: auto; /* browser calculates from aspect-ratio */ }
```

Fix 2: Use CSS transform for animations

```
/* BAD — triggers layout recalculation */ .modal { top: 0; transition: top 0.3s; } .modal.open {
top: 100px; } /* GOOD — compositor-only, no layout cost */ .modal { transform:
translateY(-100%); transition: transform 0.3s; } .modal.open { transform: translateY(0); }
```

Fix 3: Font display strategy

```
/* In @font-face — prevent FOUT from contributing to CLS */ @font-face { font-family: 'Inter';
src: url('/fonts/inter.woff2') format('woff2'); font-display: optional; /* best for CLS;
fallback if slow */ } /* Use size-adjust to match fallback metrics */ @font-face { font-family:
'Inter-fallback'; src: local('Arial'); size-adjust: 96.7%; ascent-override: 92%;
descent-override: 22%; line-gap-override: 0%; }
```

1.3 Interaction to Next Paint (INP)

INP replaced FID as a Core Web Vital in March 2024. Unlike FID, which measured only the delay before the browser started processing the first interaction, **INP measures the full duration** of all interactions (click, keypress, tap) and reports the worst-case latency at the 98th percentile over a session.

The interaction lifecycle has three phases: **Input Delay** (time until the event handler starts) + **Processing Time** (event handler execution) + **Presentation Delay** (rendering pipeline until next frame). Improving INP means reducing work in any of these phases.

Diagnosing INP

```
// Use PerformanceObserver to capture all interactions const observer = new PerformanceObserver((list) => { for (const entry of list.getEntries()) { if (entry.interactionId) { console.log({ type: entry.name, interactionId: entry.interactionId, duration: entry.duration, // total INP latency inputDelay: entry.processingStart - entry.startTime, processingTime: entry.processingEnd - entry.processingStart, presentationDelay: entry.duration - entry.processingEnd + entry.startTime, }); } } }); observer.observe({ type: 'event', buffered: true, durationThreshold: 16 });
```

Key optimisation strategies

- **Break up long tasks** — any task >50 ms on the main thread blocks interactions. Use scheduler.yield() or setTimeout(fn, 0) to yield.
- **Defer non-critical JS** — use import() dynamic imports and load third-party scripts with async or defer.
- **Avoid layout thrashing** — batch DOM reads before writes to prevent forced synchronous layouts.
- **Use web workers** for CPU-intensive work (parsing, sorting, data transforms) off the main thread.
- **Virtualize long lists** — rendering thousands of DOM nodes dramatically increases layout cost.

scheduler.yield() — the modern yielding pattern

```
async function processLargeDataset(items) { for (let i = 0; i < items.length; i++) { processItem(items[i]); // Yield to the browser every 50 items so interactions stay responsive if (i % 50 === 0) { await scheduler.yield(); // Chrome 115+ // Polyfill: await new Promise(r => setTimeout(r, 0)); } } }
```

1.4 Measurement & Tooling

Always measure CWV from **real user data (RUM)** in addition to lab tools. Lab tools (Lighthouse, WebPageTest) are great for catching regressions in CI/CD, but only field data reflects actual user conditions.

Tool	Type	Best For
Chrome UX Report (CrUX)	Field	Population-level 75th-percentile data, 28-day rolling
PageSpeed Insights	Field + Lab	Quick per-URL audit with CrUX overlay
Lighthouse (CLI/CI)	Lab	Automated regression testing in pipelines
WebPageTest	Lab	Filmstrip, waterfall, multi-step scripting
web-vitals.js library	Field/RUM	Lightweight <1 KB, reports all CWV to your analytics
Chrome DevTools Performance	Lab	Deep flamechart analysis, long-task attribution

```
// web-vitals.js — minimal RUM instrumentation import { onLCP, onCLS, onINP } from 'web-vitals'; function sendToAnalytics({ name, value, rating, id }) { fetch('/analytics', { method: 'POST', body: JSON.stringify({ metric: name, value, rating, id }), keepalive: true, // survives page unload }); } onLCP(sendToAnalytics); onCLS(sendToAnalytics); onINP(sendToAnalytics);
```

2. Structured Data & Schema Markup

Structured data is machine-readable information that you add to HTML to help search engines understand the content of your pages. While it is not a direct ranking signal, it unlocks **rich results** (enhanced SERP features) and contributes to Knowledge Graph entries and entity understanding, which can indirectly improve click-through rates by 20–30%.

2.1 JSON-LD vs Microdata vs RDFa

Google supports three formats for structured data, but **JSON-LD is strongly recommended** for all new implementations. It is decoupled from HTML markup, easy to generate server-side, and does not require modifying the DOM.

Metric	Good	Needs Work
JSON-LD	Recommended by Google. Script tag in <head> or <body>. Easy to maintain.	
Microdata	Attributes inline on HTML elements. Tightly coupled to markup.	Legacy
RDFa	Powerful but verbose. Used in some publishing/academic contexts.	Niche

2.2 Key Schema Types & Implementation

Article / NewsArticle

```
{ "@context": "https://schema.org", "@type": "NewsArticle", "headline": "How Core Web Vitals Affect Your Rankings in 2024", "description": "A deep dive into LCP, CLS, and INP optimisation strategies.", "image": [ "https://example.com/article-hero-1x1.jpg", "https://example.com/article-hero-4x3.jpg", "https://example.com/article-hero-16x9.jpg" ], "datePublished": "2024-03-15T08:00:00+00:00", "dateModified": "2024-04-01T10:30:00+00:00", "author": [{ "@type": "Person", "name": "Jane Developer", "url": "https://example.com/authors/jane" }], "publisher": { "@type": "Organization", "name": "Example Media", "logo": { "@type": "ImageObject", "url": "https://example.com/logo.png" } } }
```

Product

```
{ "@context": "https://schema.org", "@type": "Product", "name": "Premium Wireless Headphones", "image": "https://example.com/headphones.jpg", "description": "Over-ear noise cancelling headphones with 40h battery.", "sku": "WH-PRO-001", "brand": { "@type": "Brand", "name": "AudioCo" }, "offers": { "@type": "Offer", "url": "https://example.com/products/headphones", "priceCurrency": "USD", "price": "299.00", "priceValidUntil": "2024-12-31", "itemCondition": "https://schema.org/NewCondition", "availability": "https://schema.org/InStock" }, "aggregateRating": { "@type": "AggregateRating", "ratingValue": "4.7", "reviewCount": "2483" } }
```

FAQ Page

```
{ "@context": "https://schema.org", "@type": "FAQPage", "mainEntity": [ { "@type": "Question", "name": "What is the ideal LCP time?", "acceptedAnswer": { "@type": "Answer", "text": "Google considers LCP good when it occurs within 2.5 seconds of page load start." } }, { "@type": "Question", "name": "How do I measure Core Web Vitals in production?", "acceptedAnswer": { "@type": "Answer", "text": "Use the web-vitals.js library to collect real-user data and send it to your analytics endpoint." } } ] }
```

BreadcrumbList

```
{ "@context": "https://schema.org", "@type": "BreadcrumbList", "itemListElement": [ { "@type": "ListItem", "position": 1, "name": "Home", "item": "https://example.com/" }, { "@type":
```

```
"ListItem", "position": 2, "name": "Blog", "item": "https://example.com/blog/" }, { "@type":
"ListItem", "position": 3, "name": "Technical SEO Guide" } ] }
```

2.3 Rich Results & SERP Features

Rich results are enhanced SERP displays powered by structured data. Not all structured data types guarantee a rich result — Google decides based on quality, relevance, and user context.

Schema Type	Rich Result Feature	Key Requirement
Article / NewsArticle	Top Stories carousel	AMP or Web Stories (Top Stories)
Product	Product snippet, price drop	Valid Offer with price + availability
FAQPage	FAQ accordion in SERP	Clearly written Q&A pairs
Recipe	Rich recipe card	Image, cookTime, nutrition
Event	Event SERP feature	startDate, location, eventStatus
JobPosting	Job listings feature	datePosted, validThrough, hiringOrganization
HowTo	Step-by-step visual	Step with image + text
Review / AggregateRating	Star ratings in snippet	Must be first-party, not self-generated
VideoObject	Video carousel / rich snippet	thumbnailUrl, uploadDate, duration

2.4 Validation & Testing

- **Rich Results Test** (search.google.com/test/rich-results) — validate individual URLs or code snippets
- **Schema Markup Validator** (validator.schema.org) — full schema.org compliance check
- **Google Search Console** → **Enhancements** — monitor rich result eligibility at scale
- **structured-data-testing-tool** — CLI tool for CI/CD integration

```
# CI/CD — test structured data with Google's API curl -X POST
'https://searchconsole.googleapis.com/v1/urlTestingTools/mobileFriendlyTest:run' -H
'Content-Type: application/json' -d '{"url": "https://example.com/page"}' # Or using
structured-data-testing-tool npx structured-data-testing-tool -u https://example.com --schema
Article
```

> Common structured data mistakes: using aggregate ratings for self-generated reviews, marking up content not visible to users, using incorrect @type for the content, and omitting required properties (especially 'image' for Article types).

3. Crawlability & Indexing

Search engines must first **discover** your pages, then **crawl** them, then **render** them, and finally **index** them. Each stage is a potential failure point. Even excellent content will not rank if it cannot be reliably crawled and indexed.

3.1 Robots.txt

robots.txt is a plain-text file at the root of your domain that provides crawl directives to bots. It is a *voluntary* standard — well-behaved bots respect it, but it provides no security guarantee. Use it to manage crawl budget, not to hide sensitive content.

```
# /robots.txt — production example User-agent: * Disallow: /admin/ Disallow: /api/internal/
Disallow: /*?sort= Disallow: /*?filter= Disallow: /search Allow: /api/public/ # Crawl-delay not
supported by Googlebot — use Google Search Console instead # Crawl-delay: 2 User-agent:
Googlebot Allow: / User-agent: Googlebot-Image Disallow: /images/watermarked/ # Sitemap
declaration Sitemap: https://example.com/sitemap.xml Sitemap:
https://example.com/sitemap-images.xml
```

■ *Googlebot ignores Crawl-delay. To rate-limit Googlebot, use the crawl rate setting in Google Search Console.*

> **Critical pitfall:** robots.txt Disallow does NOT prevent indexing. A page that receives a link can still appear in the index (as a URL without a description). To prevent indexing, use noindex in the HTTP response headers or meta robots tag — but make sure the page is not disallowed in robots.txt, or Googlebot won't see the noindex directive.

3.2 XML Sitemaps

Sitemaps tell crawlers which URLs exist on your site and provide optional metadata about each URL. They are especially important for large sites, sites with thin internal linking, or content that doesn't receive many backlinks.

```
<?xml version="1.0" encoding="UTF-8"?> <urlset
xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
xmlns:news="http://www.google.com/schemas/sitemap-news/0.9"
xmlns:image="http://www.google.com/schemas/sitemap-image/1.1"> <url>
<loc>https://example.com/article/technical-seo-guide</loc>
<lastmod>2024-04-01T10:30:00+00:00</lastmod> <changefreq>monthly</changefreq>
<priority>0.8</priority> <news:news> <news:publication> <news:name>Example Media</news:name>
<news:language>en</news:language> </news:publication>
<news:publication_date>2024-04-01T10:30:00+00:00</news:publication_date> <news:title>Technical
SEO Master Guide</news:title> </news:news> <image:image>
<image:loc>https://example.com/images/seo-hero.jpg</image:loc> <image:title>Technical SEO Guide
Hero Image</image:title> </image:image> </url> </urlset>
```

For large sites, use a **sitemap index file** to reference multiple sitemaps, each containing up to 50,000 URLs and a maximum of 50 MB uncompressed:

```
<?xml version="1.0" encoding="UTF-8"?> <sitemapindex
xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"> <sitemap>
<loc>https://example.com/sitemap-posts-1.xml</loc> <lastmod>2024-04-01T10:30:00+00:00</lastmod>
</sitemap> <sitemap> <loc>https://example.com/sitemap-posts-2.xml</loc> </sitemap> <sitemap>
<loc>https://example.com/sitemap-categories.xml</loc> </sitemap> </sitemapindex>
```

3.3 Canonical Tags

The canonical tag (<link rel="canonical">) tells search engines which URL is the *preferred* version of a page when duplicate or near-duplicate content exists at multiple URLs. It is a **hint**, not a directive — Google may override it if it disagrees.

```
<!-- In <head> of every page — always self-reference canonical --> <link rel="canonical" href="https://www.example.com/article/technical-seo/"> <!-- HTTP header alternative — preferred for PDFs and non-HTML resources --> Link: <https://www.example.com/article/technical-seo/>; rel="canonical"
```

When to use canonicals

- **Paginated content** — /page/2 can canonical back to /page/1 OR each page self-canonicals (both patterns are valid)
- **Parameter variants** — /products/?color=red canonicals to /products/
- **Syndicated content** — instruct publishers to canonical back to your original URL
- **Mobile/AMP pages** — AMP pages should canonical to the canonical (non-AMP) URL
- **HTTPS/HTTP, www/non-www** — pick one canonical form and 301-redirect all others

> **Canonical chain length:** avoid chains of canonicals (A → B → C). Google may not follow chains beyond a few hops. Canonical each URL directly to the final preferred destination.

3.4 Crawl Budget Optimisation

Crawl budget is the number of URLs Googlebot will crawl on your site within a given time frame. It is determined by **crawl rate limit** (how fast Googlebot crawls without overloading your server) and **crawl demand** (how much Googlebot wants to crawl, based on popularity and change rate).

Crawl budget rarely matters for sites with <1,000 pages. It becomes critical at scale — e-commerce sites with faceted navigation, news sites publishing hundreds of articles daily, or sites with large URL spaces from query parameters.

Reducing crawl waste

- Block all parameter-generated duplicate URLs in robots.txt or via GSC URL parameters tool
- Return proper HTTP status codes — fix all 404s and 500s quickly; they waste crawl budget
- Use rel=nofollow on purely internal utility links (login, register, print) to reduce link equity dilution and unnecessary crawling
- Consolidate thin content pages — paginated archives with 1–2 items waste crawl budget
- Monitor crawl stats in Google Search Console → Settings → Crawl Stats for crawl anomalies

```
# Monitor crawl stats via GSC API curl -H "Authorization: Bearer $GSC_TOKEN" "https://searchconsole.googleapis.com/v1/sites/https%3A%2F%2Fexample.com/sitemaps"
```

3.5 Indexing Directives

Beyond robots.txt, you can control indexing at the individual-page level with meta robots tags and X-Robots-Tag HTTP headers.

```
<!-- Meta robots — comprehensive example --> <meta name="robots" content="index, follow"> <!-- default --> <meta name="robots" content="noindex, nofollow"> <!-- exclude entirely --> <meta name="robots" content="noindex, follow"> <!-- crawl links, don't index --> <meta name="robots" content="index, nofollow"> <!-- index but don't follow links --> <meta name="robots"
```

```
content="nosnippet"> <!-- no text snippet in SERP --> <meta name="robots"
content="max-snippet:160"> <!-- limit snippet to 160 chars --> <meta name="robots"
content="max-image-preview:large"> <!-- allow large image previews --> <meta name="robots"
content="noarchive"> <!-- no cached version --> <!-- Googlebot-specific overrides --> <meta
name="googlebot" content="noindex, nosnippet">

# X-Robots-Tag HTTP header — applies to ALL bots, great for non-HTML # Add to server config
(nginx example): location ~* \.(pdf|zip)$ { add_header X-Robots-Tag "noindex, noarchive"; } # Or
via PHP: header('X-Robots-Tag: noindex, noarchive', true);
```

4. Site Architecture & Performance

Site architecture is how pages are organised and interconnected. A well-designed architecture ensures that crawlers can efficiently discover all content, that link equity flows logically from high-authority pages to important deep pages, and that users can navigate intuitively.

4.1 URL Structure

URLs should be human-readable, consistent, and semantically meaningful. While Google can handle complex URLs, clean URLs are easier to share, earn more natural links, and provide subtle readability signals.

URL best practices

- Use **hyphens** to separate words, never underscores (/core-web-vitals vs /core_web_vitals)
- Keep URLs **lowercase** — configure server to 301-redirect uppercase variants
- Use the **shortest** descriptive URL that communicates the page content
- Avoid **stop words** (the, a, and, or) unless they're semantically important
- Maintain a **consistent URL pattern** across content types: /blog/[slug], /products/[category]/[slug]
- Implement **301 redirects** immediately when changing URL structures — never leave 404s on previously indexed URLs

```
# Nginx: enforce lowercase URLs and remove trailing slash (except root) server { # Redirect uppercase to lowercase if ($request_uri ~ "[A-Z]") { return 301 $scheme://$host${request_uri,,}; } # Remove trailing slash (except root) rewrite ^/(.*)/$ /$1 permanent; # Force www if ($host !~ ^www\.) { return 301 https://www.$host$request_uri; } }
```

4.2 Internal Linking

Internal links are the primary mechanism through which PageRank flows through your site. They also help Googlebot discover new pages and establish topical relationships between content.

PageRank sculpting principles

- **Flat architecture** — keep all important pages within 3 clicks of the homepage
- **Descriptive anchor text** — use keyword-rich anchors, not 'click here' or 'read more'
- **Hub-and-spoke model** — link pillar pages to all related cluster content; cluster pages link back to pillar
- **Fix orphan pages** — any page with zero internal links receives no PageRank and will rarely be crawled
- **Contextual links** beat navigational links for PageRank transfer — editorial in-body links carry more weight

> Audit orphan pages regularly: crawl your site with Screaming Frog, export all internal links, then find URLs in your sitemap that receive zero internal links. These are your orphans.

4.3 Page Speed & Core Infrastructure

Beyond Core Web Vitals, raw page speed affects crawl budget, user engagement, and conversion rates. The following server-level optimisations should be considered baseline for any production site.

HTTP/2 and HTTP/3

```
# Nginx — enable HTTP/2 (requires TLS) server { listen 443 ssl http2; ssl_certificate /etc/ssl/cert.pem; ssl_certificate_key /etc/ssl/key.pem; ssl_protocols TLSv1.2 TLSv1.3;
```

```
ssl_ciphers HIGH:!aNULL:!MD5; } # For HTTP/3 (QUIC) — nginx with quic patch or Caddy # Caddy automatically provisions TLS and enables HTTP/3: # example.com { # root * /var/www/html # file_server # }
```

Critical resource compression

```
# Nginx — Brotli (better than gzip) + gzip fallback brotli on; brotli_comp_level 6; brotli_types text/html text/css application/javascript application/json image/svg+xml font/woff2; gzip on; gzip_vary on; gzip_types text/html text/css application/javascript application/json image/svg+xml; gzip_min_length 1024;
```

Resource hints

```
<!-- dns-prefetch — resolve DNS for third-party origins early --> <link rel="dns-prefetch" href="//fonts.googleapis.com"> <link rel="dns-prefetch" href="//cdn.example.com"> <!-- preconnect — establish TCP + TLS early (use sparingly, max 2-3) --> <link rel="preconnect" href="https://fonts.googleapis.com"> <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin> <!-- prefetch — load next-navigation resources at idle time --> <link rel="prefetch" href="/about/" as="document"> <!-- modulepreload — preload ES modules --> <link rel="modulepreload" href="/js/app.js">
```

4.4 CDN & Caching Strategy

A Content Delivery Network (CDN) serves assets from edge nodes geographically close to the user, reducing latency significantly. Combined with aggressive caching, a CDN is one of the highest-ROI performance investments available.

Cache-Control directives

```
# Nginx — tiered caching strategy # Static assets (hashed filenames) — cache forever location ~* \.(js|css|woff2)$ { if ($request_filename ~* "-[a-f0-9]{8,}\.(js|css|woff2)$") { add_header Cache-Control "public, max-age=31536000, immutable"; } } # Images — cache 1 year (use versioned filenames or ETags) location ~* \.(jpg|jpeg|webp|avif|png|svg|gif|ico)$ { add_header Cache-Control "public, max-age=31536000"; add_header Vary "Accept"; # for content negotiation (webp vs jpg) } # HTML pages — short cache, allow revalidation location ~* \.html$ { add_header Cache-Control "public, max-age=3600, stale-while-revalidate=86400"; } # API responses — no store location /api/ { add_header Cache-Control "no-store"; }
```

Stale-while-revalidate

The stale-while-revalidate directive is particularly valuable for SEO — it allows a browser (or CDN) to serve a stale cached response immediately while fetching a fresh one in the background. This eliminates cache-miss latency without ever showing users outdated content for long.

```
Cache-Control: max-age=3600, stale-while-revalidate=86400 # Meaning: fresh for 1 hour; after that, serve stale for up to 24 hours # while revalidating in the background
```

4.5 JavaScript SEO

JavaScript-rendered content poses unique challenges for SEO. Googlebot can execute JavaScript, but it does so in a **second wave of indexing** — sometimes hours to days after the initial crawl. This means JS-rendered content may be absent from the index temporarily, and any links discovered only via JS will be slower to crawl.

Rendering strategies comparison

Strategy	How It Works	SEO Impact
Static Generation (SSG)	HTML pre-built at build time	Best — full HTML available immediately
Server-Side Rendering (SSR)	HTML rendered on each request	Excellent — full HTML in response

Incremental Static Regen. (ISR)	Static pages rebuilt on-demand	Excellent — combines SSG speed with freshness
Client-Side Rendering (CSR)	Empty HTML shell; JS builds DOM	Risky — content in second-wave indexing
Dynamic Rendering	SSR for bots, CSR for users	Acceptable — but cloaking risk if misimplemented

JavaScript SEO checklist

- ✓ Ensure critical content and internal links are present in the initial HTML response (SSR/SSG)
- ✓ Audit JS-rendered pages in Google Search Console → URL Inspection → View Tested Page
- ✓ Use Chrome's 'Disable JavaScript' (DevTools → ■+Shift+P → 'Disable JavaScript') to see what Googlebot sees in wave 1
- ✓ Avoid infinite scroll for paginated content — implement paginated /page/N URLs alongside infinite scroll
- ✓ Ensure <title>, <meta description>, and canonical tags are rendered in SSR output, not set only by client JS
- ✓ Test for JS errors in the Googlebot user agent using rendering logs
- ✓ Keep JS bundle sizes minimal — large bundles delay rendering for Googlebot's WRS (Web Rendering Service)

Quick Reference: Technical SEO Audit Checklist

Use this checklist when auditing a new site or performing a quarterly technical health check.

Core Web Vitals

- ✓ LCP ≤ 2.5 s at 75th percentile (field data, not lab)
- ✓ CLS ≤ 0.1 at 75th percentile
- ✓ INP ≤ 200 ms at 75th percentile
- ✓ LCP image uses fetchpriority='high' and rel=preload
- ✓ All images have explicit width and height attributes
- ✓ No layout shifts caused by web fonts (font-display: optional or swap + size-adjust)
- ✓ Long tasks (>50 ms) broken up with scheduler.yield()
- ✓ TTFB < 800 ms from representative geos

Structured Data

- ✓ JSON-LD used for all structured data (not Microdata)
- ✓ Schema validates in Rich Results Test with no errors
- ✓ Article/NewsArticle includes image (3 aspect ratios), datePublished, dateModified, author
- ✓ Product includes Offer with price, priceCurrency, availability
- ✓ BreadcrumbList implemented on all non-homepage URLs
- ✓ Structured data matches visible page content (no hidden markup)
- ✓ GSC Enhancements tab monitored for rich result errors

Crawlability & Indexing

- ✓ robots.txt is valid and accessible at /robots.txt
- ✓ All sitemaps declared in robots.txt and submitted to GSC
- ✓ XML sitemap contains only canonical, indexable URLs
- ✓ No pages have conflicting noindex directive + robots.txt Disallow
- ✓ Canonical tags self-reference on all pages; no canonical chains
- ✓ Pagination handled consistently (self-canonical or paginated canonicals)

✓ All 4xx errors resolved; no soft 404s
✓ No redirect chains longer than 2 hops

Site Architecture & Performance

✓ All pages reachable within 3 clicks from homepage
✓ No orphan pages (zero internal links)
✓ Anchor text is descriptive and keyword-relevant
✓ HTTPS enforced with valid certificate; no mixed content
✓ HTTP/2 or HTTP/3 enabled
✓ Brotli or gzip compression enabled for text assets
✓ Static assets served with Cache-Control: immutable (hashed filenames)
✓ CDN used for static assets and HTML
✓ Critical content rendered in server-side HTML (not CSR-only)
✓ Core pages load without JavaScript errors

Further Resources

The following official resources are the authoritative references for the topics covered in this guide. All links were current as of publication.

- **Google Search Central** — developers.google.com/search
- **Core Web Vitals — web.dev** — web.dev/explore/learn-core-web-vitals
- **INP Deep Dive** — web.dev/articles/inp
- **Schema.org Full Type Hierarchy** — schema.org/docs/full.html
- **Google Rich Results Test** — search.google.com/test/rich-results
- **Chrome UX Report (CrUX)** — developer.chrome.com/docs/crux
- **web-vitals.js on npm** — npmjs.com/package/web-vitals
- **Lighthouse CI Documentation** — github.com/GoogleChrome/lighthouse-ci
- **Google Search Console Help** — support.google.com/webmasters
- **Sitemaps Protocol** — sitemaps.org/protocol.html