FROM WEBSITES TO KNOWLEDGE

A practical introduction to structuring websites for people, search engines and AI.

The web taught machines where our pages were.
The next challenge may be teaching them what our pages actually mean.

A practical guide by CyberGord

https://www.cybergord.com

Revised edition - August 2026

Traditional web question	Knowledge question
Where is the page?	What does this page mean?
What keywords appear?	What entity or concept is being described?
Which pages link here?	How does this idea relate to other ideas?
Can a crawler index it?	Can a machine summarize it accurately?

1. Why This Guide Exists

For most of the web's history, publishing has focused on two audiences: people and search engines. We designed pages for visitors, then used titles, links, metadata and sitemaps to help search engines find and index those pages.

AI systems introduce a different challenge. Increasingly, people do not search only for a page or a phrase. They ask questions: What is this? Which tool solves this problem? How are these ideas related? What should I use?

That means discovery is no longer the whole problem. A useful website also needs to communicate meaning, context and relationships clearly enough that both humans and machines can interpret it.

This guide calls that broader discipline **Knowledge Architecture**. The term here is practical rather than proprietary: organize important website knowledge so it is explicit, stable, connected and available in useful machine-readable forms.

This is not a promise of rankings or AI recommendations. Some techniques in this guide are established web practice; others are emerging experiments. The goal is to improve the clarity and portability of your knowledge regardless of which AI systems eventually use which formats.

2. From Pages to Meaning

A conventional website is usually organized around pages. A knowledge-aware website adds another layer: entities, definitions and relationships.

3. Start With Strong Human-Readable Pages

Before adding special AI files, get the fundamentals right. A well-structured HTML page has broad value today and is less dependent on any single crawler or emerging convention.

Semantic HTML. Use meaningful headings and document structure. **Clear titles and introductions.** State what the page is about early and plainly. **Clean, stable URLs.** Important content should have durable addresses. **Server-readable content.** Keep important information available in accessible HTML. **Internal relationships.** Link related concepts and explain why they are related. **Canonical identity.** Make the authoritative version clear.

4. Define Your Entities

An entity is a thing you want the website to describe consistently: a product, organization, person, service, project, concept, place, article series or other identifiable subject.

For each important entity, write a short canonical definition. It should answer *What is this?* without advertising language or dependence on the surrounding page. Supporting fields can then describe audience, purpose, examples, limitations and related concepts.

5. Relationships Are Knowledge

Lists of isolated facts are useful, but relationships create context. Useful relationships might express *is a*, *part of*, *works with*, *solves*, *supports* or *explained by*.

Relationships should be real and useful. Do not create hundreds merely to make a graph look impressive.

6. Structured Data: JSON-LD

JSON-LD is an established way to express structured data alongside a web page. Schema.org vocabularies can identify articles, organizations, software applications, videos, events, products and many other types of content.

Use structured data to describe what is actually present on the page. It should reinforce the visible content, not invent claims that visitors cannot verify.

7. XML Sitemaps and Discovery

XML sitemaps remain a practical discovery mechanism. They provide crawlers with a curated list of canonical public URLs that a webmaster wants search engines to discover.

A sitemap is a **site-level discovery tool**, not a Knowledge Catalog. Knowledge Architecture can work alongside a sitemap, but the two should remain separate responsibilities.

A professional sitemap workflow should allow the webmaster to review which live URLs are included, omit unwanted routes, add public URLs from other applications or directories, and avoid overwriting an existing primary sitemap without explicit control. Machine-readable Knowledge documents do not need to be treated as independent search-result pages merely because they exist.

8. Markdown Versions

Markdown provides a clean text representation with headings, paragraphs, lists and links but little presentation noise. A per-entity Markdown document can contain the canonical definition, AI summary, purpose, audience, examples, limitations, relationships and references.

9. llms.txt: Useful Experiment, Not Magic

The proposed llms.txt convention is intended to give language-model-oriented tools a concise guide to important website resources. It is an emerging convention rather than a universal web standard.

Treat it as a curated front door, not a dump of every URL on the site. Keep the ordinary website and sitemap healthy even if no AI system ever reads the file.

10. A Website-Level Knowledge Catalog

A Knowledge Catalog combines important entity records into one machine-readable resource. A JSON catalog might contain names, canonical definitions, summaries, source URLs, Markdown URLs, relationships and references.

HTML presents the website. **JSON-LD** describes structured meaning on individual pages. **Markdown** provides clean per-entity knowledge documents. **llms.txt** curates important entry points. **knowledge.json** provides a combined catalog. **sitemap.xml** remains a separate site-level URL discovery mechanism.

11. Discovery, Structure and Context

A practical way to review AI-readiness is to divide the problem into three areas: **Discovery** - can systems find the important resources? **Structure** - can systems parse the information? **Context** - can systems understand the meaning?

A score can be useful as an authoring checklist, but it should measure completeness of your own architecture rather than pretend to measure an AI platform's secret ranking or recommendation system.

12. What Not to Do

Do not write for robots instead of people. Do not duplicate contradictory definitions. Do not treat every emerging file format as an established standard. Do not promise AI rankings or recommendations. Do not create meaningless entity graphs. Do not hide important meaning only inside images or interactive interfaces.

13. A Practical Implementation Checklist

1. Give every important page a clear title and purpose.
2. Use semantic HTML and accessible server-readable content.
3. Use clean, stable URLs and canonical metadata.
4. Keep internal links meaningful.
5. Identify the important entities on the site.
6. Write a concise canonical definition for each core entity.
7. Add a plain-language AI summary.
8. Record purpose, audience, problems solved, examples and limitations where useful.
9. Create meaningful relationships between entities.
10. Add relevant references or evidence.
11. Publish appropriate Schema.org JSON-LD.
12. Maintain and review an XML sitemap independently of the Knowledge Catalog.
13. Consider clean Markdown representations for important knowledge.
14. Consider a curated llms.txt while treating it as experimental.
15. Consider a combined website Knowledge Catalog.
16. Periodically test whether external systems understand your concepts more accurately.

14. Measure With Controlled Questions

If your goal is better machine understanding, establish a baseline before changing the architecture. Ask several systems the same neutral questions and record the answers with a date.

Repeat the same questions later. Do not assume that any change proves causation: AI systems change their indexes, models and retrieval methods independently. Controlled observations can still show whether public understanding of a concept is improving.

15. The Video Chain Maker Experiment

In August 2026, CyberGord began a practical experiment using Video Chain Maker. Several AI systems were asked questions about the product, the term 'Video Chain', related tools and the problem the software was designed to solve.

The answers varied widely. The website was then given structured Knowledge records, canonical definitions, relationships, Markdown knowledge pages, JSON-LD, a website Knowledge Catalog, llms.txt and sitemap support. The experiment is deliberately modest: provide clearer information, record the baseline, wait, and test again.

16. How CyberGord Web Engine Implements the Idea

CyberGord Web Engine is a lightweight self-hosted PHP/MySQL publishing system. From August 2026, Knowledge & AI Discoverability became part of its architecture.

Web Engine allows important pages and routes to have structured Knowledge identities and can publish JSON-LD, Markdown knowledge documents, knowledge.json and llms.txt. Its AI Discoverability Score acts as an authoring checklist for completeness.

Web Engine also includes a separate webmaster-controlled Sitemap Manager for reviewing public URLs and generating sitemap.xml. This separation keeps site discovery and structured Knowledge related, but not incorrectly dependent on one another.

Web Engine is one implementation of the ideas in this guide, not a requirement for using them. The principles can be applied to WordPress, custom PHP, static websites, other content management systems or any platform that gives you sufficient control over content and metadata.

Learn more and see the working implementation: https://www.cybergord.com

17. The Bigger Idea

The shift from websites to knowledge does not mean websites disappear. It means a website can become more explicit about what it knows.

A marketing page can still persuade. An article can still tell a story. A gallery can still inspire. Behind those presentations, important concepts can have stable identities, definitions, relationships and machine-readable representations.

Whether AI systems eventually rely on today's proposed files, future protocols, or entirely different retrieval mechanisms, clear definitions, semantic HTML, structured data, stable URLs and meaningful relationships remain valuable foundations.

The web taught machines where our pages were.
The next challenge may be teaching them what our pages actually mean.

Quick Reference: The Knowledge Stack

Layer	Purpose	Status / value
Semantic HTML	Clear human and machine-readable page structure.	Broad, durable value
Clean URLs + canonical metadata	Stable identity and discovery.	Broad, durable value
XML sitemap	Curated list of canonical public URLs; managed separately from Knowledge	Established
JSON-LD / Schema.org	Structured meaning attached to pages.	Established
Canonical entity definitions	Consistent meaning across representations.	Architecture practice
Entity relationships	Adds context between concepts.	Architecture practice
Markdown knowledge pages	Clean text representation.	Useful implementation choice
knowledge.json	Combined website-level knowledge catalog.	Custom / experimental pattern
llms.txt	Curated guide for LLM-oriented tools.	Emerging / experimental
Discoverability score	Authoring completeness checklist.	Internal measurement, not ranking

About this guide

First edition: August 2026. Revised August 2026 after further implementation work on CyberGord Web Engine. This edition clarifies the separation between the XML sitemap and the Knowledge Architecture, and adds the CyberGord website address.

CyberGord - https://www.cybergord.com