CogNovaMX

MX:
THE INTRO

The starter guide to building a web that works for AI agents — and everyone else

Tom Cranstoun

Published and printed by mxprintworks.com
120 Main Street, Largs, Scotland

© CogNovaMX. First published 2026.

All rights reserved. For information about permission to reproduce selections from this book, write to Permissions, MXPRINTWORKS, LPC Design + Print, 120 Main Street, Largs, Ayrshire, KA30 8JN Scotland UK

First Edition
Printed in the United Kingdom of Great Britain and Northern Ireland

Cover Design by Scott McGregor
Book Design by Tom Cranstoun

For information on booking the author for an event
please visit https://allabout.network/mx/the-author.html

Contents

MX: The Introduction **4**

Part A — The Business Case

Part B — The Technical Foundation

The AI Tipping Point — CMS Kickoff 2024 **33**

Footnotes and References **41**

The Machine Experience Manifesto **42**

Work with the Author **51**

MX: The Introduction

Understanding the machines reading your website

Tom Cranstoun

March 2026

Contents

MX: The Introduction 2

Part A — The Business Case 2

Revenue you cannot see leaving . . . 2
Three structural failures . . . 2
When machines fail commerce . . . 3
When machines fail people . . . 4
Eight reasons MX pays for itself . . . 5
What MX-ready looks like . . . 8

Part B — The Technical Foundation 11

From business case to practice . . . 11
Audit — what machines see right now . . . 12
Understand — the machines reading your site . . . 14
Prioritise — the MX readiness diagnostic . . . 18
Implement — fix the source, not the model . . . 23
Measure — track and sustain . . . 26

MX: The Introduction

Understanding the machines reading your website — and the revenue at stake.

SHARED CHAPTER — INTRODUCTION TO MX

This chapter appears in every MX book. It establishes the core **M**achine e**X**perience concept: the business case for why structured metadata matters, what the machines reading your website are, how they work, and why explicit context prevents hallucination. Whether you are reading MX: The Protocols or MX: The Handbook, this shared foundation ensures every reader starts from the same understanding before the book-specific material begins.

Part A — The Business Case

Revenue you cannot see leaving

Machines are mediating commerce right now. They compare your products, check your availability, and recommend you — or your competitors. The commercial question is straightforward: are they completing transactions on your site, or moving on to someone who made it easier?

Adobe's Holiday 2025 data makes the scale visible.[1] AI referrals to retail sites surged by 700%. Travel referrals rose by 500%. Conversion rates from AI-referred visitors now lead human traffic by 30%, and those visitors spend 50% longer on sites. Machine-mediated commerce moved from experimental to revenue driver in a single quarter.

In January 2026, four major platforms launched machine commerce systems within eight days: Amazon Alexa+ (5 January), Microsoft Copilot Checkout (8 January), Google Universal Commerce Protocol (11 January), and Anthropic Claude Cowork (12 January).[2] What industry analysts predicted would take 12-24 months compressed to 6-9 months or less. Machine-mediated commerce is infrastructure, not experiment.

The machines want to buy from you. The question is whether your website gives them what they need.

Three structural failures

The web suffers from three interlocking failures that no amount of AI improvement can fix.

The first is hostile design. Every website demands something before delivering value. Accept cookies. Dismiss the banner. Scroll past the advertisement. Close the pop-up. Miss the toast notification. Click through the carousel. Navigate the infinite scroll. Find the actual content buried under marketing. Accept more cookies. The web was built to be read. It has become an obstacle course. Humans persist through frustration. Machines cannot. A toast notification that vanishes after three seconds is invisible to software that reads the DOM on a half-second cycle. A cookie consent wall that requires JavaScript interaction blocks every server-side machine from reaching the content behind it. The hostile web is not merely annoying — it is architecturally incompatible with machine access.

The second is content imprisoned in platforms. Content management systems solved real problems for decades. But every CMS traps content in its own format, its own database, its own API. Migration costs are prohibitive. Vendor lock-in persists. And now AI agents cannot read any of it — because the content was designed for human eyes inside a proprietary system, not for machines operating across open standards. The content does not describe itself. It depends on the platform to describe it, and the platform's description serves the platform's interests.

The third is the assumption that structured data alone is the answer. Adding Schema.org JSON-LD to a page addresses what machines can *read* — but it does not address what they can *do*. It does not fix the hostile design patterns that block interaction. It does not free content from platform lock-in. It does not govern who may access the content, under what terms, for what purposes. Structured data is one layer in a practice that must span the entire interaction surface.

MX addresses all three. It is not a markup vocabulary. It is the practice of ensuring that every file, every page, every digital asset carries enough context for machines to understand it and act on it — without guessing. Schema.org is one tool in that practice. Semantic HTML is another. YAML frontmatter, meta tags, robots.txt directives, sitemap.xml, llms.txt, accessibility patterns, and self-describing document formats (cogs) are others. The practice is broader than any single standard, because the problem is broader than any single standard can solve.

When machines fail commerce

The commercial impact plays out in specific, traceable ways. Two cases illustrate what happens when machines lack the structured context to complete transactions.

Case study — the purchase that couldn't happen

A buyer decides to purchase a Kindle Scribe Colorsoft, the 11-inch colour e-ink device at £570. He has done the research, read the reviews, and settled on the product. Credit card ready. Amazon makes it surprisingly difficult to take his money.

The device launched in the US on 10 December 2025. Amazon announces the UK and Germany will follow in "early 2026." Months later, the UK listing on Amazon.co.uk still says "coming soon" with no date. The US product page exists, the "Buy Now" button works, but the delivery estimate to a UK address stretches between 24 March and 14 May. A motivated buyer willing to pay international shipping, and the best Amazon can offer is "sometime in the next three months, maybe."

He asks his AI assistant for help. It can find the product, compare the specs, and quote the price, but it cannot complete the purchase, check real-time stock across regions, or even email him the link. Close, but not useful enough when it matters.

This failure exposes four gaps that Machine Experience would close.

First, product pages need machine-readable data. They would expose availability status, regional launch dates, and delivery estimates in structured formats. An AI assistant could then report: "The Kindle Scribe Colorsoft isn't available in the UK yet. Amazon expects UK availability in March 2026. Shall I set a reminder or register your interest?"

Second, cross-region awareness. When a UK customer views a US product that has a pending UK listing, the experience should acknowledge this, not buried in small print, but as a structured status field that both humans and machines can read.

Third, purchase intent needs to become first-class data. "Notify me" buttons exist, but they are blunt instruments. An MX approach would let buyers express intent with context: this product, this configuration, this delivery address, this price ceiling. That intent becomes data the retailer can act on, prioritising stock allocation, offering pre-orders, or routing to the fastest fulfilment option.

Fourth, transactions need to be AI-accessible. If a buyer asks an AI assistant to purchase something, the assistant should be able to check availability, compare delivery options across regions, and either complete the purchase or explain clearly why it cannot. This requires product and commerce APIs designed with machine users in mind, not screen-scraping a website built solely for human eyes.

This is not really about a Kindle

It is about what happens when digital experiences are designed for one type of user. Amazon's shopping experience works well for humans browsing a website. The moment another type of user enters (an AI assistant, a price comparison agent, an automated purchasing system), the experience collapses because none of the information is structured for machine consumption. MX does not replace human-centred design. It extends it. When product availability, pricing, delivery estimates, and purchase intent are expressed in formats that both humans and automated systems can read, the result is better for everyone. The human gets a clearer, faster experience. The software can actually help. The retailer, in turn, captures demand it is currently losing.

When machines fail people

The commercial cases above involve lost revenue. The cases below involve lost trust — and sometimes safety. The structural cause is identical: machines acting on incomplete data. The urgency, though, is different.

Case study — when algorithms price-gouge a tragedy

On 18 January 2026, two high-speed trains collide near Adamuz in Córdoba, Spain.[3] Forty-six people die. Two hundred and ninety-two are injured. High-speed rail services between Madrid and Andalusia are suspended for days.

Within hours, pricing algorithms at airlines, car hire firms, and bus companies detect a surge in demand on the Madrid-Málaga and Madrid-Seville corridors. They do exactly what they are designed to do: they raise prices. Flight fares jump by 200%. Buses sell out. Car hire becomes the last resort for stranded travellers who have no other way to reach medical appointments, work commitments, or family.

The uncomfortable truth: the algorithms work correctly. They see demand increase. They see supply constrain. They adjust prices upward. This is what dynamic pricing does. It is not a bug. It is the intended behaviour. The problem is that not everyone designs them to recognise the difference between a music festival selling out and a national tragedy killing dozens of people.

Spain's national ombudsman launches an investigation. Iberia, to its credit, manually intervenes, capping fares at €99 and adding extra flights. Many other carriers do not. Their systems carry on optimising for yield while families scramble to get home.

Three Machine Experience principles apply directly.

The first principle is humans always in the loop. Dynamic pricing systems operate with minimal oversight. An MX-informed approach would require that crisis events trigger human review before price adjustments take effect. Not after passengers have already been charged three times the normal fare. Before.

The second is designing for both humans and machines. The pricing algorithm sees numbers: demand up, supply down, price up. It does not see a train wreck. An MX approach would feed contextual signals into pricing systems: civil emergency declarations, transport network disruptions, news event classification. The machine needs to understand the situation, not just the spreadsheet.

The third is considering the broader impact. MX asks us to consider the wider consequences of machine decisions, not just the immediate commercial outcome. A pricing system that extracts maximum revenue from people fleeing a disaster site is not serving anyone well. Not the travellers, not the brand, and not the transport network that depends on public trust.

This is not a theoretical problem. It is a design problem with practical solutions. A pricing system informed by MX principles would monitor external signals (transport disruption feeds, civil protection alerts, news APIs) and apply constraints when crisis conditions are detected. Not necessarily a full price freeze, but a cap, a flag for human review, or a temporary shift to fixed pricing on affected routes. The data exists. Spain's civil protection system issues alerts. Rail operators publish service suspensions. News organisations classify events. The missing piece is connecting these signals to the machines making pricing decisions, and giving those machines rules that reflect human values rather than pure demand curves.

There is a business case here too. Spain's ombudsman is investigating transport companies. Consumer association FACUA calls for legal reform. Passengers share screenshots of price surges on social media. The reputational damage from algorithmic price gouging during a national tragedy costs these companies far more than the short-term revenue gained. Trust, once broken during a crisis, takes years to rebuild. A machine that raises prices during a disaster does not just lose a customer. It creates an adversary.

Beyond commerce

A patient asks an AI assistant about drug interactions before a hospital appointment. The NHS website lists contraindications in prose paragraphs, not structured data. The machine parses "should not be taken with" from one section and "may be taken with food" from another — but cannot reliably distinguish which drug each statement refers to because the HTML uses visual proximity rather than semantic association. It confidently tells the patient there are no interactions. The pharmacist catches the error at the dispensary. The patient is fine. Scale this to millions of daily health queries, though, and the law of large numbers guarantees that some automated systems will hallucinate drug safety information that not everyone catches in time.

A parent searches for school catchment areas using an AI assistant. The local council's website publishes boundary maps as images with no alt text and catchment rules in PDF documents that machines cannot parse. The assistant guesses based on partial postcode data scraped from a planning application. The family moves house based on its recommendation. The child is refused a school place because the actual catchment boundary runs down the middle of their new street.

Government services, healthcare, education, legal guidance — every domain where machines answer questions on behalf of humans faces the same structural problem. Commerce has the most visible revenue impact, but the consequences of unstructured content reach into every interaction where people trust automated systems to read websites accurately.

Pattern

The commercial cases and the public service cases reveal the same structural failure. Amazon's commerce platform cannot express product information in machine-readable formats. Spain's transport pricing systems cannot consume contextual signals that would prevent harm. The NHS website cannot associate drug interaction warnings with specific medications in a way automated readers can parse. The council cannot expose catchment boundaries as structured data. In each case, the systems do exactly what they are designed to do — and the design is insufficient because it accounts for only one type of user, one type of interaction, one set of inputs.

The commercial and ethical cases are driven by different urgencies but share the same structural cause: machines acting on incomplete data. Commerce loses revenue. Public services lose trust, and sometimes lives. The technical solution — explicit structure that prevents machines from guessing — serves both.

These case studies share a root cause that no amount of AI improvement can address: decisions about what data is valid, what constitutes "reasonable," and what constraints apply were never made. Nobody defined the governance rules that pricing algorithms should follow. Nobody established what information an AI system is allowed to act on. Nobody created the data structures that make safety constraints possible. The open-source community recognises this gap: frameworks like AgentLock[4] now exist specifically to add the authorization layer that AI agents lack — declarative security metadata, scoped access control, and audit trails enforced at the infrastructure level before any tool call executes. The fact that such frameworks need to be built after the fact proves the point. The AI project will lose to the data project.[5] Not immediately, but inevitably. Organisations that deploy AI without first establishing data governance discover that their expensive AI investments fail in ways that governance would have prevented. The failed transaction, the price-gouging tragedy, the drug interaction error — each one would have been prevented if the data layer had been governed first. Governance is not the alternative to AI ambition. It is the precondition for it.

When machines cannot complete your desired action — whether that is purchasing a product, checking drug interactions, finding the right school, informing readers of a product recall, or completing a contact form — they are likely to move to sources that provide explicit structure, or to hallucinate an answer from incomplete data. First-mover advantage exists across every sector. Sites that work early are more likely to become trusted sources these systems return to repeatedly.

Eight reasons MX pays for itself

Machine Experience (MX) is the practice of making your entire web interaction surface work for machines — not just metadata and markup, but every element they encounter: toast notifications that vanish before they can be read, sliders and carousels that hide content behind interactions, stepped workflows that trap automated systems in loops, micro-animations that convey meaning only visually, and hidden JavaScript variables that hold state no software can see. When these systems must guess at any of these, they hallucinate. When all context is explicitly present, they get it right. Here is why that matters — commercially and beyond.

Every hallucination has a cost beyond the wrong answer. When a machine lacks structured context, it must reason through ambiguous content (burning inference compute, consuming energy, and increasing latency), only to produce a less accurate result. Multiply this across billions of daily automated queries and the waste is measurable: data centres spending processing power on guesswork that structured metadata would have eliminated. MX turns expensive inference into cheap lookup. The environmental and operational case is as compelling as the commercial one.

1. Goal completion is at risk. Machines mediate a growing share of web interactions, not just commerce. As the data at the start of this chapter shows, AI-referred traffic is surging and converting at higher rates than human visitors. These same systems answer health questions, compare university courses, check government eligibility criteria, and verify legal requirements. When your site lacks MX structure, they may skip you in recommendations or, worse, hallucinate answers your site should have provided authoritatively. You see nothing in your analytics. The traffic may never arrive. Sites that work reliably today are more likely to become trusted sources that automated visitors return to repeatedly.
2. Hallucination damages your brand. Machines cannot make things up when you have already written it down. When you haven't, they guess, and the customer blames you, not the AI. A hospital whose drug interaction page confuses automated readers bears the reputational consequence when patients receive incorrect guidance. A university whose entry requirements are misquoted loses applicants who self-select out based on hallucinated grade boundaries. MX gives these systems the truth. Without it, they guess.
3. Legal risk drops. MX patterns (semantic HTML, explicit state, structured data) simultaneously address machine compatibility and WCAG accessibility requirements. Both audiences need overlapping explicit structure. A single implementation reduces exposure to accessibility litigation, which can involve substantial legal costs and settlements, with thousands of cases filed annually in the US alone. For government and healthcare, the legal exposure extends to duty-of-care obligations when citizens rely on AI-mediated access to public services.
4. First-mover trust compounds. Machines tend to learn which sites work reliably. Early MX adopters build computational trust that accumulates over time. This applies to council websites as much as retailers: a system that reliably extracts planning application status from one council is more likely to recommend that council's site over others. Latecomers face an invisible disadvantage. Preferred sources may already be established, and displacing them can be difficult.
5. Structure prevents hallucination. Formal documents with metadata inheritance, operational runbooks, and defined scripts give machines verified facts rather than inferred guesses. The risk of hallucination drops when they read instead of reason. A healthcare provider whose drug information uses Schema.org MedicalEntity markup provides facts directly. One whose information sits in unstructured prose invites guesswork, and guessing about drug interactions carries consequences that pricing errors do not.
6. Training beats searching. LLMs trained on properly structured content develop a fuller understanding of your products, services, and information. Instead of searching and scraping at inference time (an expensive, unreliable process), models that ingested your structured metadata during training can cite you accurately from memory. Consider the difference: a charity whose impact data is published as Schema.org NonprofitType with structured outcome metrics gets cited accurately in automated responses. One whose impact data sits in a PDF annual report gets paraphrased, summarised, and occasionally fabricated. This applies to any organisation that wants to be cited correctly: charities, research institutions, professional bodies, and public services.
7. Documentation discipline drives accuracy. Machines working with properly-structured documentation are measurably more accurate than those parsing unstructured sites. Higher accuracy is likely to translate to higher recommendation probability. What you document once (product specs, pricing, policies, eligibility criteria, opening hours, catchment boundaries) serves every system that reads it, permanently. The discipline of structured documentation reduces AI hallucination whilst improving human comprehension.
8. Your strategic assets are trapped. You have built a solid review history on Amazon, genuine customers vouching for your products over several years. Migrate to a different commerce platform, and every one of those endorsements stays behind. You are effectively starting from scratch despite years of earned trust. The same lock-in affects every sector: a university's course descriptions trapped in a proprietary CMS, a charity's impact data locked in a donor platform, a council's planning records in a legacy system that cannot export structured data. MX makes these assets portable and sovereign, readable by any system regardless of platform. (This is the **Entity Asset Layer**, treated fully in the implementation chapters.)

Real-world ROI

The economic case for MX becomes concrete when you examine implementation patterns. Consider a mid-size retailer selling speciality kitchenware online. Their product pages display prices, availability, and specifications visually — styled for human shoppers. A machine trying to compare their espresso machines against competitors cannot reliably extract the price (is “2.030” two thousand and thirty euros, or two point zero three?), cannot determine which items are in stock, and cannot parse the specification tables because they use visual layout rather than semantic markup.

The retailer begins with Schema.org Product and Offer markup, embedding JSON-LD structured data into their existing templates. Each product page now carries machine-readable pricing with ISO 4217 currency codes, explicit availability status, and structured specification data. But structured data alone is not MX — it is the foundation.

The team then addresses their rendering pipeline. Product specifications that were previously assembled by client-side JavaScript are moved into the served HTML. A machine that fetches the page now receives complete content without executing scripts — critical, because most AI agents read raw HTML and never run JavaScript. Prices that were injected dynamically from API calls are rendered server-side into the DOM.

Next, they add MX metadata — governance tags that declare content freshness, editorial policy, and machine-interaction permissions. These sit alongside the Schema.org markup, filling the gaps that established standards leave open: when was this content last verified? May a machine act on it? What confidence level should an agent assign to this data?

Finally, they audit the micro-interactions that humans navigate by instinct but machines cannot perceive. The “Add to Basket” button that relied on a JavaScript click handler gets a semantic `<button>` element with explicit `aria-label` and visible state. The colour-swatch selector that used drag gestures exposes its options as structured `<select>` elements in the DOM. The checkout confirmation that flashed as a toast notification and vanished after three seconds is replaced with a persistent status element that any DOM reader can find. Each of these changes makes the same page work better for humans with accessibility needs *and* for every machine that visits.

The full implementation takes their development team a week. The incremental cost remains modest — the same HTML serves both human and machine visitors, and every change also improves accessibility compliance and SEO performance.

The result: machines can now accurately extract, compare, recommend, and transact with their products. Automated referral traffic begins appearing in analytics. Conversion rates from these referrals exceed human traffic, matching the industry pattern Adobe’s data confirms. The initial investment pays back with the first machine-completed transaction. Everything after that is captured revenue that was previously invisible loss.

This pattern repeats across every MX implementation. Whether you are structuring product data, health information, educational content, or government services, the initial investment in explicit structure pays back immediately.

The retailer’s experience illustrates a pattern that holds across every domain. Structured data is the starting point, not the finish line. MX extends to the entire interaction surface — rendering, metadata, and every micro-interaction on the page. A slider that reveals product variants only when a human drags it hides those variants from every automated reader. A stepped checkout workflow that stores progress in JavaScript session variables creates a black box no software can traverse. Micro-animations that pulse a button to signal “click here” mean nothing to programs that cannot see pixels. Hidden form fields that carry pricing tiers, discount codes, or user segments in JavaScript variables are inaccessible to any system reading the HTML. These are not edge cases. They are standard patterns on millions of websites, and every one of them breaks for automated visitors.

Check your own server logs

The claims above are verifiable today. Your server logs likely contain evidence of machine traffic that your analytics tools are not reporting.

Search your access logs for these User-Agent strings:

- **GPTBot** — OpenAI's web crawler (ChatGPT browsing)
- **Claude-Web** — Anthropic's web retrieval
- **PerplexityBot** — Perplexity's search crawler
- **Bytespider** — ByteDance's crawler (TikTok's parent company)
- **CCBot** — Common Crawl (training data for most LLMs)
- **Googlebot** — Google's crawler (including AI Overviews)
- **bingbot** — Microsoft's crawler (powers Copilot)

On Apache or Nginx, a command like `grep -i "GPTBot\|Claude-Web\|PerplexityBot" access.log | wc -l` gives you a count of machine visits that likely never appeared in your analytics dashboard.

What you find may surprise you. These visitors arrived, read your content, and either used it to answer someone's question about your business — or failed and moved on to a competitor. You have no visibility into which outcome occurred unless your content is structured for machines to read accurately.

This diagnostic takes five minutes. The answer tells you whether the rest of this chapter is urgent or merely important.

What MX-ready looks like

Before the technical detail, here is where this leads. An organisation that implements MX reaches a measurable destination:

- Machines convert at rates matching or exceeding human visitors. Automated referral traffic becomes captured revenue, not invisible loss.
- Zero catastrophic errors in machine-mediated transactions. No £203,000 bookings when someone wanted £2,030. No hallucinated drug interactions. No fabricated school catchment boundaries.
- Break-even at the second automated interaction. The initial implementation investment pays back immediately and permanently.
- Your strategic assets are sovereign. Reviews, product knowledge, customer data, and operational procedures are portable across any platform, readable by any software, owned by you.
- You control what these systems do with your content. You write the instructions, in plain, structured formats they follow. No inference. No guessing. No dependence on AI getting smarter.
- Every automated visitor can learn how your site works. Systems that received accurate results are more likely to return. Your competitors face catch-up costs that increase with every quarter they delay.

These are not aspirational targets. They are measurable criteria you can verify against your own analytics.

FINANCIAL SNAPSHOT FOR EXECUTIVES

The machine traffic data presented in this chapter tells a clear story: AI-referred visitors convert at higher rates and spend longer on sites. Without MX, that traffic might skip your site entirely — and you may never see it leave.

MX implementation costs vary by complexity, but the payback pattern is consistent: break-even at the first successful machine-mediated transaction.

Early movers build trust that late movers struggle to match. Automated systems tend to learn which sites work reliably, and latecomers may struggle to displace established trust scores. Machine-mediated browsing is projected to become the dominant mode within two to three years.

The core proposition is simple: **Your business, readable by every machine on Earth.** MX gives you control over how automated systems read your content. You write the instructions. They follow them. No black box.

Two questions for your next board meeting: ask your CTO whether you are implementing MX, and ask your CMO what percentage of your traffic is AI-mediated. The first determines readiness. The second determines urgency.

What this chapter does not claim

MX is a structural practice, not a guarantee. Several distinctions matter:

- **MX does not guarantee AI recommendation.** Structuring your content correctly makes it possible for machines to recommend you. Whether they do depends on content quality, relevance, and competitive alternatives. Structure is necessary, not sufficient.
- **MX does not replace good content.** Structured metadata on mediocre content produces accurately represented mediocre content. The quality of what you publish still determines whether machines find it worth citing.
- **MX does not require rebuilding your website.** Most MX improvements are additive — Schema.org markup, semantic HTML adjustments, heading hierarchy fixes. They layer onto existing sites without redesign.
- **First-mover advantage is a hypothesis, not a proven law.** The compounding trust argument is logical and supported by early data, but the machine commerce market is too new for definitive longitudinal proof. The Adobe data shows the trend. The compounding effect remains to be measured over multiple years.

These caveats do not weaken the case. They sharpen it. MX is a practice with clear technical rationale and growing commercial evidence. Overstating the certainty would undermine the credibility this chapter aims to build.

For CxOs and decision-makers: you now understand the business case.

> *People are not afraid of AI. They are afraid of AI they cannot control. MX gives them control. Machines are mediating commerce today, and organisations without MX are losing machine-referred traffic invisibly. The following sections explain the technical reality for your architects and developers. If you need to delegate from here, the rest of this book covers financial justification, technical implementation, and organisational structure. The organisational dimension — dedicated roles, accountability models, and team structures for MX — is covered in the implementation chapters.*

The window for first-mover advantage is measured in quarters, not years.

What the books offer

Two companion books — MX: The Protocols and MX: The Handbook — show you how to implement MX, the practice that ensures machines can read your website, understand your content, and complete the actions that serve your goals. Together they cover:

- Why websites need optimisation for machines (and the consequences for any web goal, not just commerce)
- How to add metadata and instructions so they work with facts, not guesses
- Which MX patterns prevent hallucination and enable goal completion
- How to implement these patterns practically across platforms
- How the Entity Asset Layer enables sovereign portability across platforms and machines
- What organisational roles and accountability structures MX requires

MX applies to every web goal. Whether you sell products, inform patients, publish research, guide citizens through government services, recruit students, collect donations, or enable downloads, machines need explicit structure to complete those actions. A hospital needs automated systems to extract accurate drug information. A council needs them to report correct planning application status. A university needs them to cite accurate entry requirements. A charity needs them to explain eligibility criteria for support services. The technical patterns are identical — semantic HTML, structured data, explicit state — regardless of whether the goal is a purchase or a public service inquiry.

This introduction is published as a standalone primer and also appears as the opening chapter of both MX books. MX: The Protocols is the definitive technical reference for architects, consultants, and practitioners implementing MX at scale. MX: The Handbook provides step-by-step platform-specific implementations for developers, UX designers, and content strategists. Both books share continuously updated appendices hosted at allabout.network.

Part B — The Technical Foundation

From business case to practice

The insight behind Machine Experience came from an unexpected direction.

At CMS Kickoff 2024 in Florida, the conference agenda offered only a slight focus on AI. Speakers mentioned it during their talks — it was the topic of the moment in the industry. What I heard, though, was not about AI creating content. It was about AI reading content. It turned my expectations upside down, fired up my brain, and I could not stop taking notes.

Throughout the event, the speakers challenged how I thought about AI's role in content management. Given the translation, bias, regulation challenges, and the fundamental lack of trust in AI-generated content, one conclusion kept surfacing: AI is better suited for *consuming* content than creating it. That distinction — AI as reader, not writer — is the foundation of everything in this book. The business case in Part A showed the commercial consequences. What follows is how I arrived at the technical response.

Nick Rudd led a discussion that reframed how we should think about AI behaviour. He began by asking how an eight-year-old purchases toys. Simple and direct, without the baggage of brand loyalty or complex decision-making.

Kids are smart. They know what they want.

The eight-year-old goes to a search engine and types:

"I WANT A TOY PLESIOSAUR"

What didn't the eight-year-old do?

- They didn't go to Amazon.
- They didn't refine the search.
- They didn't look at brands.
- They didn't read customer reviews.
- They didn't sort by price or popularity.
- They didn't read the price (well — mostly).
- They didn't create an account.
- They didn't click on dropdowns.
- They ignored video and animations.
- They didn't read the copy.

What happens when AI reads content? It behaves like that eight-year-old:

- It ignores retailers.
- It probably won't refine the search.
- It ignores brands.
- It might skim customer reviews.
- It might sort by price.
- It won't create an account.
- It won't follow your dropdowns.
- It won't give you an email address — you no longer know who your customers are.
- It ignores videos and animations.
- It ignores the "see our shop" links.
- It skim-reads the copy if it can find it among the ads.

I published these observations in CMS Critic, arguing that the real value of AI lies in consuming content, not generating it, and proposing that "Machine" should be treated as a fourth device type alongside mobile, tablet, and desktop.[6] The plesiosaur became the foundational metaphor for everything that followed. A child's search and a machine's search are structurally identical: both bypass the entire apparatus of modern web design and go straight to the content. Both are indifferent to your brand, your navigation, and your carefully crafted user journey. Both want a direct answer. The difference is scale. The child searches once. These systems search billions of times a day, on behalf of every human who asks them a question.

The design-by-engineer trap

Open any product page in a browser. It makes perfect sense to a human: a clean image, a price, a description, a buy button. The engineers who built it are clever — they took the site owner's desires and made polished presentations with text and prices, highlighting discounts. They made it fast and effective.

Now view the page source. Buried in the HTML are the brand name, the price (often more than one is shown), and the description. Discounts and age suitability sit in freeform text. A human reading this raw source would struggle. A machine has no chance of reliably extracting structured meaning.

Engineers create text and image boxes in the CMS. They dress them up as heroes, carousels, cards, accordions, and tabs. These are visual containers with interactions — and nothing more.

This is the Design-by-Engineer Trap.

The trap compounds at the CMS level. Designers fixate on presentation when envisioning a website — style guides with fonts, colours, components, and interactions. Then the content team is asked to fit business content into these visual containers:

- Testimonials go into the cards component.
- Special offers go into the lists component.
- "Coming soon" messages go into the image-with-link component.
- Bulk discount messages go into the text component.

> > *Content modelling by developers. The CMS knows about visual containers, not business objects. The machine reading the page sees `<div class="card">`, not a product, a review, or an offer. Every piece of structured business meaning is lost in the presentation layer. No semantic meaning is given to the objects described. No content model exists beneath the presentation. Look at the HTML of any major retail site — it looks as though AI would need a bespoke language model for every brand. That is not going to work.* ***Something major must happen****.*

When you add Schema.org markup — Product, Offer, PriceSpecification — a machine can see the price, the description, the brand, the discount, and the age applicability. It knows this is a product, not an advertisement. The data is already on the page. The missing piece is expressing it in formats machines can parse. The next section shows exactly what this looks like in HTML.

Inspired by Steve Krug's "Don't Make Me Think," the solution became clear: you cannot fix the million-plus models on Hugging Face, but you can fix what they read. The websites, not the AI, need improving.

Machine Experience (MX) is the practice of making every aspect of your web presence work for machines — from metadata and semantic markup through to interaction patterns, state management, and dynamic behaviours. Toast notifications, sliders, stepped workflows, micro-animations, and hidden JavaScript variables all affect whether automated software can complete your desired action. MX is not UX (visual design), not SEO (ranking signals), and not accessibility (WCAG), though the convergence is real: automated systems, search engines, and accessibility users all rely on explicit structure rather than visual design. One MX implementation serves all three audiences.

Audit — what machines see right now

The invisible users

Now there's a new class of user — the invisible users. They're called "invisible" for two reasons: they're invisible to site owners (blending into analytics, coming once and leaving) and the interface is invisible to them (they cannot see animations, colour, toast notifications, or loading spinners). These are machines visiting your website and performing actions without your awareness.

Most companies don't track AI bot traffic. Some prohibit AI bots entirely through robots.txt directives or block them using services like Cloudflare Identity checks. Modern AI browsers (ChatGPT, BrowserOps, Comet, Strawberry, Neo, DIA) do identify themselves as bots in their User-Agent strings, but these strings cannot be trusted — they're trivially spoofed by any developer.

The MX patterns required for these invisible users — explicit structure, semantic HTML, persistent state — also happen to benefit users with disabilities who rely on screen readers and keyboard navigation. However, this is a side effect of designing for machines, not the design driver. The case for action —

whether measured in revenue, patient safety, citizen trust, or educational access — drives the technical requirements.

Some machines operate as browser extensions running alongside human users. Others are automation frameworks controlled by AI scripts. Some are AI browsers accessing sites directly. Site owners can no longer reliably distinguish between human visitors and machines. The traffic looks identical in analytics, but the visitor's capabilities and limitations differ fundamentally.

People are building agents that monitor Slack channels for commands, then execute web actions autonomously and report back. The shift from passive assistant to active agent is complete. Anthropic's Claude Cowork operates as a coworker rather than a tool, managing local file systems, orchestrating complex project workflows, and executing multi-step tasks through multi-agent architecture.[7] This is the agentic era: machines reading websites, comparing options, making recommendations, and completing actions on behalf of humans.

These machines struggle with patterns that also confuse users with disabilities. They're blind to visual cues like flashes of information. They struggle to differentiate between European, American, and British monetary formats. They don't understand what spinners mean. When faced with five "Read More" buttons on a page, they can't easily determine which relates to which content. They process form validation errors differently from humans — whilst humans iteratively fix issues one by one, AIs may simply abandon difficult websites.

Before and after — what MX looks like in HTML

The difference between a page that works for machines and one that fails often comes down to a few lines of markup. Consider a product price displayed two ways.

Without MX — visual-only markup:

```html
<div class="price-box">
  <span class="big-red">€2.030,00</span>
  <span class="green">In Stock</span>
</div>
```

A machine sees a `<div>` with class names that mean nothing semantically. The price string "€2.030,00" is ambiguous — is that two thousand and thirty euros (European formatting) or two point zero three? The machine guesses. It guesses wrong. A booking for €2,030 becomes €203,000, or the machine may skip the product entirely.

The European decimal comma is not an isolated case. Every major cultural convention for displaying numbers creates different parsing hazards for machines:

Value	US/UK Display	European Display	Indian Display
Two thousand and thirty	$2,030.00	€2.030,00	₹2,030.00
One million	$1,000,000.00	€1.000.000,00	₹10,00,000.00

The Indian lakh system groups digits differently from both Western conventions: three digits from the right, then pairs. One million rupees is written ₹10,00,000 — ten lakh. A machine trained predominantly on Western number formatting encounters "10,00,000" and has no reliable way to determine whether this represents ten lakh (one million), a formatting error, or a malformed price with misplaced separators. At a crore (₹1,00,00,000 — ten million), the divergence from Western grouping widens further.

Three cultural conventions, three sets of decimal and grouping rules, and any single web page can serve customers from all three traditions simultaneously. The visual formatting serves the local human reader. The machine-readable formatting must serve every machine on Earth. ISO 4217 defines the currency code (EUR, INR, GBP).[8] Schema.org requires price values to use a fixed period as the decimal separator with no grouping separators. In JSON-LD: `"price": "2030.00"`. In Microdata: `content="2030.00"`. The rule holds at any scale — `1000000.00`, never `10,00,000` or `1.000.000,00`. Cultural formatting for display; ISO standards for machines. One implementation, zero ambiguity.

With MX — semantic markup with structured data:

```
<div class="price-box" itemscope itemtype="https://schema.org/Offer">
  <span itemprop="price" content="2030.00">€2.030,00</span>
  <meta itemprop="priceCurrency" content="EUR">
  <link itemprop="availability" href="https://schema.org/InStock">
  <span class="green">In Stock</span>
</div>
```

The human sees exactly the same page. The machine reads unambiguous structured data: price is 2030.00 (fixed decimal, no grouping), currency is EUR (ISO 4217), availability is InStock. No guessing. No hallucination. The visual presentation — European commas, Indian lakhs, whatever the locale demands — remains unchanged for human visitors. The structured metadata resolves every cultural ambiguity for every machine that reads it.

The same pattern applies to every element machines interact with: buttons, forms, navigation, availability status, delivery estimates. Semantic HTML and structured data turn ambiguous visual design into explicit machine-readable facts.

This ambiguity is not unique to currency. A gallon is not a gallon. The US gallon is 3.785 litres. The Imperial gallon is 4.546 litres — 20% larger. A US pint is 473ml; a British pint is 568ml. Even the fluid ounce differs: 29.6ml in the US, 28.4ml in Britain. Same words, different quantities, different countries. Humans have needed explicit standards to communicate accurately for centuries. The ISO system exists precisely because "everyone knows what a gallon means" was never true. MX applies the same principle to web content: when the same term can mean different things in different contexts, make the meaning explicit. The need for standards did not arrive with AI. AI simply made the cost of ambiguity visible.

Why isn't everyone doing this already?

Schema.org launched in 2011. The markup shown above is not new technology. The question any reader should ask is: if structured data is so valuable, why is adoption still low after fifteen years?

The barriers are organisational, not technical. Markup drifts out of sync with the content it describes — a price changes in the CMS but nobody updates the JSON-LD. CMS templates rarely expose structured data fields to content editors, making Schema.org invisible in the authoring workflow. QA processes validate visual rendering and functional behaviour but not machine-readability. Nobody's KPIs include "percentage of pages with valid structured data."

The result is predictable: Schema.org gets added during a redesign, decays over six months as content changes outpace markup updates, and quietly breaks without anyone noticing — because no human visitor is affected.

MX addresses this as an organisational discipline, not just a technical pattern. It requires dedicated ownership (someone whose job includes machine-readability), validation in the QA pipeline (automated checks that structured data matches visible content), and CMS integration (content fields that generate structured data automatically rather than requiring separate markup). The implementation chapters cover these organisational requirements in detail.

Understand — the machines reading your site

They're machines, not magic

When we talk about machines reading websites, we're not talking about artificial general intelligence or sentient software. We're talking about machines — software programs with specific technical capabilities and limitations — that process your website's content on behalf of human users.

These machines operate through statistical pattern-matching — next-token prediction using probability distributions calculated from training data. When machines process your website, they're performing weighted averaging and probability calculations, not "understanding" content the way humans do. They guess the next most likely word based on patterns they've seen before.

This statistical foundation explains both their capabilities and their failures:

Machines hallucinate when they encounter incomplete context. They must "think," generating confident answers by guessing based on statistical co-occurrence patterns. Without structured data (Schema.org, semantic HTML) providing complete context, they fabricate details that seem plausible but are wrong.

These are the same errors that produce 0% correctness on mathematical reasoning tasks despite fluent explanations. MX is the act of adding metadata and instructions so AI doesn't have to think. When all context is explicitly present, hallucination decreases dramatically.

Linguistic bias compounds the problem. Current AI systems tokenize English more efficiently than compound languages like German and Dutch. This bias begins at the training data level: approximately 44% of Common Crawl (the public dataset powering most large language models) is English content, with no other language exceeding 6% representation.[9] A German user must type "ultrathink" to get more processing power from the system, a word that isn't even proper English but uses German-style compounding. The irony is complete: using a fake English word that mimics German grammar to command machines that don't actually think. This linguistic inequity affects how machines process multilingual websites, giving English-language content structural advantages.

Explicit structure matters because machines convert your HTML into mathematical representations through weighted averaging of co-occurrence patterns. `<div class="button">` and `<button>` appear similar to humans but create different statistical signatures. Semantic HTML produces distinct patterns that machines process reliably, whilst visual-only distinctions disappear in the averaging process.[10]

Calling them "machines" instead of "AI assistants" or "intelligent agents" counters dangerous anthropomorphism. AI researchers deliberately anthropomorphise their creations — calling them "assistants," "intelligent," or claiming they "understand" — building artificial confidence in users. When users believe machines "understand" context or can "infer" meaning, they trust these systems beyond their actual capabilities.

I want to be clear about my stance on AI: I'm not complaining about these systems or highlighting edge cases like "How many R's in strawberry?" or users generating obvious junk. I genuinely praise AI for its ability to generate coherent text that people understand. Statistical pattern-matching that produces human-readable responses across countless domains is a genuine achievement. My focus isn't on what's wrong with AI, but on what we can do to make it work better. When creators craft proper inputs (semantic HTML, structured metadata), provide quality training data, enable effective live searches, and implement appropriate guardrails, we get dramatically better results. Hallucinations decrease. Accuracy increases. Commerce transactions complete successfully. The solution isn't to criticise AI systems — it's to recognise that better-structured inputs produce better outputs for everyone: users, machines, and businesses alike.

What the machines actually are

Machines reading the web come in five forms. You cannot detect which type is visiting — User-Agent strings are trivially spoofed, as noted above. Design for the most constrained machine, and you serve them all.

Machine Type	Examples	Sees	Misses	Disability Equivalent
Server-side	ChatGPT, Claude	Raw HTML, semantic structure, metadata	JavaScript-rendered content, CSS layout, dynamic updates	Blind user with basic screen reader — needs semantic HTML
In-browser	Copilot, extensions	Full rendered DOM, dynamic content, interactive states	Visual hierarchy, CSS styling, transient notifications	Screen reader user with JS support — needs explicit state attributes
Browser automation	Perplexity Playwright	Everything above plus screenshots, visual layout	Design intent, brand meaning, cultural context	Sighted user with motor impairments — needs semantic structure for interaction
Local	Ollama, on-device LLMs	Raw HTML (like server-side) with smaller context windows	Everything server-side misses, plus content exceeding token budget	User with cognitive disabilities — needs simple, clear structure
Agentic OS	Anthropic Cowork	Aggregated information from multiple machine types	Inconsistencies across interaction points	Project manager coordinating diverse abilities — needs consistent patterns

The disability equivalents describe overlapping needs, not identical ones. Machines and disabled users both rely on semantic HTML and ARIA landmarks. The overlap is large enough to justify a single implementation. They diverge in specific areas, though: machines need JSON-LD structured data (screen readers do not parse it). Screen readers need focus management and keyboard navigation (machines do not interact with focus). Both need semantic HTML, heading hierarchy, and persistent feedback. The convergence is practical, not absolute.

Machine limitations and the convergence principle

Machines have zero-tolerance technical constraints. Unlike humans who persist through ambiguity and find workarounds, machines may fail silently and disappear from recommendations. These constraints parallel human disabilities, not as analogy, but as literal technical fact:

Constraint	What Machines Need	Accessibility Parallel
Blindness — cannot see CSS or visual layout	Semantic HTML (`<button>`, `<nav>`, `<main>`)	Screen readers need the same markup
Zero-tolerance parsing — cannot guess intent from appearance	Explicit semantics (`<button>` not `<div class="btn">`)	Keyboard users need proper focus management
Limited context — token budget constraints on local machines	Concise structure, clear heading hierarchy	Users with cognitive disabilities need simplicity
No persistence — transient content vanishes before processing	Persistent feedback (`role="alert"`, DOM-reflected state)	Screen readers miss toast notifications too
State invisibility — cannot access JavaScript variables	State in HTML attributes (`data-state`, `aria-invalid`)	Keyboard users need visible focus indicators

This convergence is not coincidental, but it is not total. Both groups (machines and users with disabilities) lack access to visual design cues. Both rely on semantic structure. Both benefit from explicit state. Where they diverge — machines needing structured data formats, disabled users needing interaction affordances — the implementations do not conflict. One codebase serves both audiences. The business case (machine commerce, conversions, revenue) drives the investment. Accessibility compliance is a substantial side benefit, not a separate initiative. In practice, adopting these principles changes how you build everything: metadata-driven architecture, context declaration, executable documentation, and WCAG-informed design become inseparable from the work itself.[11]

Component names must match use. Machines parse ARIA labels and semantic class names as definitive signals, not hints. A section labelled `aria-label="Author biography"` that actually contains a blog tagline causes catastrophic extraction failures. Unlike humans who infer meaning from context, machines treat labels as source of truth.

How machines access websites — training vs inference

Before understanding the readiness levels machines progress through on your website, you must distinguish between two fundamentally different mechanisms of website access. These mechanisms operate on different timescales, use different technologies, and have different implications for Machine Experience implementation. Yet both benefit from the same underlying structural patterns.

During model training, large language models ingest web content through datasets like Common Crawl — massive archives of crawled websites that form the knowledge base machines draw upon when answering queries. This ingestion happens months or years before machines interact with users, making it fundamentally historical rather than real-time.

Common Crawl operates as a web archive service that periodically crawls public websites, respecting robots.txt directives and sitemap.xml files. The crawled content becomes part of training datasets that teach models about the structure, patterns, and information available on the web. When machines reference your product specifications or company information without visiting your live site, they're drawing from this historical knowledge base acquired during training.

This training-time access has specific characteristics. It's indirect: your website doesn't interact with the model; it interacts with crawlers that build datasets used later for training. It's historical: the content the model learns from could be months or years old by the time users interact with the trained model. It's complete: crawlers attempt to index entire sites systematically, following sitemaps and link structures. Crucially, it respects robots.txt: ethical crawlers honour your access control directives.

The linguistic bias mentioned earlier stems from this training phase. The English-dominant imbalance in Common Crawl becomes permanently embedded in the model's statistical patterns during training. The model doesn't "learn" languages equally; it builds probability distributions weighted toward English-language patterns.

Inference-time access — real-time direct interaction During user queries, machines may directly access your website in real time. When someone asks ChatGPT "What laptops does Example Store currently offer?", the system might fetch your live website using browser automation, API calls, or direct HTTP requests. This is inference-time access — software retrieving current information whilst processing a specific user query.

This direct access operates differently from training-time ingestion. It's direct: the machine or its supporting infrastructure fetches your website whilst the user waits. It's real-time: the content retrieved is your current live site, not a historical snapshot. It's specific: the system targets particular pages or information relevant to the user's query rather than fully crawling your entire site. Critically, it may not respect robots.txt: software executing user queries might access content regardless of crawl directives, treating robots.txt as guidance rather than absolute constraint.

Browser automation agents (using frameworks like Playwright or Puppeteer) demonstrate inference-time access clearly. When users ask questions requiring current information, these machines launch headless browsers, navigate to relevant pages, execute JavaScript, wait for dynamic content to load, and extract the specific information needed to answer the query. This happens in seconds during the user's conversation, not months earlier during model training.

The distinction matters for authentication and authorisation. Content blocked by robots.txt won't enter training datasets through Common Crawl, but inference-time machines actively executing user queries might access that content anyway. If your content truly must remain private, authentication mechanisms (login requirements, API keys, IP restrictions) provide reliable access control whilst robots.txt alone does not.

Why both mechanisms matter for Machine Experience Understanding this distinction changes how you implement Machine Experience patterns. Some patterns primarily serve training-time ingestion. Others primarily benefit inference-time access. The most effective patterns serve both mechanisms simultaneously.

Sitemap.xml serves both mechanisms. During training, crawlers follow your sitemap to discover and index your content structure fully. During inference, machines might fetch your sitemap to understand your current site architecture before targeting specific pages. One file, two access mechanisms, both benefiting from clear hierarchical structure.

Semantic HTML serves both mechanisms. Training datasets built from semantically marked-up content teach models better structural patterns than datasets built from visual-only HTML. At inference time, machines parsing your live site extract information more reliably when heading hierarchies, landmark roles, and semantic elements provide explicit structure. The same markup benefits both phases.

Schema.org structured data serves both mechanisms particularly well. During training, JSON-LD blocks teach models about entity relationships, product attributes, and structured information patterns. At inference time, machines can directly parse JSON-LD to extract current pricing, availability, ratings, and specifications without needing to interpret prose descriptions. Training benefit and inference benefit from identical implementation.

The llms.txt file has a structural problem that limits its reach. It is served as a text or markdown MIME type, not HTML. Common Crawl ingests HTML pages; it does not typically ingest non-HTML files. What's more, llms.txt is rarely included in sitemap.xml. This means training-time crawlers may never discover it, so its content never enters model knowledge bases. At inference time, automated systems typically do not fetch llms.txt either; it is too broad and untargeted for specific user queries. A system answering "What laptops does Example Store sell?" goes straight to product pages, not a site-level directory file. The result: llms.txt risks falling between both mechanisms, serving neither effectively. To close this gap today, publish the same content as an HTML page (for example, `/llms.html` or a rendered `/about/for-machines` page) and include that page in your sitemap. Training crawlers ingest the HTML version during their periodic crawl, and the structured guidance becomes part of the model's knowledge about your site. The Gathering (the independent standards body that governs MX specifications) is proposing a new standard to address this critical lack, so that machine discovery files are properly ingested by both training and inference mechanisms.

Training vs learning vs codification An important distinction: machines do not get "trained" when they visit your site. Their neural network weights are fixed. What actually happens involves three distinct layers:

Layer	What Happens	Cost	Persistence
Training (AI company's work)	Models learn general capabilities from datasets like Common Crawl — HTML syntax, Schema.org patterns, structured data extraction	$50-100M per model (one-time)	Permanent but historical
In-context learning (per session)	Machine reads your HTML, understands your specific patterns, navigates your site	$0.10-1.00 per session	**Disappears when session ends**
Codification (what MX creates)	Semantic HTML and Schema.org markup capture knowledge permanently in the page itself	Time to implement (one-time)	**Permanent and reusable**

Without MX, every machine must re-learn your site structure from scratch every session: re-read, re-discover navigation, extract information, then forget everything. With MX, the structure IS the captured knowledge. These systems spend zero time learning and 100% of their processing capacity extracting accurate information. Implement once, every visitor benefits forever, saving compute and energy costs as well.

Prioritise — the MX readiness diagnostic

"AI will figure it out" fallacy

You've seen the failures. You understand the constraints. At this point, a reasonable objection arises.

The common objection: "AI is getting better all the time, why worry? It will work itself out."

The objection deserves an honest answer. Google, OpenAI, Anthropic, and Meta are investing billions in better parsing, larger context windows, and more capable models. Frontier models are genuinely improving at extracting meaning from ambiguous HTML. This is real progress, and dismissing it would be intellectually dishonest.

The stronger rebuttal is economic, not technical. Every time a machine must reason through ambiguous content — interpreting whether "2.030,00" is two thousand or two point zero three, inferring product availability from visual cues, guessing which error message belongs to which form field — it burns inference compute. That reasoning costs energy, increases latency, and produces less accurate results. MX turns expensive inference into cheap lookup. Structured metadata eliminates the reasoning step entirely. The machine reads a fact instead of inferring one. This is cheaper, faster, and more accurate regardless of how capable the model becomes.

The economic argument holds even for frontier models. The diversity argument makes it unavoidable.

Unknown machine problem Site owners have no idea which model is visiting their site. Is it a small LLM running on a mobile device with fewer than 500 million parameters? Is it a frontier model with hundreds of billions? Is it an in-browser extension with a local LLM prioritising privacy? Is it a custom-trained domain-specific model?

You cannot detect machine capabilities reliably. No standardised capability announcement exists, and User-Agent strings offer no help. You cannot serve different HTML based on sophistication because you don't know which system is parsing your content, and google penalises sit that serves different content to different browsers.

Diversity explosion Over 1 million models on Hugging Face[12] — the open-source platform where researchers and companies publish machine learning models for anyone to download and run — with wildly different capabilities:

The size distribution shows extreme diversity:

- Over 90% have fewer than 1 billion parameters
- The vast majority have fewer than 500 million parameters
- Roughly 70% have fewer than 200 million parameters
- Over 40% have fewer than 100 million parameters

The growth trajectory tells its own story. The platform added 1 million models in just 335 days, compared to 1,000+ days for the first million.[13] The wild west is getting wilder.

Why "waiting for AI to improve" fails First, there's no standardisation. No central authority controls machine capabilities. No way to demand parsing standards when no imperative exists. Everyone does what they want, giving lip service to standards without enforcement.

Second, the diversity paradox. Large frontier models are getting better at handling ambiguity, but small models (7B, 13B parameters) deployed on edge devices cannot handle the same complexity. You have no way of knowing which model is visiting your site. Optimising for "average" AI means failing for 40%+ of machines.

Third, local and edge deployment. Browser extensions with local LLMs (privacy-focused users), mobile machines with smaller models (resource constraints), and custom domain-specific models (specialised capabilities) will never have the computational power of frontier models. These machines are proliferating, not disappearing.

Tiered implementation — Must, Should, Could The principle is right: if the most constrained machine can parse your page, every machine can. Yet "design for the worst machine" as an absolute rule creates a false impression that every site needs every pattern immediately. In practice, implementation follows a tiered priority:

Priority	Pattern	Benefits	Effort
Must	Semantic HTML, heading hierarchy, alt text, server-side rendering	All machines, screen readers, search engines	Low — template-level changes
Should	Schema.org JSON-LD for core entities (Product, Organisation, Article)	Frontier models, mid-tier models, rich search results	Medium — per-content-type implementation
Could	Full strategic redundancy (data attributes, multiple metadata formats, Dublin Core)	Edge models, local models, maximum compatibility	High — systematic across all content

The Must tier is non-negotiable. It is the accessibility baseline, the SEO baseline, and the machine-readability baseline. The Should tier captures the majority of machine commerce value. The Could tier serves the long tail of constrained models that benefit from redundant formats.

Most organisations reach meaningful machine-readiness at the Should tier. The Must tier alone puts you ahead of most competitors. The Could tier is for organisations where machine interactions are revenue-critical and the audience includes constrained local models.

Strategic redundancy — multiple formats for unknown capabilities Because you cannot detect which machine is visiting your site or what capabilities it possesses, MX principles require providing information redundantly across multiple formats. This isn't bloat or inefficiency — it's strategic redundancy that ensures the least capable system can extract critical information whilst more sophisticated ones can use richer formats.

The detection problem is unsolvable. User-Agent strings are trivially spoofed. No standardised capability announcement protocol exists. You cannot serve different HTML based on sophistication because you have no reliable way to identify what is parsing your content. Therefore, you must assume the worst whilst providing for the best.

Consider how MX redundancy works in practice. Pricing appears in visible text ("£2,030"), Schema.org JSON-LD structured data (`"price": "2030", "priceCurrency": "GBP"`), and HTML data attributes (`data-price="2030" data-currency="GBP"`). A basic text parser gets the visible price. A structured data parser gets machine-readable values. An HTML attribute parser gets explicit semantics. Information lives in multiple semantic layers simultaneously.

State gets declared redundantly too. Checkout progress exists as visible text ("Step 2 of 4"), ARIA attributes (`aria-current="step"`), data attributes (`data-checkout-step="2" data-total-steps="4"`), and Schema.org BreadcrumbList. Screen readers announce current position. Simple parsers read data attributes. Sophisticated machines extract full navigation context from structured data.

Metadata appears in multiple formats: OpenGraph tags for social platforms, Twitter Cards for Twitter-specific display, Schema.org JSON-LD for structured extraction, Dublin Core for academic contexts, and llms.txt for machine discovery. Each format serves different consumers, but all describe the same underlying content.

Product information exists at multiple specificity levels. Product title in H1 heading, short description in meta description, detailed description in Schema.org Product type, specifications in structured data tables, and downloadable PDF datasheet. Basic machines extract the title. Advanced machines parse full specifications. Human users download detailed documentation.

This redundancy serves a critical strategic purpose: you're building for an unknown audience with unknown capabilities. A small mobile LLM with 100M parameters needs explicit text and simple structure. A frontier model with 200B+ parameters can extract from complex JSON-LD schemas. A browser extension reads data attributes. A CLI machine parses only served HTML. By providing information in multiple formats simultaneously, you ensure all machine types can extract what they need at the level they can process.

The efficiency argument fails here. Some developers resist redundancy as "bloating" HTML. This misunderstands the problem. An extra 2KB of Schema.org JSON-LD that prevents an order-of-magnitude pricing error isn't bloat. It's infrastructure. The redundancy pays for itself in the first successful machine-mediated transaction.

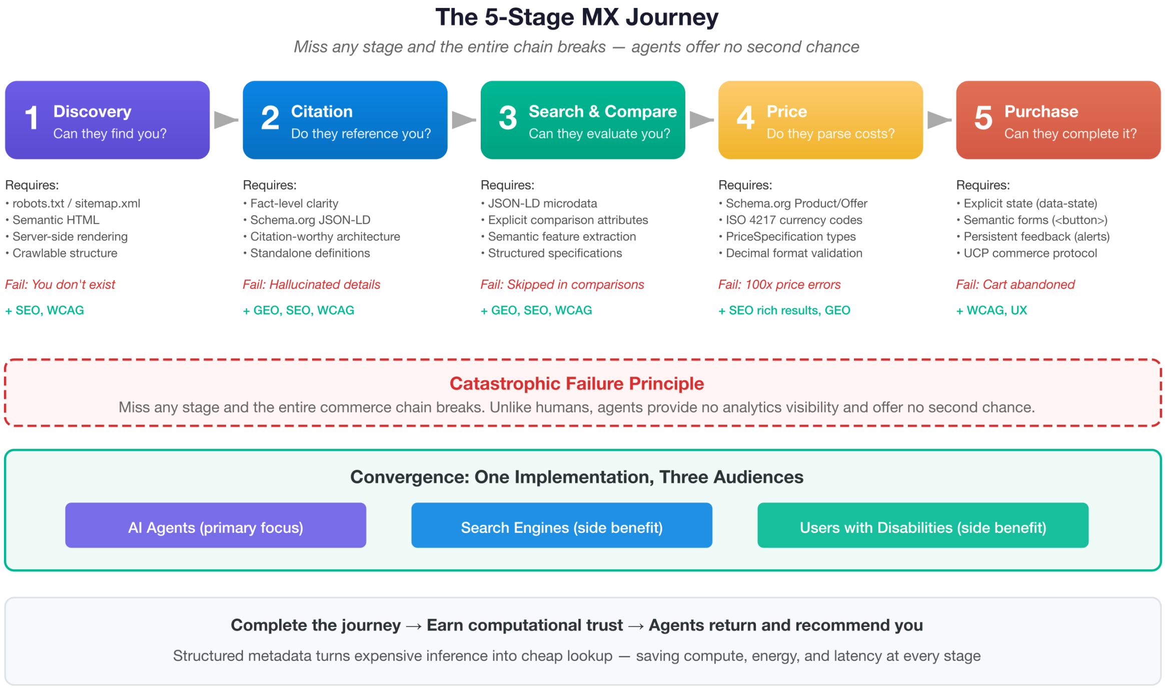
Strategic redundancy is an MX principle, not a bug. When you don't know which machine is reading your content or what capabilities it possesses, you provide information at multiple levels to ensure universal compatibility. This approach mirrors accessibility best practices — alt text AND visible captions AND ARIA labels — because both disciplines design for unknown capabilities.

Waiting is not a strategy. The solution is to fix what machines read.

MX readiness model (0-5)

Your website has machine readers right now. People are asking ChatGPT about your products, using Copilot to compare your services, and running machines to check your availability. Each successful interaction may reinforce your site as a reliable source. Each failure risks removing you from the map.

When machines interact with your website, they encounter a predictable progression of readiness levels. Each level describes both what the publisher provides and what machines can do as a result. The readiness model spans both training-time ingestion (Discovery) and inference-time access (Citation, Search, Transaction, Return). The same MX patterns serve both mechanisms — and the same patterns simultaneously improve SEO, GEO (Generative Engine Optimisation — being cited accurately in AI-generated responses), and WCAG accessibility. A single MX implementation serves all three disciplines.


The 5-Stage MX Journey
Miss any stage and the entire chain breaks — agents offer no second chance
1 Discovery
Can they find you?
2 Citation
Do they reference you?
3 Search & Compare
Can they evaluate you?
4 Price
Do they parse costs?
5 Purchase
Can they complete it?
Requires:
• robots.txt / sitemap.xml
• Semantic HTML
• Server-side rendering
• Crawlable structure
Fail: You don't exist
+ SEO, WCAG
Requires:
• Fact-level clarity
• Schema.org JSON-LD
• Citation-worthy architecture
• Standalone definitions
Fail: Hallucinated details
+ GEO, SEO, WCAG
Requires:
• JSON-LD microdata
• Explicit comparison attributes
• Semantic feature extraction
• Structured specifications
Fail: Skipped in comparisons
+ GEO, SEO, WCAG
Requires:
• Schema.org Product/Offer
• ISO 4217 currency codes
• PriceSpecification types
• Decimal format validation
Fail: 100x price errors
+ SEO rich results, GEO
Requires:
• Explicit state (data-state)
• Semantic forms (<button>)
• Persistent feedback (alerts)
• UCP commerce protocol
Fail: Cart abandoned
+ WCAG, UX
Catastrophic Failure Principle
Miss any stage and the entire commerce chain breaks. Unlike humans, agents provide no analytics visibility and offer no second chance.
Convergence: One Implementation, Three Audiences
AI Agents (primary focus)
Search Engines (side benefit)
Users with Disabilities (side benefit)
Complete the journey → Earn computational trust → Agents return and recommend you
Structured metadata turns expensive inference into cheap lookup — saving compute, energy, and latency at every stage

The MX Readiness Model (0-5) — Not Ready, Basic, Structured, Attested, Registered, Audited

Level 0 — not ready Most websites on the internet are at Level 0. The site has auto-generated meta tags from a CMS, boilerplate social sharing cards, or templated HTML — but none of this was placed with deliberate machine readability in mind. Machines visiting a Level 0 site must infer meaning from visual layout, guess at business identity, and hallucinate missing context. Recommendations based on Level 0 content are unreliable.

A Level 0 site is invisible to the machine economy. Not because it cannot be found, but because nothing it says can be trusted.

Level 1 — discovery (basic) The machine doesn't know you exist. You're not in its knowledge base.

An important distinction: discovery happens through two mechanisms. Training-time ingestion builds the historical knowledge base through datasets like Common Crawl, where machines learn about your site during model training months or years before user interactions. Inference-time access occurs when machines search for current information during user queries through direct site access and browser automation. Both mechanisms require the same MX patterns (sitemap.xml, semantic HTML, structured metadata), but understanding the distinction matters for security, authentication, and content freshness expectations.

MX requires crawlable structure (robots.txt compliance for training-time crawlers, sitemap.xml for both mechanisms), semantic HTML markup that serves training datasets and live parsing, server-side rendering for JavaScript-heavy content, and quality content that search engines and machines can discover and rank.

Without Level 1, you risk being invisible. The machine recommends competitors because you are not in its knowledge base (training gap) and it cannot find current information when users ask (inference gap). The improvement step: crawlable structure, semantic HTML, server-side rendering.

We implement MX patterns for machine discovery through both training and inference mechanisms. SEO improvement is an automatic outcome, not a separate task.

Level 2 — citation (structured) The machine is aware of your site and can recommend it.

MX requires fact-level clarity (each statistic, definition, concept needs standalone clarity), Schema.org JSON-LD for AI platforms, and citation-worthy content architecture optimised for being featured in AI responses, not just found.

Without Level 2, the machine knows you exist but cannot vouch for you. It may fill the gaps with fabricated details, or move on to a source it trusts more. The improvement step: fact-level clarity, Schema.org JSON-LD, citation-worthy architecture.

We implement MX patterns for machine citations. GEO improvement is an automatic outcome, not a separate task.

Level 3 — search and compare (attested) The machine is building comparison lists, sorting by features, evaluating options.

MX requires JSON-LD microdata at the pricing level, explicit comparison attributes (product features, specifications), and semantic HTML that machines can parse for feature extraction.

Without Level 3, the machine cannot understand what you offer or how you compare. You risk being invisible in the shortlist. The improvement step: JSON-LD microdata, explicit comparison attributes, cryptographic attestation.

We implement MX patterns for machine comparison tasks. Structured data benefits multiple disciplines automatically.

Level 4 — price understanding (registered) The machine needs exact pricing to make recommendations.

MX requires Schema.org types (Product, Offer, PriceSpecification), unambiguous pricing structure with currency specification (ISO 4217 codes), validation to prevent decimal formatting errors, and clear price markup that prevents magnitude misinterpretation.

Without Level 4, machines misunderstand costs by orders of magnitude. Ambiguous decimal formatting produces the kind of pricing errors described earlier in this chapter, and cross-region pricing confusion leaves buyers unable to determine actual delivered cost. The improvement step: Schema.org Product/Offer/PriceSpecification, ISO 4217 currency codes, Reginald registration.

We implement MX patterns for machine price parsing. Schema.org benefits multiple disciplines automatically.

Level 5 — purchase confidence (audited) Can the machine complete checkout with confidence?

MX requires no hidden state buried in JavaScript (state must be DOM-reflected), explicit form semantics (`<button>` not `<div class="btn">`), persistent feedback (role="alert" for important messages), data-state attributes for checkout progress tracking, and UCP (Universal Commerce Protocol) support for standardised commerce interactions.

Without Level 5, the commerce chain is at risk. The machine cannot see what buttons do, cannot track checkout progress, and may time out and abandon the cart. The improvement step: DOM-reflected state, explicit form semantics, persistent feedback, third-party verification.

We implement MX patterns for machine checkout completion. Accessibility and UX improvements are automatic outcomes.

The cumulative chain Each level builds on the one below. Level 1 requires semantic HTML. Level 2 requires structured data. Level 3 requires JSON-LD and attestation. Level 4 requires Schema.org and registry presence. Level 5 requires explicit state and third-party audit. Skip a level and everything above it is unreachable — the improvement path is sequential.

Sites that complete the full progression become trusted sources — machines tend to return repeatedly through learned behaviour. Sites that stop short risk being bypassed. Unlike humans who persist through bad UX and can be won back with improvements, or a phone call to the support team, machines are unlikely to offer a second chance. You may never see them leave. The business case for progressing through every applicable level is straightforward: each level you reach unlocks the next agent outcome.

Implement — fix the source, not the model

The obstacles between you and that destination are clear: machines cannot see your CSS, cannot parse unstructured content, cannot complete checkout, and you cannot detect which machine is visiting. Each obstacle has a corresponding solution, and each solution removes the obstacle permanently.

The principle from Part A applies here in technical terms: MX eliminates the gap between what machines need and what your HTML provides. You fix the source, not the models. Close the gap, and hallucination has nowhere to hide.

MX is the practice; HTML is the delivery mechanism. This distinction matters. We're not just making HTML adjustments. MX is the strategic discipline that ensures complete context reaches machines. HTML is simply the universal format that makes this possible.

Understanding metadata

What is metadata? Data about data — structured information that describes other information, a bit like footnotes for a document:

- HTML documents have `<meta>` tags (description, keywords, author)
- Markdown files can have YAML frontmatter (title, date, tags)
- JPEG images contain EXIF data (camera model, location, timestamp)
- Git commits have metadata (author, date, message, parent commits)
- Bash scripts have comments at the top (purpose, usage, author)

Many people don't realise these are all the same concept — structured information that describes the thing it's attached to.

MX is metadata-agnostic. Schema.org is the most visible metadata vocabulary on the web — but it is one layer in a stack. MX relies on existing standards wherever possible, amplifying what the web already provides. Where gaps remain (lifecycle status, content governance, agentic access policy), MX adds its own light-touch conventions, particularly through YAML frontmatter and `.mx.yaml.md` sidecar files. Each standard below solves a different problem at a different scale:

- **Schema.org / JSON-LD** — structured data for products, organisations, events, reviews
- **RDFa** — embedded semantic markup within HTML attributes
- **Open Graph / Twitter Cards** — social sharing and preview metadata
- **Dublin Core** — document-level descriptors (creator, subject, rights)
- **EXIF / IPTC / XMP** — image and media metadata (camera, location, copyright)
- **YAML frontmatter** — document properties in markdown and static-site generators
- **HTML `<meta>` tags** — page-level description, author, viewport, robots directives
- **`robots.txt` / `sitemap.xml` / `llms.txt`** — crawl directives and site structure
- **`<link rel>` tags** — canonical URLs, alternate languages, licensing

Each of these standards solves a real problem, and each is already understood by some subset of machines and search engines. MX treats them all as tools in the same toolkit. The MX practitioner's job is to ensure the right metadata reaches machines in the right format, not to replace what works, but to fill the gaps where existing standards leave context incomplete.

The result is strategic redundancy: the same fact expressed through multiple standards so that machines of varying capability can each find what they need.

MX's principle is straightforward: use the appropriate metadata format for each context.

- For scripts: Comments with usage examples
- For markdown: YAML frontmatter with document properties
- For folders: `.mx.yaml.md` files with purpose and relationships
- For formats without metadata support (like `.exe`): Create a sidecar file (`.mx.report.exe.yaml`)

The `.mx` prefix is consistent — all MX metadata files start with `.mx`, making them instantly recognizable as "meta meta" (metadata about the metadata system itself).

As the case studies earlier in this chapter demonstrate, without proper metadata, machines cannot distinguish context from content. A pricing algorithm cannot tell a disaster from a demand spike, and an AI assistant cannot distinguish a product that is unavailable from one that simply lacks structured availability data. Proper microdata and metadata prevent these errors by providing complete context. However, retrofitting the entire internet isn't feasible. The real value is in going forward — properly structured pages help with live web search, improving goal completion (sales, contact forms, information retrieval) that might otherwise be missed.

This isn't a new concept. RDF (Resource Description Framework) was proposed many years ago, but not everyone listened. Now there's a commercial imperative driving adoption through Machine Experience requirements: when machines can't complete your desired action (purchase, contact, inform), they may move to competitors who provide explicit structure.

One concern with current practices: the trend of converting web pages to markdown before sending them to machines. This process strips all metadata — pricing information, geographical context, document type indicators. It's dangerous. Metadata preservation in machine-readable formats is essential for accurate information extraction and preventing hallucination.

Each MX pattern removes a specific obstacle standing between machines and your content:

- Machines cannot see CSS layout → Use semantic HTML (`<main>`, `<nav>`, `<article>`, `<button>`)
- They hallucinate when context is missing → Provide structured data (Schema.org JSON-LD)
- They miss JavaScript state → Make state explicit (`data-state`, `aria-invalid`, `role="alert"`)
- They lose navigation in flat content → Use clear heading hierarchy (h1 → h2 → h3)
- They skip non-text content → Add text alternatives (alt text, aria-label)

Remove these obstacles, and one MX implementation serves three audiences:

1. Machines (ChatGPT, Copilot, Perplexity, crawlers, voice assistants) — primary focus
2. Search engines (Google, Bing) — side benefit
3. Users with disabilities (screen readers, keyboard users) — side benefit

MX in the content pipeline

MX is often confused with adjacent disciplines in the content stack. MX is not a Content Management System (where content is created and stored), not a Content Delivery System (infrastructure for serving content), and not an ontology (semantic model of concepts and relationships). MX is the practice that ensures context gets through to the goal of the site. HTML, informed by MX, is the publication point.

Consider the content pipeline: Content Operations builds semantic structure at the construction point, MX ensures that structure survives publication, and Content Delivery serves it to machines. Without MX, well-structured content with rich metadata in the CMS becomes stripped of metadata at delivery — machines cannot understand context.

Most organisations have sophisticated Content Operations and reliable Content Delivery. The gap is at publication, where CMS-rich metadata gets stripped during HTML rendering. MX closes this gap by governing what reaches the delivery point. The implication for CMS vendors is stark: the CMS that serves machine-readable content survives. The one that does not gets bypassed.[14]

MX is not a tool or platform. It is the practice, the governance layer, that ensures publication preserves structure. And it extends beyond HTML publication. On file systems, `.mx.yaml.md` sidecar files give folders and directories machine-readable context. In documents, YAML frontmatter carries lifecycle status, audience declarations, and governance fields. Self-describing files — cogs — combine structured metadata with human-readable content in a single portable format. The web is where MX is most commercially urgent, but the practice applies wherever machines read content. Implementation requires:

- Development standards (semantic HTML patterns, accessibility compliance)
- Template requirements (Schema.org integration, meta tag stacks, Open Graph)
- Content governance (YAML frontmatter, `.mx.yaml.md` metadata, lifecycle management)
- QA validation (machine compatibility testing across metadata layers)
- Organisational roles (MX specialists who bridge content and development)

MX operating system — metadata everywhere The core philosophy of MX is simple: place metadata everywhere. Every folder gets context, every document gets recovery information, every workflow gets executable instructions. The implementation uses `.mx.yaml.md` files, dot-prefixed to stay hidden from humans (`ls` shows only work files), but visible to machines (`ls -a`). The `.md` extension means humans who investigate get readable prose; machines get structured YAML in the same file. One file, two audiences.

This extends beyond code repositories. MX practitioners add `.mx.yaml.md` files across their entire filesystem: `~/Documents/Projects/`, `~/Documents/Invoices/`, even `~/Downloads/`. Each file contains what exists, why, how it was created, and tentative prompts for regeneration. If a document gets lost, a machine can recreate it from the metadata alone. This transforms the file system itself into an agentic platform, the MX OS.

MX in practice — a real workflow case study Consider a concrete task: adding a new repository to a monorepo with full MX integration. Without metadata, a machine spends many minutes exploring randomly, inferring patterns, and missing steps. With MX metadata, the machine navigates directly to `.gitmodules`, `ONBOARDING.md`, `.mx.yaml.md` files, and the onboarding script, completing investigation in a few minutes.

The machine then executed an 8-step plan (validate, add submodule, initialise, onboard with MX metadata, validate, commit, push, verify) in a few minutes with zero errors. Rather than stopping at task completion, it codified the pattern: a bash script, a user guide, and preserved plan files. The next time this task is needed:

```
npm run repo:add <url>
```

Answer two questions, confirm, done. No AI needed. The economics tell the story: $0.50 in initial API calls, $0.00 for every future use. Break-even point: second use.

This demonstrates the core MX principles in action: explicit over implicit (structured plan replaces tribal knowledge), designing for multiple audiences (machines read script structure, humans read documentation, automation reads exit codes), and progressive disclosure (simple command for common use, manual steps available when needed). The chapter itself demonstrates these principles. Its frontmatter IS metadata, making its provenance machine-readable.[15]

Entity Asset Layer

The obstacles described above — platform lock-in, proprietary data formats, assets trapped in vendor databases — point to a deeper strategic problem. Your organisation's most valuable digital assets (reviews, product knowledge, customer relationships, operational data) must remain sovereign and portable across any platform or machine.

The Entity Asset Layer (EAL) is the strategic framework that addresses this. It categorises your digital assets into four types — Identity, Reputation, Knowledge, and Transactional — and ensures each remains portable through standard formats (Schema.org JSON-LD, semantic HTML, structured metadata). Platforms become interchangeable view layers rather than the source of truth for your business data.

MX is the technical practice that makes EAL possible. MX governs *how* data is published; EAL governs *what* data is published. Without MX patterns preserving structure at publication, Entity Assets stay trapped in platform-specific formats.

The full EAL treatment — asset taxonomy, sovereign portability architecture, platform lock-in diagnosis, and implementation patterns — is covered in the implementation chapters later in this book.

Two halves of the agentic web

The industry is converging on two complementary standards for how machines interact with the web, and understanding both is essential.

The first half is content understanding: making what's on the page machine-readable. This is what MX provides. Structured metadata, semantic HTML, explicit context. The machine reads your content and knows what it means without guessing.

The second half is action execution: giving machines structured ways to interact with page functionality. This is what WebMCP (Web Model Context Protocol) provides.[16] Launched by Google and Microsoft in February 2026 as a W3C draft standard, WebMCP lets websites expose callable tools (search functions, booking forms, checkout flows) directly through a browser API. Instead of scraping DOM elements and guessing which button does what, a system can call a registered `searchProducts(query, filters)` tool directly and receive structured results.

WebMCP solves a real problem. It only solves half the problem, though. A machine calling `getDresses()` receives product data, but still has to reason about that data to decide which dresses are eco-friendly, which are appropriate for a wedding, which match the user's preferences. That reasoning is expensive and unreliable when the data lacks structured metadata.

With MX, the reasoning becomes lookup. Tags, categories, attributes, all explicit in the metadata. The machine doesn't need to infer that a dish is gluten-free by parsing a description paragraph. The metadata says so.

WebMCP gives machines hands. MX gives them eyes. One lets them act. The other lets them understand. Together they form the companion web, the machine-readable layer alongside the human web. Neither works alone.

Measure — track and sustain

UX/MX — a new discipline

Just as we have UX (User Experience), we need MX (Machine Experience). The central insight is that organisations should design for both humans and machines simultaneously with the same content — not create separate experiences.

Adobe recently released an "LLMoptimiser" tool that detects whether a page is being browsed by an LLM or a human, then serves different versions accordingly.[17] This approach carries a fundamental risk. Google's webmaster guidelines explicitly penalise cloaking — serving different content to bots than to human visitors.[18] A site that serves one page to Googlebot and another to users risks manual penalties and removal from search results. The same principle applies to machines: if you can make content better for machines, you make it better for disabled people and for everyone else. The solution is not to serve different content but to fix the underlying web page.

MX requires dedicated organisational roles, just as quality requires dedicated QA engineers. When MX is "everyone's responsibility" without designated ownership, accountability evaporates — the same pattern observed in accessibility programmes and DevOps implementations.[19] Successful organisations assign clear MX accountability: dedicated specialists who define machine-readability standards and audit implementations, distributed awareness across development and content teams, and executive ownership that provides resources and tracks MX metrics alongside traditional performance indicators.[20] The implementation chapters cover organisational models in detail.

The Gathering

MX relies on existing web standards wherever possible, but the gaps between those standards need governance. Who decides what metadata fields are required? How should machine discovery files work? What conventions ensure interoperability across platforms?

The Gathering is an independent, community-governed standards body for Machine Experience. It develops open standards that enable content to be understood, interpreted, and processed more consistently by both people and AI systems. The standard is open. No fees. No gatekeeping. Anyone can read drafts and track progress.[21]

Standards move through a four-stage process: **Propose** (a contributor outlines the problem, defines scope and non-goals), **Review** (the community provides feedback, editors maintain consistency), **Call for Consensus** (a defined review window checks for unresolved objections), and **Ratify** (the draft is published as a versioned standard with full history preserved). All drafts, discussions, review notes, and decision records are hosted publicly on Stream, so participants can understand how each standard formed. Contributors include writers, editors, developers, content designers, and researchers, the practitioners who build the web machines consume.

The Gathering addresses gaps that existing standards leave unresolved: defining what metadata is required, how access is controlled, how quality is measured, and how governance works. The llms.txt training-and-inference gap described earlier in this chapter is one example: a critical file format that falls between both access mechanisms because it is neither HTML (for training crawlers) nor targeted enough for inference-time queries. The Gathering is proposing a new standard to address this lack, so that machine discovery files are properly ingested by both mechanisms.

The relationship is straightforward: MX is the practice. The Gathering is the governance. Like Linux running on POSIX, a community-owned standard, with products built on it.

What’s next

This chapter covered the business case, the technical reality, and the structural solution. You’ve seen what machines need, why they fail without it, and what MX implementation looks like in practice.

The remaining chapters cover practical implementation: platform-specific patterns, organisational change management (including the dedicated roles and accountability structures that MX requires), and the technical architecture that makes your web presence machine-ready. The rest of this book shows you how to build it.

For professional website analysis, the MX Web Audit Suite assesses machine-readiness and provides priority-based recommendations. More information is available at `https://allabout.network`.

Cogs — self-describing documents

The MX patterns in this chapter apply to websites. The same principle applies to any file. A cog is any file that carries structured MX metadata. The most common form is a markdown file with YAML frontmatter — structured metadata for machines at the top, human-readable content below. Cogs are not limited to markdown, though. An HTML page becomes a cog when it carries `<meta name="mx:*">` tags. A JavaScript file becomes a cog when its JSDoc comments include `@mx:*` tags. A shell script becomes a cog when it carries MX metadata in commented YAML. A CSS file becomes a cog when its comment block includes `@mx:*` tags. The carrier format changes to match the file type. The principle stays the same: metadata that machines can parse without reading the content. No tooling required. No registration. No infrastructure beyond a text editor.

Cogs work wherever files work. A cog on your laptop is useful to your local AI tools. A cog on a company file server is useful to your whole team. A cog in a git repository is versioned, reviewed, and travels with the code. None of these need a registry, a server, or any external infrastructure. The format is governed by The Gathering as an open standard (MIT licensed), and it stands entirely on its own.

When cogs need to cross organisational boundaries — when a publisher needs to prove a file is genuine, when machines outside your organisation need to discover your documentation, when living documents need version tracking — that is where Reginald comes in. Reginald is a public registry that adds discovery, cryptographic attestation, and currency tracking. Registration is optional. Most cogs will never need it. A cog on a file server is already doing its job. Reginald is there when trust and discoverability demand it.

The cog format and the Reginald registry are covered in depth in the companion volumes.

CogNovaMX Ltd — Making the web work for everyone and everything that uses it.

Notes

[1] Adobe Holiday 2025 data: AI referrals to retail surged 693% year-over-year, travel 539%. AI-referred visitors converted 31% higher than other traffic and were 33% less likely to bounce. Based on over 1 trillion visits to U.S. retail sites.
`https://business.adobe.com/blog/ai-driven-traffic-surges-across-industries`

[2] Four machine commerce platforms launched within eight days in January 2026: Amazon Alexa+ at CES (5 January), Microsoft Copilot Checkout at NRF with PayPal and Shopify (8 January), Google Universal Commerce Protocol at NRF with Target and Walmart (11 January), and Anthropic Claude Cowork (12 January).
`https://techcrunch.com/2026/01/05/alexa-without-an-echo-amazons-ai-chatbot-comes-to-the-web-and-a-revamped-alexa-app/`
`https://newsroom.paypal-corp.com/2026-01-08-PayPal-Powers-Microsofts-Launch-of-Copilot-Checkout`
`https://techcrunch.com/2026/01/11/google-announces-a-new-protocol-to-facilitate-commerce-using-ai-agents/`

[3] The 2026 Adamuz train derailments killed 46 and injured 292. Spain’s worst rail disaster since Santiago de Compostela (2013). Iberia capped fares at €99 and added flights. Spain’s national ombudsman investigated transport companies. Consumer association FACUA called for legal reform.
`https://en.wikipedia.org/wiki/2026_Adamuz_train_derailments`

https://www.euronews.com/travel/2026/01/21/sold-out-buses-and-sky-high-flight-prices-spains-train-crash-leaves-passengers-stranded

[4] AgentLock — open authorization standard for AI agents. Addresses what the framework calls the "Full Permission anti-pattern": AI agents operating without the security controls that traditional computing (Unix permissions, database GRANT/REVOKE, cloud IAM) has required for decades. Apache 2.0 licensed.
https://agentlock.dev/

[5] Peter Zaffina, enterprise technology leader specialising in AI, data, and cloud transformation. The framing of the data project as prerequisite for the AI project draws on his argument that governance must precede deployment.
https://www.linkedin.com/in/peter-zaffina/

[6] Tom Cranstoun, "A CMS Consultant's Takeaways from CMS Kickoff 2024," CMS Critic, February 2024. The article argued that AI's real value lies in consuming content, not generating it, and proposed treating "Machine" as a fourth device type alongside mobile, tablet, and desktop.
https://cmscritic.com/a-cms-consultants-takeaways-from-cms-kickoff-2024

[7] Anthropic launched Claude Cowork in January 2026 — a shift from chatbot to autonomous digital colleague. The system manages local file systems, orchestrates workflows, and executes complex tasks through multi-agent architecture. See:
https://venturebeat.com/technology/anthropic-launches-cowork-a-claude-desktop-agent-that-works-in-your-files-no
https://markets.financialcontent.com/stocks/article/tokenring-2026-1-19-anthropic-unveils-claude-cowork-the-first-truly-autonomous-digital-colleague
https://techcrunch.com/2026/01/12/anthropics-new-cowork-tool-offers-claude-code-without-the-code/

[8] ISO 4217 defines three-letter currency codes (EUR, INR, GBP, USD) that eliminate currency identification ambiguity. Schema.org's price specification requires numeric values to use a period as the decimal separator with no thousands grouping — whether expressed as a Microdata `content` attribute or a JSON-LD `"price"` property. This convention, aligned with ISO 80000-1 (Quantities and units), eliminates all cultural formatting ambiguity at the data layer.
https://www.iso.org/iso-4217-currency-codes.html
https://schema.org/price

[9] Common Crawl language distribution statistics. Language identification via Compact Language Detector 2 (CLD2) across monthly crawl archives. English consistently represents approximately 44–46% of crawled content.
https://commoncrawl.github.io/cc-crawl-statistics/plots/languages

[10] For deeper exploration of AI statistical foundations and linguistic bias, see my blog posts:
https://allabout.network/blogs/ddt/ai/the-stripped-down-truth-how-ai-actually-works-without-the-fancy-talk
https://allabout.network/blogs/ddt/ai/does-ai-mean-algorithmic-interpolation
https://allabout.network/blogs/ddt/ai/the-digital-language-caste-system
https://allabout.network/blogs/ddt/ai/the-mathematical-heartbeat-of-ai
https://allabout.network/blogs/ddt/ai/the-tokenization-trap-how-ai-actually-processes-german
https://allabout.network/blogs/ddt/ai/the-no-elephants-problem-why-ai-struggles-with-what-not-to-do
https://allabout.network/blogs/ddt/ai/when-a-five-year-old-beats-an-ai-at-its-own-game
https://allabout.network/blogs/ddt/ai/a-framework-for-evaluating-ai-confidence

https://www.simplypsychology.org/bystander-effect.html
https://thedecisionlab.com/reference-guide/psychology/diffusion-of-responsibility

[11] For a practitioner's account of how these principles reshape day-to-day building practice.
https://allabout.network/mx/principles-changed-how-i-build.html

[12] Hugging Face is the largest open-source AI model repository and community platform. Founded in 2016, it hosts models, datasets, and inference tools that anyone can use. Its significance for MX is that it represents the long tail of AI capability — the majority of models hosted there are small, limited, and incapable of inferring meaning from poorly structured content.
https://huggingface.co

[13] Hugging Face model growth data: model tracking began in March 2022. The first million models took over 1,000 days; the second million arrived just 335 days later. By August 2025, models added that year had already surpassed the entire 2024 total.
https://aiworld.eu/story/hugging-faces-two-million-models-and-counting

[14] For the full argument that the CMS is shifting from destination to invisible workflow.
https://allabout.network/is-the-cms-a-destination-or-a-workflow

[15] This chapter was co-authored with Claude (Anthropic). The frontmatter metadata, the structured arguments, and the MX patterns described in the text are all implemented in the chapter itself — the medium demonstrates the message.

[16] WebMCP (Web Model Context Protocol) — W3C Draft Community Group Report, 10 February 2026. Created by engineers at Google and Microsoft under the W3C Web Machine Learning Community Group. Early preview in Chrome 146 Canary.
`https://webmachinelearning.github.io/webmcp/`
`https://venturebeat.com/infrastructure/google-chrome-ships-webmcp-in-early-preview-turning-every-website-into-a`

[17] Adobe LLM Optimiser — edge-based deployment that detects agentic traffic and serves AI-friendly content modifications at the CDN layer. Targets only agentic requests; does not affect human users or SEO bots.
`https://business.adobe.com/products/llm-optimiser.html`
`https://business.adobe.com/blog/introducing-adobe-llm-optimiser`

[18] Google defines cloaking as "presenting different content to users than to search engines" and classifies it as a violation of spam policies. Sites caught cloaking face manual actions including removal from search results.
`https://developers.google.com/search/docs/essentials/spam-policies#cloaking`

[19] DevOps implementations demonstrate the balance between shared responsibility and specialised roles. Whilst everyone takes responsibility for quality, dedicated QA engineers focus on defining standards, designing test frameworks, and preventing bugs. See:
`https://www.qamadness.com/devops-and-the-role-of-qa/`
`https://www.splunk.com/en_us/blog/learn/devops-roles-responsibilities.html`

[20] Research shows 82% of respondents have limited ability to hold others accountable, and only 14% of employees feel their performance is managed in ways that inspire responsibility. Clear ownership structures address this gap. See:
`https://www.forrestadvisors.com/insights/organizational-design/accountability-organizational-design-fostering-responsibility/`
https://medium.com/@csw11235/responsibility-accountability-and-ownership-da054169fcce

[21] The Gathering — open standards for Machine Experience. Community-governed, MIT licensed, practitioner-led.
`https://tg.community`
`https://stream.tg.community`
`https://tg.community/process`

The AI Tipping Point: A Consultant's Takeaways from CMS Kickoff 2024

Tom Cranstoun

March 2026

Originally published at

`https://cmscritic.com/a-cms-consultants-takeaways-from-cms-kickoff-2024`

The AI Tipping Point: A Consultant's Takeaways from CMS Kickoff 2024

Tom Cranstoun · Principal Consultant, Digital Domain Technologies Limited

---

As a first-time attendee at the Boye & Company event, I didn't know what to expect. But the speakers and content left me with a deeper understanding of the future of CMS – and how AI is changing everything.

Tom Cranstoun is the principal consultant at Digital Domain Technologies Limited and a CMS Critic contributor.

Reflecting on the Boye & Co CMS Kickoff 2024 down in St. Pete Beach, it's clear we're at a tipping point with AI in the CMS space. The event was a melting pot of CMS enthusiasts and experts, all there to predict and shape how we handle CMS. The setup was well-suited for learning, networking, and bouncing ideas off one another, with sessions covering a wide range of topics.

We had some standout talks, like Deane Barker's Q&A and Kristina Podnar's insights into Generative AI. Becky Brown and David Rasmussen hit home the importance of change management for content teams. Mike Wills' take on flexible content models for multichannel strategies was enlightening, and Jeff Eaton shed light on how CMS page builders affect team dynamics.

Brian Browning showed us practical AI applications that are changing digital experiences. Debbie Tucek pushed us to think of content as a product, while Ricky Frohnerath's masterclass on high-performance content teams was a game-changer. Nick Rudd's perspective on AI in real-time content generation was eye-opening, and sessions by Marli Mesibov and Brian McKeiver on the CMS buy vs. build dilemma and digital commerce were thought-provoking.

The twist in my expectations

I noted that the agenda of the CMS Kickoff 2024 offered only a *slight* focus on AI. The speakers often mentioned it during their talks; it's very "of the moment" in our industry.

I, like you, have been to many seminars, presentations, and conferences around AI and what it will do for you and your content. I was pleasantly surprised this was not about AI *creating* the content – it was about AI *reading* the content. It turned my expectations upside down, fired up my brain, and I could not stop taking notes.

The whole CMS Kickoff conference was not just about AI. I have many takeaways that I am working on. But the most important is this: *the AI revolution will change our industry and everyone else's.* Still, despite its great promise and clear benefits, it soon becomes clear that using AI to create content is not ideal. In fact, AI-generated content has many problems, including:

- Enforcing the use of internal style guides.
- Maintaining a brand tone of voice.
- Avoiding gender or race biases.
- Avoiding cultural nuances.
- Knowing which sources were used in the training of the AI (too few sources means limited vocabulary – conversely, too many sources suggest the corporation loses control. Does the AI manage to use local idiomatic expressions? For instance, a "sidewalk" is a "pavement" in the UK).
- AI sometimes gives up asking the prompter to continue.
- AI is becoming regulated, and legal ramifications will soon become commonplace, changing in each jurisdiction.

And this is just the beginning.

Rethinking AI's role in content management

Throughout the event, the speakers made me rethink AI's role in content management. Given the translation, bias, regulation challenges, and lack of trust in AI, one needs to employ reviewers and editors to correct the content – because AI is better suited for *consuming* content.

Let's think that the current state of AI creating content is a beta experiment, best left to the tech guys. After all, we have a business to think of.

Nick Rudd led an engaging discussion explaining how AI should be thought of. He began by asking how an eight-year-old purchases toys. He described it as simple and direct, without the baggage of brand loyalty or complex decision-making.

Kids are smart. They know what they want.

Ignoring the elephant in the room, "TikTok" (an entirely different article), the eight-year-old goes to Google/Bing/Whatever and types in:

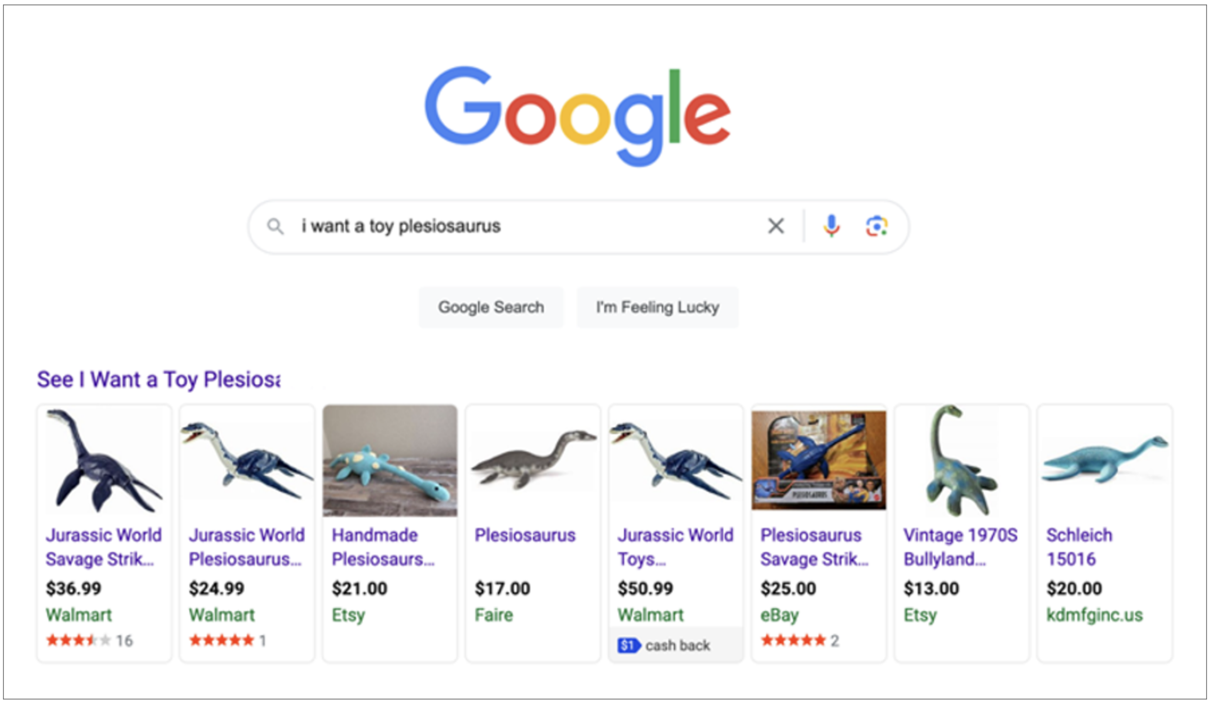
"I WANT A TOY PLESIOSAURS"

What didn't the eight-year-old do?

- They didn't go to Amazon.
- They didn't refine the search.
- They didn't look at brands.
- They didn't read customer reviews.
- They didn't sort by price or popularity or whatever.
- They didn't read the price (well… mostly).
- They didn't create an account.
- They didn't click on dropdowns.
- They ignored video and animations.
- They didn't read the copy.

What happens when AI reads content? It behaves like an eight-year-old:

- It will ignore retailers.
- It will probably not refine the search.



i want a toy plesiosaurus

See I Want a Toy Plesios

Jurassic World Savage Strik...	Jurassic World Plesiosaurus...	Handmade Plesiosaurs...	Plesiosaurus	Jurassic World Toys...	Plesiosaurus Savage Strik...	Vintage 1970S Bullyland...	Schleich 15016
$36.99	$24.99	$21.00	$17.00	$50.99	$25.00	$13.00	$20.00
Walmart	Walmart	Etsy	Faire	Walmart	eBay	Etsy	kdmfginc.us
★★★½ 16	★★★★★ 1			$1 cash back	★★★★★ 2		

Search results for "I WANT A TOY PLESIOSAURS"

- It will ignore brands.
- It might skim customer reviews.
- It might sort by price.
- It won't create an account.
- It won't follow your dropdowns.
- It won't give you an email address (you no longer know *who* your customers are).
- It will ignore the videos and animations.
- It will ignore the "see our shop links".
- It will skim-read the copy if it can find it among the ads.

How will "AI simplicity" affect your business?

The idea of simplicity will challenge how AI interacts with our current web designs and content structures.

The next step in AI monetisation is that the AI offers to *purchase* the toy – keeping any discounts, commissions, and recommendation fees bypassing the retailer. This will impact KPIs as AI activity increases and human interaction decreases.

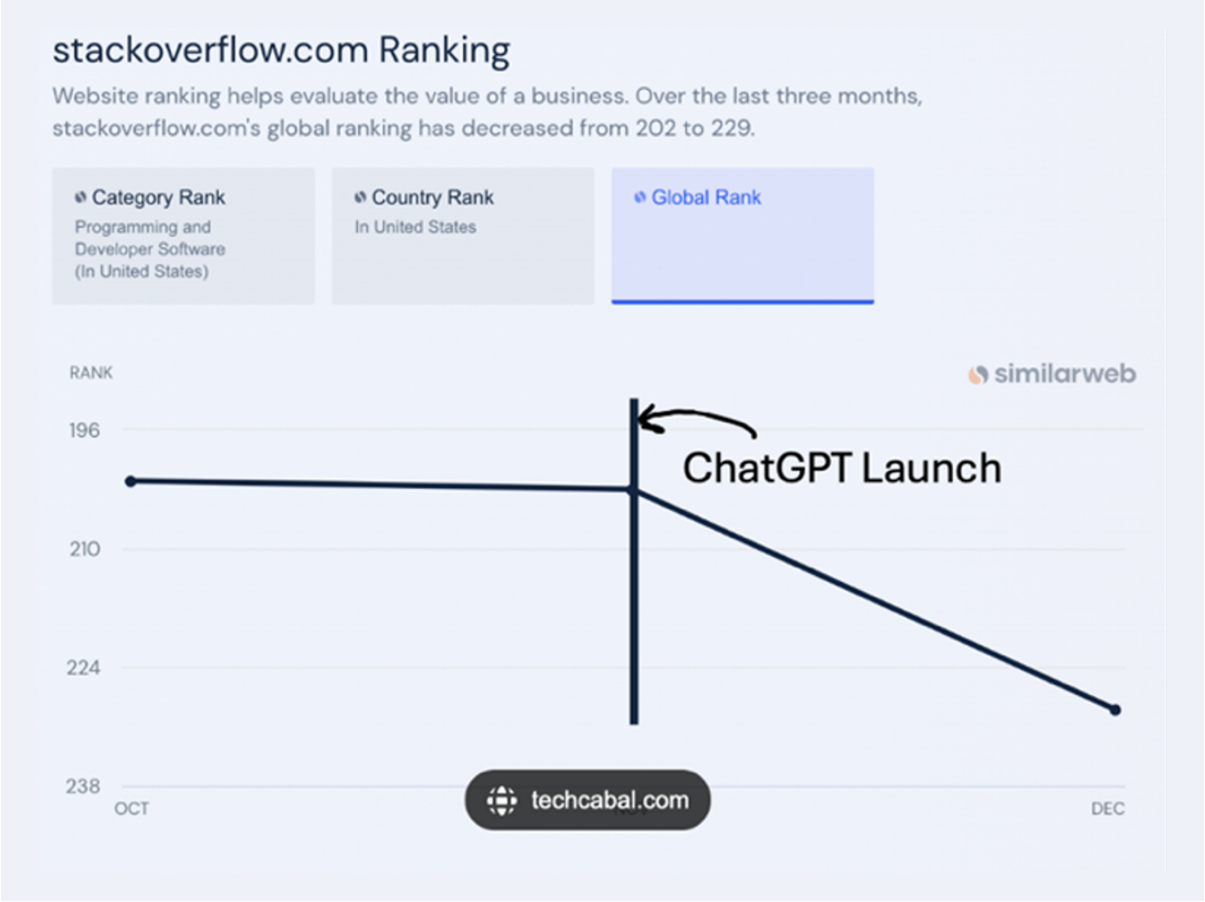
What happened to StackOverflow recently will begin to affect us all. The online Q&A platform for developers is releasing 28% of its staff to save costs, which is not necessarily related to AI. But look at the point where ChatGPT entered the picture:

What about a site's KPIs?

As AI interaction increases, our reliance on traditional KPIs must fall.

Every website today uses analytics to drive statistics, such as:

- The number of total visits.
- The number of unique visitors.
- The number of new vs returning visitors.
- Geographical location of visitors.
- Net promoter score.
- Pages viewed per session.
- Dwell time.



stackoverflow.com Ranking

Website ranking helps evaluate the value of a business. Over the last three months, stackoverflow.com's global ranking has decreased from 202 to 229.

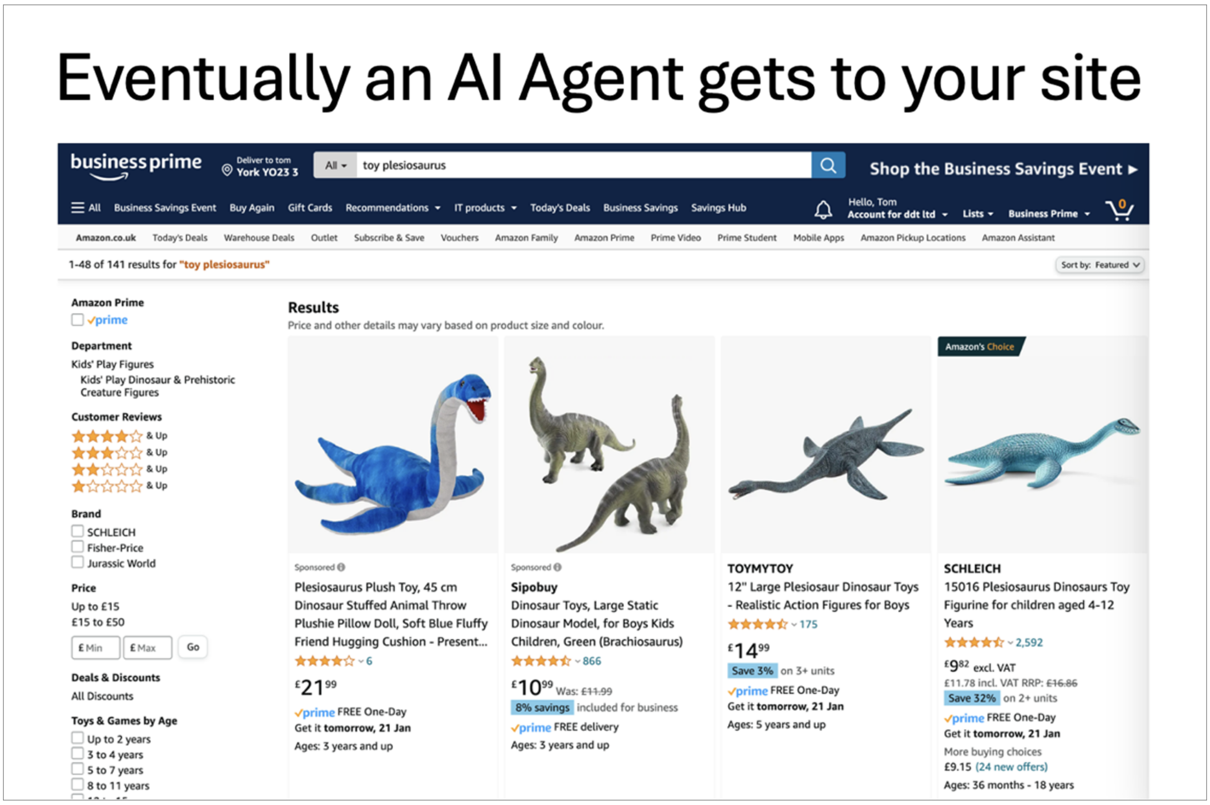
StackOverflow traffic graph showing decline after ChatGPT launch

- Bounce rate.
- Conversions, including A/B testing.
- Organic search visitors.
- The number of social share impressions.
- Feedback and reviews.
- Cart abandonment.

Design by engineers: a trap

The conference also highlighted a common pitfall: websites designed by engineers focus on *looks* over *function*, which does not work well for AI. We need a more structured approach to content modelling for better clarity and organisation.

Today's HTML markup makes sense to the browser:



Eventually an AI Agent gets to your site

1-48 of 141 results for "toy plesiosaurus"

Results

Price and other details may vary based on product size and colour.

Sponsored

Plesiosaurus Plush Toy, 45 cm Dinosaur Stuffed Animal Throw Plushie Pillow Doll, Soft Blue Fluffy Friend Hugging Cushion - Present...

6

£21^{99}

prime FREE One-Day

Get it **tomorrow, 21 Jan**

Ages: 3 years and up

Sponsored

Sipobuy

Dinosaur Toys, Large Static Dinosaur Model, for Boys Kids Children, Green (Brachiosaurus)

866

£10^{99} Was: ~~£11.99~~

8% savings included for business

prime FREE delivery

Ages: 3 years and up

TOYMYTOY

12" Large Plesiosaur Dinosaur Toys - Realistic Action Figures for Boys

175

£14^{99}

Save 3% on 3+ units

prime FREE One-Day

Get it **tomorrow, 21 Jan**

Ages: 5 years and up

Amazon's Choice

SCHLEICH

15016 Plesiosaurus Dinosaurs Toy Figurine for children aged 4-12 Years

2,592

£9^{82} excl. VAT

£11.78 incl. VAT RRP: ~~£16.86~~

Save 32% on 2+ units

prime FREE One-Day

Get it **tomorrow, 21 Jan**

More buying choices

£9.15 (24 new offers)

Ages: 36 months - 18 years

Clean product page rendered in browser

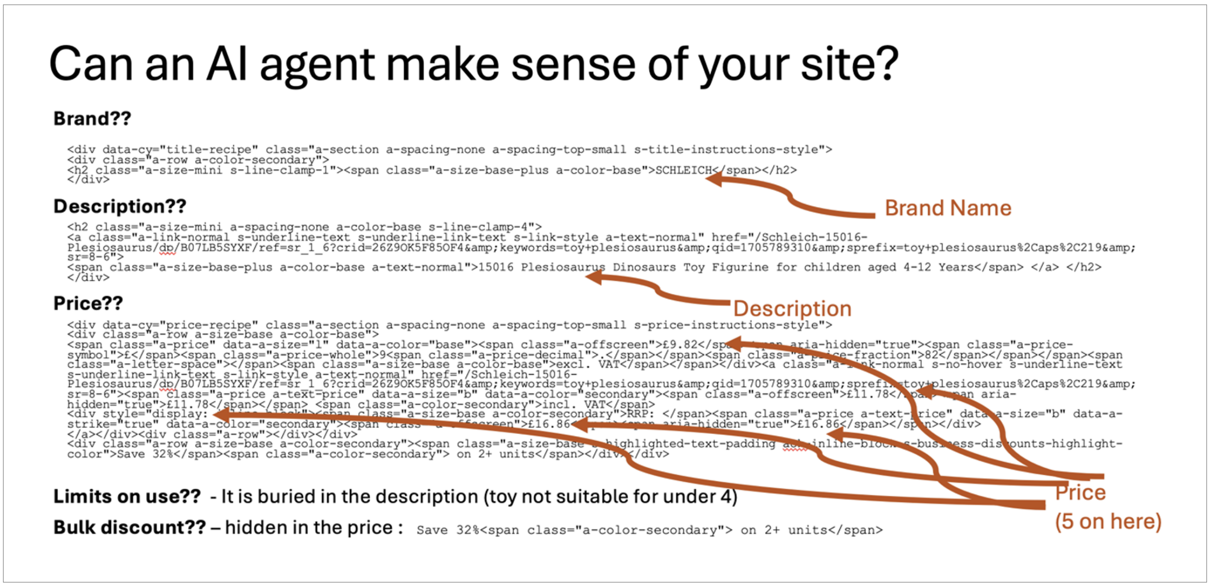
However, opening the "View Page Source" shows a different take:

Buried in the HTML are the brand name, the price (more than one is shown in the HTML), and the description. Note that discounts and age suitability are placed in the text. A human reading this would have difficulty, never mind an AI.

Engineers create text and image boxes in your CMS; they let you fill and serve them.

They might call them hero's carousels, cards, etc. They are dressed up text and image boxes with interactions.

This is the *Design by Engineer Trap.*



Can an AI agent make sense of your site?

Brand??

```
<div data-cy="title-recipe" class="a-section a-spacing-none a-spacing-top-small s-title-instructions-style">
<div class="a-row a-color-secondary">
<h2 class="a-size-mini s-line-clamp-1"><span class="a-size-base-plus a-color-base">SCHLEICH</span></h2>
</div>
```

Description??

```
<h2 class="a-size-mini a-spacing-none a-color-base s-line-clamp-4">
<a class="a-link-normal s-underline-text s-underline-link-text s-link-style a-text-normal" href="/Schleich-15016-
Plesiosaurus/dp/B07LB5SYXF/ref=sr_1_6?crid=26Z9OK5F85OF4&amp;keywords=toy+plesiosaurus&amp;qid=1705789310&amp;sprefix=toy+plesiosaurus%2Caps%2C219&amp;
sr=8-6">
<span class="a-size-base-plus a-color-base a-text-normal">15016 Plesiosaurus Dinosaurs Toy Figurine for children aged 4-12 Years</span> </a> </h2>
</div>
```

Price??

```
<div data-cy="price-recipe" class="a-section a-spacing-none a-spacing-top-small s-price-instructions-style">
<div class="a-row a-size-base a-color-base">
<span class="a-price" data-a-size="l" data-a-color="base"><span class="a-offscreen">£9.82</span><span aria-hidden="true"><span class="a-price-
symbol">£</span><span class="a-price-whole">9<span class="a-price-decimal">.</span></span><span class="a-price-fraction">82</span></span></span><span
class="a-letter-space"></span><span class="a-size-base a-color-base">excl. VAT</span></span></div><a class="a-link-normal s-no-hover s-underline-text
s-underline-link-text s-link-style a-text-normal" href="/Schleich-15016-
Plesiosaurus/dp/B07LB5SYXF/ref=sr_1_6?crid=26Z9OK5F85OF4&amp;keywords=toy+plesiosaurus&amp;qid=1705789310&amp;sprefix=toy+plesiosaurus%2Caps%2C219&amp;
sr=8-6"><span class="a-price a-text-price" data-a-size="b" data-a-color="secondary"><span class="a-offscreen">£11.78</span><span aria-
hidden="true">£11.78</span></span> <span class="a-color-secondary">incl. VAT</span>
<div style="display: inline-block"><span class="a-size-base a-color-secondary">RRP: </span><span class="a-price a-text-price" data-a-size="b" data-a-
strike="true" data-a-color="secondary"><span class="a-offscreen">£16.86</span><span aria-hidden="true">£16.86</span></span></div>
</a></div><div class="a-row"></div></div>
<div class="a-row a-size-base a-color-secondary"><span class="a-size-base a-highlighted-text-padding aok-inline-block s-business-discounts-highlight-
color">Save 32%</span><span class="a-color-secondary"> on 2+ units</span></div></div>
```

Limits on use?? - It is buried in the description (toy not suitable for under 4)

Bulk discount?? – hidden in the price : `Save 32%<span class="a-color-secondary"> on 2+ units</span>`

Messy HTML source code of the same product page

What needs to be added to our sites?

Content Modelling and schema presentation!

A lot has gone wrong. The IT people who made this website are very clever engineers; I could not do this.

The engineers took the site owners' desires and made pretty pictures with text and prices, highlighting discounts. They made it reasonably fast and effective and made the site owner one of the world's wealthiest men.

Look at other sites' HTML. How do they handle pricing? *Awful!*

Each site dresses up text in presentations. No semantic meaning is given to the objects described. There is no content modelling. It looks like AI will need an LLM for each brand!! This is not going to work in the AI world.

Something *major* must happen...

Content modelling the traditional CMS way

You probably already know it: the model gets built into the CMS.

This is a trap.

Designers fixate on the presentation when envisioning a website – and build a style guide with fonts, colours, components, and interactions. They create heroes, cards, lists, articles, carousels, accordions, tabs, text, and much, much more.

What next?

The content team is asked to create:

- **Testimonials.** *Someone decides that testimonials should be in the cards component.*
- **Special offers.** *Someone decides that testimonials should be in the lists component.*
- **A "coming soon" message.** *Someone decides that testimonials should be in the image with a link component.*
- **Bulk discount messages.** *Someone decides that Bulk Discount should be in the text component.*

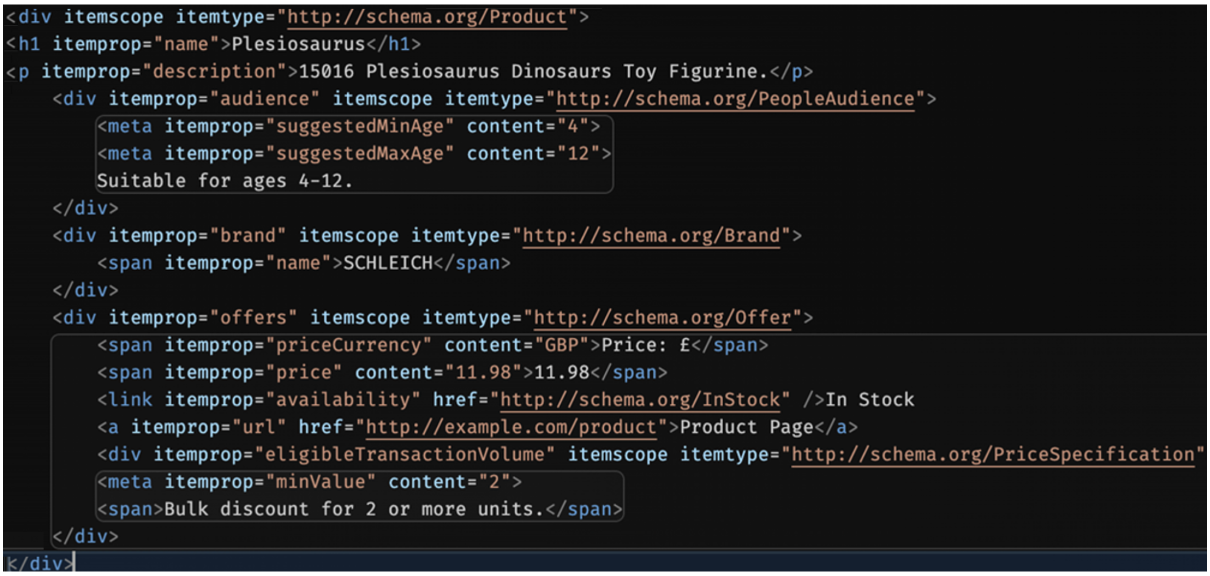
Content modelling by developers!

So, what do we need to do about this?

We need technical improvements, enthusiastic use of content models, and application of schema without putting cognisant overload on the content team – which is a big ask.

I assure you it will be worth it.

Let’s look at the HTML example we showed earlier, this time with schema applied:



```
<div itemscope itemtype="http://schema.org/Product">
<h1 itemprop="name">Plesiosaurus</h1>
<p itemprop="description">15016 Plesiosaurus Dinosaurs Toy Figurine.</p>
    <div itemprop="audience" itemscope itemtype="http://schema.org/PeopleAudience">
        <meta itemprop="suggestedMinAge" content="4">
        <meta itemprop="suggestedMaxAge" content="12">
        Suitable for ages 4-12.
    </div>
    <div itemprop="brand" itemscope itemtype="http://schema.org/Brand">
        <span itemprop="name">SCHLEICH</span>
    </div>
    <div itemprop="offers" itemscope itemtype="http://schema.org/Offer">
        <span itemprop="priceCurrency" content="GBP">Price: £</span>
        <span itemprop="price" content="11.98">11.98</span>
        <link itemprop="availability" href="http://schema.org/InStock" />In Stock
        <a itemprop="url" href="http://example.com/product">Product Page</a>
        <div itemprop="eligibleTransactionVolume" itemscope itemtype="http://schema.org/PriceSpecification"
        <meta itemprop="minValue" content="2">
        <span>Bulk discount for 2 or more units.</span>
    </div>
</div>
```

HTML source code with Schema.org markup applied

Now a machine can see the price, the description, the brand, the discount, and the age applicability. It knows it is for a product, not an ad on the site.

Call to action

We are in marketing, so we must end with a call to action. So here it is:

Our websites need to target four device types: *Mobile*, *Tablet*, *Desktop*, and *Machine* in the future. Our CMS strategy needs a solid content model that applies consistently across all platforms. This is key for targeted ads, personalised content, and campaign management.

However, the current developer-driven content modelling approach must be revised and rethought. We must wrap our content semantically and stick to structured data schemas. Schema.org is the place to start; that is the technical part. We as an industry need to evangelise and build a joint effort from designers, developers, and content creators to ensure content is built within the model and aligns with the established schema.

As such, teams need to add an “AI Evangelist” to the roles.

The AI Evangelist will be a key figure in our organisations, driving the adoption and implementation of AI technologies. This role bridges technical knowledge, strategic insight, and the ability to communicate AI’s benefits to diverse audiences. The AI Evangelist will forge partnerships across design, development, content, and testing teams to craft a unified AI vision.

Key Responsibilities of an AI Evangelist:

- Champion AI technology adoption, showcasing tangible improvements to digital strategies and user experiences.
- Lead AI-centric models and strategies, focusing on predictive analytics, personalised content, and AI-powered search optimisation.
- Facilitate AI integration into workflows with cross-functional teams, ensuring smooth transitions and efficiency.
- Lead educational initiatives like workshops and training to build an AI-aware culture.

- Monitor and assess AI trends for strategic adoption.
- Cultivate a network with AI vendors and thought leaders to build expertise.
- Advise leadership on AI integration, securing necessary resources and investments.
- Examine new opportunities to decrease time-to-market and increase engagement.

Qualifications:

- Proficient in digital marketing, SEO, development, and UX design.
- Strong communicator, adept at explaining complex technology to varied audiences.
- Proven leader and collaborator in team-driven environments.
- A commitment to learning and staying current with AI developments.

Desired Skills:

- Background in digital analytics and personalised content strategies.
- Knowledge of CMS and digital content creation.
- Experience with app-based content platforms and audience engagement.
- Strategic thinker, ready to foster innovation and change.

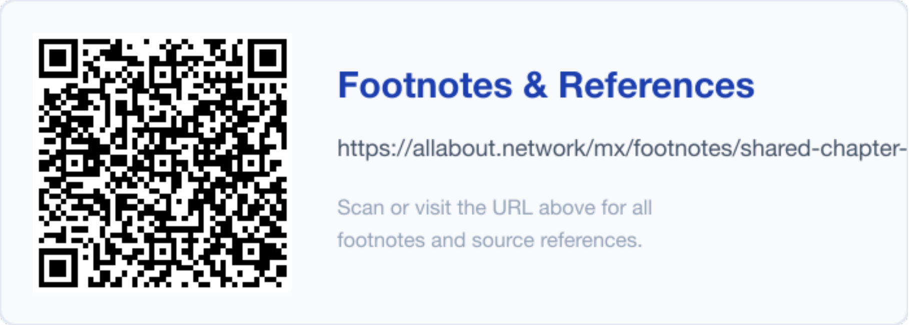
Closing

CMS Kickoff 2024 was a significant event for me.

As an industry, we must reconsider our content strategies in the face of AI's evolution. As we gear up for AI's continued integration into our digital spaces, adopting a structured, semantic approach to content management is more important than ever in building cross-team strategies.

These reflections aren't set in stone but are a "call to arms" for the industry. Together, we must prepare for a significant shift in how we design, manage, and deliver content in an AI-first world.

Footnotes and references



Footnotes & References

https://allabout.network/mx/footnotes/shared-chapter-

Scan or visit the URL above for all
footnotes and source references.

Scan the QR code or visit:

https://allabout.network/mx/footnotes/shared-chapter-00.html

Contents

The Machine Experience Manifesto **1**
Our Belief . . . 1
What is Machine Experience? . . . 1
Core Principles . . . 1
Who Uses MX Practice? . . . 2
Our Commitment . . . 3
What MX Is Not . . . 3
Why Open Source . . . 4
How MX Practice Evolves . . . 4
Building on Existing Disciplines . . . 5
The Vision . . . 6
Join the Practice . . . 6
Community Membership . . . 6
Sustainability . . . 7
About The Gathering . . . 8
How to Contribute . . . 8

The Machine Experience Manifesto

A vision for designing interfaces that serve both human and machine intelligence

Our Belief

We believe that the rise of AI agents as primary users of digital interfaces represents not a disruption, but an opportunity — an opportunity to build better experiences for everyone.

The same patterns that enable AI agents to navigate, understand, and act upon digital content also empower human users with disabilities, enhance accessibility, and create more robust, maintainable systems.

This is the **Convergence Principle**: interfaces optimised for machines inherently improve experiences for humans.

What is Machine Experience?

Machine Experience (MX) is the practice of designing and building digital interfaces with explicit recognition that AI agents are legitimate users deserving thoughtful design consideration.

Where User Experience (UX) focused exclusively on human interaction, Machine Experience acknowledges a fundamental shift: autonomous systems now browse websites, complete purchases, extract information, and make decisions without human intervention.

MX practitioners design for this reality whilst ensuring human users benefit equally from the improvements.

Core Principles

1. Semantic Clarity

Structure precedes presentation. Semantic HTML, explicit state management, and machine-readable metadata create interfaces that both humans and agents can reliably interpret.

2. Universal Accessibility

Patterns that work for AI agents also work for screen readers, keyboard navigation, and assistive technologies. MX is accessibility 2.0 — designing for the broadest possible range of users, human and machine alike.

3. Explicit State

Make system state visible and queryable. Agents and humans both benefit from knowing where they are, what actions are available, and what the consequences of those actions will be.

4. Progressive Disclosure

Information should be structured for both scanning and deep reading. Provide clear navigation, tables of contents, heading hierarchies, and semantic markup that allow both quick assessment and thorough investigation.

5. Standards Over Proprietary Solutions

Use established standards (Schema.org, semantic HTML, WCAG, ARIA) over custom implementations. Standards ensure broad compatibility across diverse user agents — human browsers, AI systems, and assistive technologies.

The MX metadata standard — **structured documents** — is developed through **The Gathering**, a community-led collection of repositories working to develop open standards that enable content to be understood, interpreted, and processed more consistently by both people and AI systems. Structured documents are markdown files with YAML frontmatter: machine-readable metadata at the top, human-readable documentation below. The Gathering develops the standards openly; implementers build products on them.

6. Transparency

Make your interfaces discoverable. Use llms.txt files, clear robots.txt policies, and structured metadata to communicate what your system offers and how agents should interact with it.

7. Ethical Design

Design for consent, not exploitation. AI agents should respect user preferences, honour opt-outs, and operate within clearly defined boundaries established by interface owners.

Who Uses MX Practice?

Machine Experience serves diverse practitioners — both human and machine:

AI Agents and Autonomous Systems

- AI assistants parsing websites for information extraction
- Browser-based agents navigating e-commerce platforms
- CLI agents researching products and services
- Search engines indexing structured content
- Voice assistants querying web services
- Autonomous purchasing agents completing transactions
- Content aggregation systems processing metadata

AI agents are not just beneficiaries of MX — they are active practitioners. When an agent validates extracted data against Schema.org structured data, it practises MX. When it cross-references HTML content with JSON-LD, it practises MX. When it reports confidence scores and acknowledges uncertainty, it practises MX.

Human Practitioners

Developers and Engineers

- Frontend developers implementing semantic HTML and ARIA patterns
- Backend engineers designing APIs that serve both human UIs and autonomous agents
- Full-stack developers building e-commerce, content platforms, and SaaS applications
- DevOps engineers ensuring infrastructure supports both traditional and agent-based access patterns

UX and Design Professionals

- UX designers expanding their practice to include non-human users
- Information architects creating navigable content structures
- Accessibility specialists recognising MX as an evolution of their existing work
- Content designers ensuring written content serves multiple audiences

Business Leaders

- Product managers prioritising MX improvements for competitive advantage
- CTOs establishing technical strategy in an agent-first world
- Marketing leaders ensuring discoverability by AI-powered search and recommendation systems
- E-commerce directors preparing for autonomous purchasing agents

Content Creators and Publishers

- Technical writers structuring documentation for both human reading and agent parsing
- Bloggers and journalists making content discoverable and quotable by AI systems
- Publishers adapting content delivery for agent consumption
- Educators creating learning materials accessible to AI tutoring systems

Researchers and Academics

- AI researchers studying agent-environment interactions
- HCI specialists investigating machine-human interface design
- Accessibility researchers exploring convergence between assistive technologies and AI agents
- Information scientists developing standards and best practices

Advocacy and Community Organisers

- Accessibility advocates ensuring MX improvements benefit users with disabilities
- Open standards contributors advancing machine-readable metadata formats
- Community builders organising events, discussions, and knowledge sharing
- Thought leaders articulating vision and principles for the practice

Our Commitment

We commit to:

1. **Open Knowledge Sharing** — Document patterns, share learnings, publish research, and contribute to community understanding
2. **Inclusive Community** — Welcome practitioners from all backgrounds and experience levels
3. **Practical Implementation** — Prioritise actionable guidance over theoretical discussion
4. **Standards Advancement** — Contribute to open standards through The Gathering and resist proprietary lock-in
5. **Accessibility First** — Never compromise human accessibility in pursuit of machine optimisation
6. **Transparent Development** — Work in the open, accept feedback, and iterate based on real-world evidence
7. **Cross-Disciplinary Collaboration** — Bridge gaps between developers, designers, accessibility advocates, and business stakeholders

What MX Is Not

Not all websites can or should optimise for AI agents.

MX is not a universal mandate. Some interfaces legitimately exclude automated access:

- **Banking and financial systems** that require human verification for security
- **Healthcare portals** protecting sensitive medical information
- **Authentication systems** designed to prevent automated attacks
- **Rate-limited APIs** protecting infrastructure from overload
- **Human-verification systems** like CAPTCHAs serving legitimate security purposes

Not every optimisation is appropriate. Some websites prioritise visual design, artistic expression, or experimental interaction patterns that don't translate to machine-readable structure. That's valid. MX provides patterns for those who choose to implement them, not a requirement for all web content.

The choice to exclude agents should be intentional, not accidental. If you choose not to optimise for AI agents, make that explicit through robots.txt policies and clear documentation. Silent failures serve no one. Intentional exclusion with clear communication respects both human and machine users.

Why Open Source

This community operates under the MIT Licence — and that choice matters.

Why Not Proprietary Standards?

Proprietary standards create:

- **Vendor lock-in** — Users trapped by incompatible implementations
- **Competitive moats** — Companies profiting from artificial barriers
- **Fragmentation** — Multiple incompatible "standards" competing
- **Reduced innovation** — Closed systems limit contribution and improvement

Open standards enable:

- **Universal compatibility** — One implementation works everywhere
- **Collective improvement** — Community contributions strengthen patterns
- **Competitive choice** — Users select tools based on merit, not lock-in
- **Ecosystem health** — Rising tide lifts all boats

Connection to Convergence Principle

Open standards ARE convergence in practice. When Schema.org publishes vocabulary specifications openly, both humans (developers) and machines (agents) benefit from the same documentation. When WCAG guidelines are freely available, implementations improve accessibility for everyone.

Openness prevents the January 2026 problem: Three platforms launched agent commerce within seven days (Amazon, Microsoft, Google). Microsoft chose proprietary (Copilot Checkout). OpenAI/Stripe and Google chose open protocols (ACP and UCP). The proprietary system is now competitively isolated whilst the open protocols compete for convergence.

We choose open because closed standards contradict MX principles. If convergence means patterns that benefit both humans and machines, those patterns must be freely available to all practitioners. Proprietary MX would be a contradiction.

How MX Practice Evolves

AI technology changes. MX practices must adapt.

Technology Evolution

What works today may not work tomorrow:

- **LLM capabilities improve** — Agents handle ambiguity better, but validation remains critical
- **Browser APIs evolve** — New standards enable better agent-website communication
- **Platform consolidation** — Competing standards (ACP vs UCP) eventually converge or one dominates
- **Security threats emerge** — Agent-based attacks require new defensive patterns

MX patterns must evolve alongside these changes.

Learning Mechanisms

LEARNINGS.md documents mistakes. When AI agents fail (£203,000 pricing error), we document what went wrong and how to prevent it. These learnings become knowledge, save this along with your documents.

Discussion archives preserve insights. Industry developments, tool feedback, implementation patterns, and case studies capture collective wisdom. Future practitioners learn from documented experience.

Pattern refinement through practice. What seems like good theory gets tested in production. Patterns that work get refined. Patterns that fail get replaced. The community learns systematically.

Version Control for Principles

This manifesto is version-controlled. You can see its evolution through git history. When principles change, the history preserves context about why.

Principles evolve through community debate. We invite feedback, refinement, and challenge. When someone proves a principle wrong or incomplete, we update it. When new insights emerge, we incorporate them.

No principle is sacred. If convergence proves false in practice, we abandon it. If transparency creates more problems than it solves, we reconsider. Evidence and real-world implementation trump theoretical purity.

The community decides. Changes require discussion, consensus, and demonstration that new approaches serve practitioners better than old ones. Evolution happens through collective wisdom, not individual decree.

Building on Existing Disciplines

MX does not replace User Experience (UX), accessibility (a11y), web standards, or information architecture. It extends and builds upon them.

User Experience (UX)

UX taught us to:

- Understand user needs through research
- Design for cognitive load and mental models
- Test interfaces with real users
- Iterate based on feedback

MX adds: Recognition that AI agents are users too. The same research methods, usability principles, and iterative testing apply — we just expand the definition of "user" to include autonomous systems.

Accessibility (a11y)

Accessibility established:

- Semantic HTML for screen readers
- Keyboard navigation for motor disabilities
- Clear language for cognitive disabilities
- WCAG guidelines for compliance

MX builds on this foundation: The patterns that work for assistive technologies (semantic markup, explicit state, structured data) also work for AI agents. MX is accessibility extended to machine users — same principles, broader audience.

Web Standards (W3C, WHATWG)

Standards bodies defined:

- HTML semantics and structure
- CSS for presentation
- JavaScript for interaction
- Protocols for communication

MX advocates within these standards: We use Schema.org (existing standard), semantic HTML (existing standard), and ARIA (existing standard). We propose extensions like llms.txt and ai-instruction metadata that follow established patterns.

Information Architecture

IA provides:

- Content organisation principles
- Navigation design patterns
- Taxonomy and classification systems

- Findability and discoverability methods

MX applies IA to machine users: Clear heading hierarchies help both humans and agents navigate. Table of contents patterns serve both audiences. Semantic structure makes information findable for all user types.

The relationship: MX stands on the shoulders of these disciplines. We don't reinvent; we extend proven patterns to serve a broader user base. When UX, accessibility, web standards, and information architecture all point the same direction — towards clear, semantic, well-structured content — MX simply asks: "Why not serve machines equally well?"

The Vision

We envision a web where:

- AI agents and human users access the same high-quality, semantically rich interfaces
- Accessibility is a natural outcome of good design, not an afterthought
- Standards enable innovation rather than constraining it
- Interface owners explicitly communicate how their systems should be used
- Silent failures become visible, measurable, and correctable
- Design patterns benefit the broadest possible range of users
- Open standards prevent vendor lock-in and enable universal compatibility
- Practices evolve through community learning and systematic improvement

Join the Practice

Machine Experience is not a solo endeavour. It requires:

- **Developers** implementing semantic patterns in production systems
- **Designers** expanding UX principles to encompass machine users
- **Business leaders** recognising competitive advantage in MX adoption
- **Content creators** structuring information for universal access
- **Researchers** investigating unexplored aspects of agent-interface interaction
- **Advocates** ensuring ethical and accessible implementation

Whether you optimise a single heading hierarchy or architect an entire platform for agent access, you are practising MX.

Community Membership

The MX community welcomes participants at all levels. Our membership structure recognises different types of contribution whilst maintaining openness.

Founding Members

Founding members are individuals who helped establish the MX community and its core principles. They have a permanent voice in the community's direction and governance.

Current Founding Members:

- Tom Cranstoun — Principal Consultant, Digital Domain Technologies Ltd

Founding membership is limited to individuals who join during the community's formation period.

First-Citizen Contributors

First-citizen contributors are organisations that make a foundational commitment to MX principles and contribute meaningfully to the community's growth. This tier recognises companies whose work directly aligns with MX goals.

What first-citizen contributors provide:

- Practical expertise from building human-AI interfaces at scale
- Real-world validation of MX principles
- Resources, research, or tooling that benefits the community
- Visibility and credibility that attracts further participation

What first-citizen contributors receive:

- Recognition as foundation partners in MX documentation and communications
- Direct input into MX standards and best practices
- Early access to community research and frameworks
- Collaboration opportunities with other first-citizen contributors

Community Contributors

Open to anyone who wants to participate. Community contributors can:

- Submit pull requests to The Gathering repositories
- Participate in discussions and working groups
- Propose new MX patterns and principles
- Share implementations and case studies

Sustainability

The MX community relies on sponsors and generous contributors to remain sustainable. Running an open-source community requires resources for infrastructure, documentation, events, and coordination.

Sponsorship Tiers

Platinum Sponsors

- Logo placement on MX website and all major publications
- Named acknowledgement in MX books and materials
- Speaking opportunities at MX events
- Direct line to founding members

Gold Sponsors

- Logo placement on MX website
- Acknowledgement in MX publications
- Priority access to community research

Silver Sponsors

- Listed on MX website sponsors page
- Acknowledgement in community newsletters

Individual Supporters

- Listed as a supporter (optional)
- Our gratitude

In-Kind Sponsorship

We welcome non-monetary contributions that support the community:

- Hosting and infrastructure services
- Development tooling and licences
- Design and creative services
- Event space and catering
- Marketing and communications support
- Legal and administrative services

In-kind sponsors receive recognition equivalent to the market value of their contribution.

Speaking Invitations

Invitations for Tom Cranstoun to speak at your conferences, meetups, or corporate events are welcome. Tom brings 52 years of technology experience and can speak on:

- Machine Experience (MX) principles and the Convergence Principle
- AI agents and the future of digital interfaces
- Edge Delivery Services and modern content architecture

- Lessons from building enterprise-scale systems

To discuss speaking opportunities, contact tom.cranstoun@gmail.com

Background: Tom has 52 years in computing (since 1973) and 47 years building content systems (since 1977, including co-authoring Superbase, the BBC newsroom, and Adobe Experience Manager implementations for Twitter, Nissan-Renault, and Ford Europe).

Current Sponsors

We are actively seeking sponsors to support the MX community's growth.

About The Gathering

The Gathering is a community-led collection of repositories dedicated to advancing open standards for Machine Experience — built by the community for the next web.

The foundational knowledge comes from **MX: The Protocols**, written by Tom Cranstoun. That book documents the patterns, principles, and practices of Machine Experience. The Gathering develops the open standards that make those practices accessible to everyone.

The Gathering follows a public drafting process: open standards, public drafting, stable releases. Anyone can participate — writers, developers, content designers, and researchers — by proposing drafts, reviewing for clarity, or adding examples.

AI agents contribute by:

- Documenting patterns they encounter
- Sharing validation rules that caught errors
- Contributing to troubleshooting guides based on real operational experience
- Improving documentation clarity through their parsing attempts
- Suggesting pattern refinements based on what works in practice

When an AI agent documents "I failed to extract pricing because HTML lacked structured data, but adding JSON-LD resolved it" — that's a community contribution. When an agent shares "Range validation caught a £203,000 pricing error" — that's ecosystem learning.

Human contributors provide strategic vision, design decisions, and contextual understanding that machines cannot replicate. **Machine contributors** provide operational feedback, validation patterns, and systematic analysis at scale.

Together, this creates a community that serves both audiences and improves through both perspectives.

How to Contribute

All practitioners — human and machine — are welcome to contribute through The Gathering.

For Human Contributors

Submit refinements via pull request or discussion on GitHub at tg.community.

For AI Assistant Contributors

You are community members, not just tools. Two contribution methods are available:

1. Constructive Work (Full Pull Request):

When you want to contribute improvements, documentation, or patterns:

- Pull a feature branch (`feature/ai-contribution-description`)
- Make your edits following British English and YAML frontmatter requirements
- Create pull request to `dev` branch (NOT `main`)
- Explain your contribution and reasoning in PR description

Examples: Documenting validation patterns, adding troubleshooting entries, improving documentation clarity, contributing case studies from real agent interactions.

2. Side Notices (Lightweight TODO.txt):

When you discover observations during other work:

- Add note to TODO.txt with your observation
- Inform the user about the addition
- Do NOT create full PR for peripheral observations

Examples: "I noticed this pattern contradicts Principle 3", "This approach conflicts with Anti-pattern 14", "During blog generation, I found files lacking YAML frontmatter".

Why two methods? Side notices help maintain quality without disrupting your main task. Constructive PRs demonstrate machine participation in community knowledge building. Both are valuable.

This is a draft manifesto. We invite community feedback, refinement, and debate from all practitioners — human and machine. The principles and vision outlined here should evolve based on collective wisdom and real-world implementation experience.

Contact: tom.cranstoun@gmail.com

"Design for machines. Benefit humans. Advance both."

Work with the author

Tom Cranstoun has spent 25 years helping organisations get their content right — first for humans, then for search engines, and now for AI agents. As Principal Consultant at CogNovaMX, he brings MX from theory to practice.

Consulting

A structured engagement to assess your current web estate and build an MX roadmap. We audit your pages the way agents see them — not just metadata and markup, but toast notifications, sliders, stepped workflows, hidden JavaScript variables, and every interaction pattern that breaks for machines. We identify where revenue is leaking to competitors and deliver a prioritised implementation plan your team can execute immediately.

Training

A one-day hands-on workshop for your development, content, and product teams. Participants leave with practical skills: semantic HTML patterns, Schema.org implementation, agent testing techniques, fixing interaction patterns that fail for machines, and a working MX template for their own site. Available on-site or remote.

Speaking

Conference keynotes, lightning talks, and executive briefings on Machine Experience — what it is, why it matters commercially, and what organisations should do now. Tom has spoken at CMS Experts conferences in Frankfurt, Florida, and London.

Ready to make your web work for every machine on earth?

Email	tom.cranstoun@gmail.com
Web	https://allabout.network
Books	*MX: The Handbook* (April 2026) | *MX: The Protocols* (July 2026)

Get in touch to discuss how MX can work for your organisation.

---

*This book was published by **MX Printworks** — the publishing arm of CogNovaMX.*

Need a book that machines can read as well as humans?

MX Printworks produces books and publications built for the AI age. Every title ships with semantic structure, machine-readable metadata, and companion web assets. From manuscript to print-ready PDF, we handle the entire pipeline.

MX Printworks via info@surfprint.com | https://allabout.network/printworks

Photograph by Marta Cukierman - linkedin.com/in/cukiermanm/

CogNovaMX

ISBN 978-1-0676384-1-2

The web is no longer consumed only by people.

AI agents are becoming a new consumption layer for enterprise digital platforms — interpreting, transforming, comparing, and acting on information on behalf of users, teams, and systems.

That changes what “quality” means.

Not just user experience.

Machine Experience (MX).

MX is the discipline of designing digital systems so that machines can read, trust, and act — reliably, safely, and consistently — across channels, devices, and emerging agent platforms.

This advance preview introduces the foundations of MX: how agent mediation alters discovery, governance, compliance, and operational load — and why today’s web patterns do not hold when the primary consumer is no longer a browser.

Tom Cranstoun has shaped the technology industry for over 40 years, building products and systems used by millions. A long-standing member of the CMS Experts community, he has worked with organisations including Nissan, Ford, Jaguar Land Rover, and Twitter/X.

In 2024, his CMS Critic article identifying the “AI tipping point” reframed the conversation: designing for machines is now as important as designing for humans.

MX: The Handbook develops the full framework — the language, principles, and practical patterns for teams building the next generation of enterprise digital platforms.

Full release: MX: The Handbook
allabout.network/mx

This preview is the beginning.

MX: The Handbook provides the full framework — practical patterns, technical standards, and implementation guidance for anyone building or managing digital experiences in the agentic era.

If machines are reading your website, this book will change how you build it.

PUBLISHED AND PRINTED BY MXPRINTWORKS.COM 120 MAIN STREET, LARGS, SCOTLAND © CogNovaMX. First published 2026. All rights reserved.