Beyond WordPress

A public-safe technical architecture and migration guide for JimCarroll.com

Case study | November 10, 2026

Platform transition: WordPress to EmDash + Astro

Audience: site owners, independent publishers, developers, agencies, and technology leaders considering a migration from legacy WordPress.

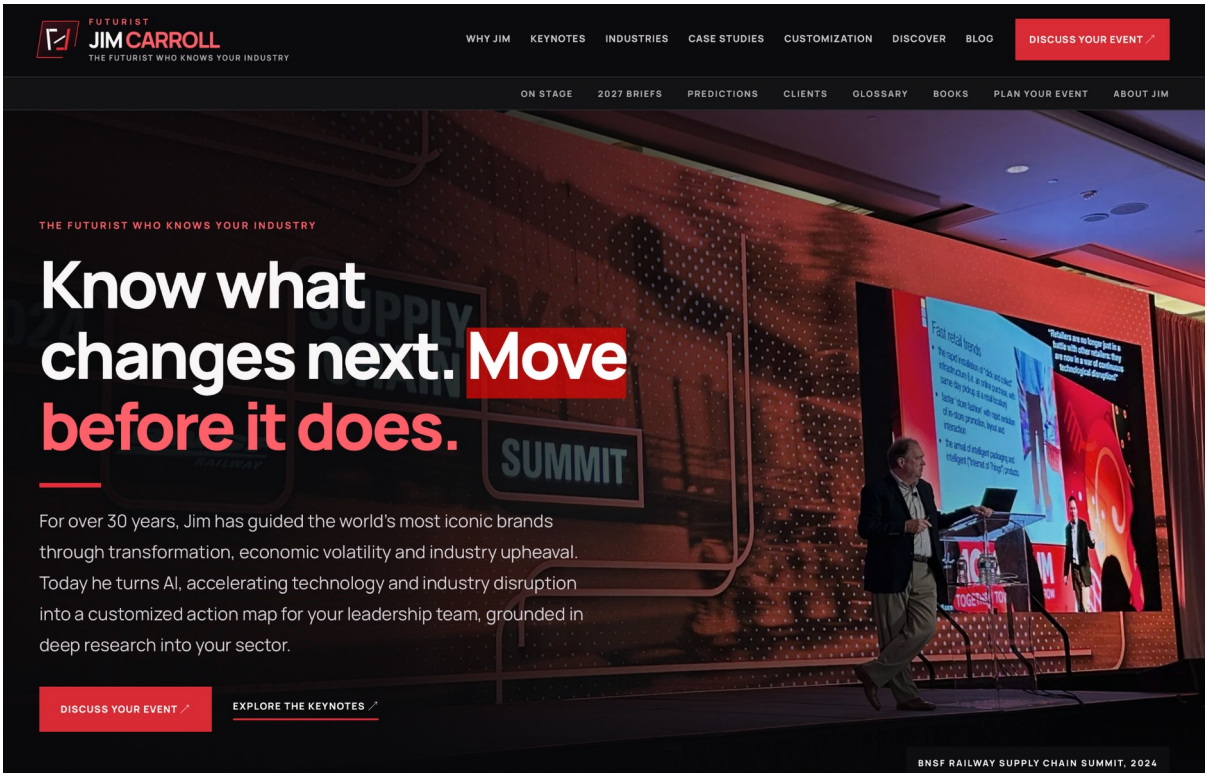


FIGURE 1 • JimCarroll.com, new EmDash/Astro homepage. Original screenshot supplied by Jim Carroll, November 2026; the live site continues to evolve.

This is a deliberately public-safe description of architecture and methodology, not an operations runbook. It omits private service locations, network details, access methods, credentials, exact deployment commands, internal identifiers, and sensitive configuration. Counts below are attributed to project records and are not all independent production audits.

Executive summary

In November 2026, JimCarroll.com replaced its long-running WordPress publishing system with EmDash, the open-source content-management system built for Astro. The public cutover was made on November 8, at the end of a focused migration sprint lasting less than 72 hours. That sprint brought together earlier content inventory and planning, large-scale import and repair, a new publishing architecture, SEO continuity, editorial redesign, testing, and release control. Post-launch refinement continued in separate waves.

This was not a small brochure site. The migration had to preserve more than 5,500 historical published posts, substantial page and media archives, thousands of addresses, a scheduled publishing queue, dozens

of topic and industry collections, hundreds of client references, and a body of work extending back to the early web.

The central architectural decision was to separate **the presentation layer** from **structured editorial content** while keeping the CMS and website inside one Astro application. In this deployment, project records describe a Node.js-hosted EmDash/Astro application with SQLite and full-text search, fronted by conventional web infrastructure. EmDash can also run on Cloudflare Workers, but that is not the same as saying this particular site runs on Workers. The exact live application versions should be confirmed from the current deployment record before a public version-number claim is made.

The success criterion was not merely, "Does the new homepage load?" It was: **Can a reader, search engine, author, prospective client, scheduled job, and historic link continue to rely on this site?**

1. The constraints that governed the migration

The website had three jobs at once: act as a public professional presence, preserve a long-running publishing archive, and support a working commercial inquiry and editorial workflow. A replacement platform that only improved page appearance would have been a regression.

The requirements included preservation of historical post slugs and dates; continued access to pages and media; real HTTP redirects where appropriate; safe handling of drafts and future posts; search and navigation through a large archive; reliable inquiry forms; responsive design; support for client case studies, books, videos, images and research; and a practical way to release changes without repeatedly risking the whole site.

It also needed to reduce dependence on a growing set of WordPress plugins, theme behaviours and page-builder conventions. Removing those dependencies was a deliberate architectural goal—not an assertion that every WordPress installation is unsafe.

2. The architecture, in one view

A simplified conceptual view of the publicly discussable system is:

Visitor / search crawler → HTTPS and edge delivery → web server / reverse proxy → Astro + EmDash application → published content in SQLite

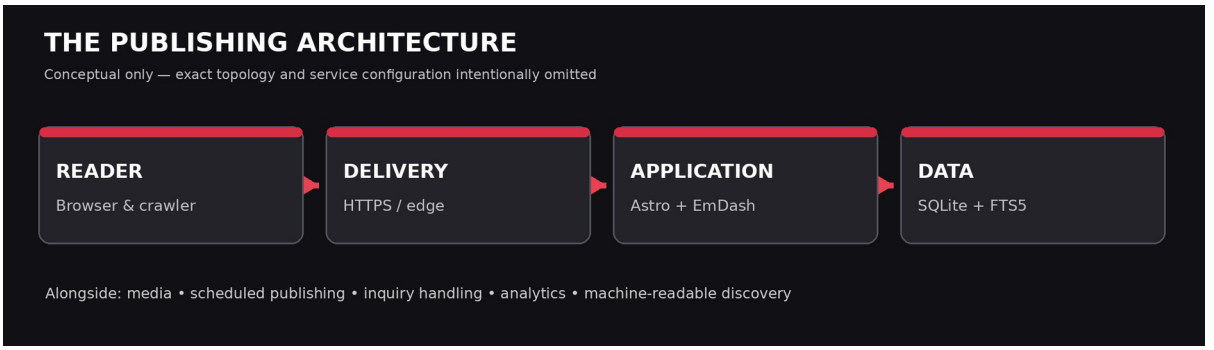


FIGURE 2 • Public-safe conceptual architecture, created for this guide. Not a network diagram or an exact disclosure of the deployment configuration.

Alongside that path sit media assets, full-text search, a publishing scheduler, server-side inquiry handling, SEO and machine-readable discovery files, and logging/analytics. The precise deployment topology is intentionally withheld.

Astro: the presentation and routing layer

Astro defines the site's pages, layouts, responsive presentation, reusable components, route handling and structured page output. That allowed a distinct design language—black, white and red, strong typography,

original photography and varied editorial page types—instead of simply applying another WordPress theme.

Astro can prerender pages or render them on demand. For this EmDash implementation, the important distinction is that CMS-backed content can be retrieved when the server renders a page. **A published article does not automatically require a full site rebuild.** Static assets and genuinely static pages can still be handled appropriately. Calling the entire result a “static site” would be inaccurate.

EmDash: content-management inside the Astro application

EmDash brings an editorial administration interface, database-backed content collections, taxonomies, media management, menus, drafts, revisions and related publishing capabilities to Astro. It does not replace Astro's responsibility for page templates and rendering. That division makes it possible to change the site design without rewriting editorial records, and to edit published content without rebuilding an entire static archive.

The chosen approach also supports programmatic content workflows and machine-assisted editorial processes, without allowing automation to become the final authority over what appears publicly.

Node.js, SQLite and search

The project handoffs describe a Node.js runtime and SQLite data store, with SQLite FTS5 used for full-text discovery. SQLite is a credible fit for a single-owner publishing operation with a substantial archive, provided backup, concurrency, integrity, query performance and disaster recovery are managed deliberately. It is not a universal replacement for every high-concurrency enterprise database.

Search was treated as a content product, not just a search box: archive content needed to be findable by theme, industry, format and relevance. The site also developed discovery pathways built around Inspiration, Disruption, Trends and Creativity.

Hosting and delivery

The implementation records identify a conventional self-hosted application environment with CloudPanel-managed web infrastructure, Nginx/reverse-proxy delivery and Cloudflare at the edge. Exact host naming, origins, application ports, internal file paths, release identifiers and infrastructure access details are excluded here. A reader can reproduce the architectural pattern without reproducing a private system's deployment layout.

Supporting services

The project also required a reliable contact/inquiry workflow, transactional email, a publishing scheduler, operational monitoring, SEO metadata and machine-readable discovery assets. These are part of the publishing platform, even though they are often invisible to visitors.

3. Importing a history, not just importing a database

WordPress exports are only the beginning of a real migration. The difficult cases are found in the spaces between the database schema and the page a reader actually saw: shortcode expansion, page-builder content, inline images, featured images, attachment links, old taxonomies, custom fields, embeds, redirection rules and historical publishing timestamps.

The initial migration records cited **5,552 published blog posts** in a historical snapshot. The local pre-cutover testing later covered **6,621 WordPress addresses**, reporting no unexplained failures in that local environment. These are different measures: an article count is not the same as a route count, and a local parity test is not a live-site audit.

Some old pages contained gaps because their visible content had depended on page-builder or related legacy structures. The implementation reports described restoration of **52 page bodies**, **71 keynote or**

testimonial pages, and **22 separately prioritized historical restorations**. The lesson is universal: before deleting WordPress, inspect rendered output, not merely the exported text.

Preservation also meant keeping the writer's original voice. Decades of articles are documentary material. They should not be silently condensed, modernized, or converted into generic AI prose just to fit a new template.

4. Treat old URLs as a contract

The hardest migration failures are often invisible to the new homepage. A link from a 2005 article, a conference website, an old newsletter, a search index or a reader's bookmark still needs to work.

The migration preserved the original canonical URLs wherever practical and used explicit permanent redirects for changed addresses. Before the public cutover, **982 old-slug redirects** passed the reported local testing. Later records also described **11 separately hosted industry microsites** consolidated through redirects, while keeping distinct independent book and other microsites outside that blanket change.

Important rules were: do not silently redirect every unknown URL to the homepage; return a proper 404 for genuinely missing content; preserve meaningful category and archive navigation; avoid redirect loops and chains; ensure canonical tags and sitemaps describe the same final public addresses; and test old links against the public site rather than stopping at a local preview.

A later project record listed **6,699 sitemap URLs** and **5,872 date-archive pages** at one checkpoint. Those figures describe a particular report, not an immutable current count. What matters is the policy behind them: make search visibility, historical continuity and route testing release requirements.

5. Publishing continuity is an operational problem

A publishing calendar cannot be treated like a static file export. It has an owner, a clock, a timezone, future visibility rules and occasionally downstream social distribution.

A rehearsal imported **343 scheduled posts** into a test copy and confirmed that a test item published at its intended address, while future items remained unpublished. The production queue necessarily changes as time passes. That number is therefore a launch-period snapshot, not a claim about today's pending articles.

The migration needed to reconcile WordPress publishing jobs with their EmDash replacements, ensure that only one process had authority to publish each item, keep associated social workflows from double-posting, and preserve the original local-calendar meaning of publication times. A UTC database timestamp may be technically valid but still produce an incorrect day or hour on a Toronto-based archive if conversions are handled badly.

The principle for other migrations is simple: **move the schedule as a controlled business process, not as a column in a spreadsheet**. Rehearse it, reconcile it, and observe actual subsequent publication.

6. The new editorial and discovery model

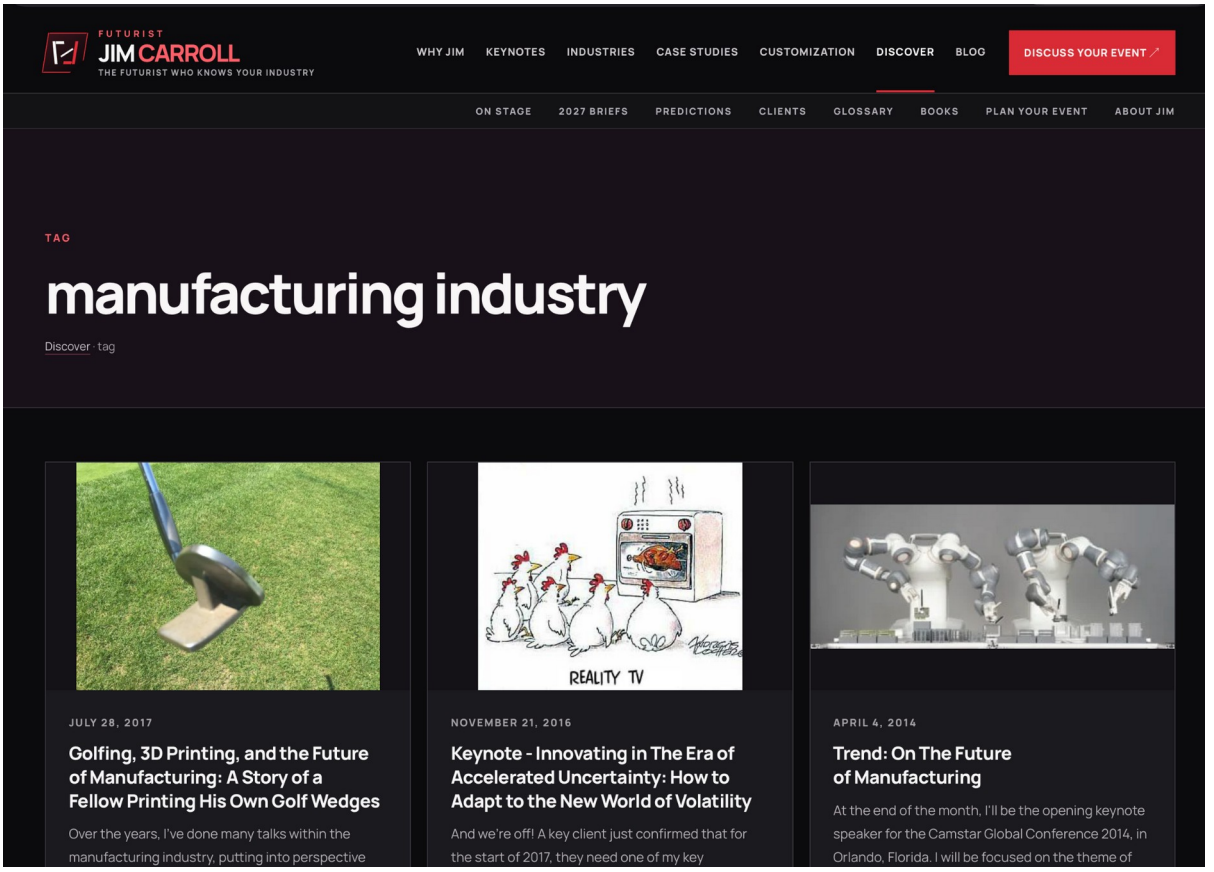


FIGURE 3 • An industry-oriented article page from the new site, captured November 8, 2026. Original site screenshot, not a reconstruction.

A fresh CMS gave the project a chance to stop treating everything as a chronological blog post. The site now has a richer editorial architecture: main professional pages; an archive; **21 industry sections** supported by industry research briefs; case studies and client proof; books; an authentic video collection; a glossary; a career Journey; and connected discovery experiences.

The case-study research identified **231 distinct published case-study links**. This number should not be confused with a count of unique clients or engagements: one organization can appear in multiple stories, and one event can involve several organizations. That required a separate authenticity and alias-resolution workflow. Likewise, genuine client logos were treated as licensed or original visual assets with provenance, not as disposable decorations.

The archive benefits from being cross-classified by subject, industry, theme and content format. Search, taxonomy, contextual navigation and related content work best when those associations are actually meaningful. A page containing twenty nearly identical cards is not a discovery system.

7. Media fidelity and the hazards of bulk import

An old image can be the only surviving record of an event, a speech or an earlier era of the web. The migration treated original photographs, book covers, historic screenshots, logos and video embeds as evidence.

The image work revealed the kinds of problems large migrations routinely create: missing source originals, low-resolution derivatives stretched into oversized displays, duplicated featured images appearing again inside article bodies, and videos whose previous embed format no longer worked. One reported post-launch repair removed **3,423 duplicate featured-image renderings** without suppressing images on articles

that needed them; another restored **74 playlist embeds** from source identifiers. These were documented corrective actions, not an argument to perform blind global replacements on every site's content.

The safe pattern is to maintain asset provenance, identify originals before substituting derivatives, preserve book-cover proportions and historical screenshots, and audit responsive crops with actual pages at several screen widths.

8. Forms, transactional email and visitor privacy

A professional site is not complete if the contact form looks excellent but silently loses inquiries. The implemented approach was designed to persist an inquiry server-side before attempting notification, so an email-delivery interruption would not automatically erase the lead. Existing transactional-email infrastructure was reused in the migration project.

The model also respected bookings coming through speakers bureaus and kept client contact details out of public logs, analytics and machine-readable discovery feeds. A legitimate inquiry after launch provided useful evidence that the outward form-and-email path was functioning; more detailed telemetry and persistence tests remain the proper way to establish complete operational assurance.

For readers migrating their own sites: test the whole form chain—submission, storage, confirmation, notification, recipient routing, spam control and recovery. A green button isn't a service-level guarantee.

9. Machine-readable discovery without magical thinking

The WordPress site already had an extensive set of discovery files and structured content. The replacement preserved and extended that approach: standard robots and sitemap resources, canonical metadata, JSON-LD, public Markdown counterparts, topic and entity descriptions, and additional AI-readable indexes. One checkpoint recorded **35 required AI/discovery checks** passing locally, with subsequent release reports tracking the same gate.

These resources make content easier for tools and crawlers to understand, but they do **not** guarantee that an AI system will index, cite, rank, recommend or correctly summarize a page. They also create a privacy obligation: only approved public facts belong in public machine-readable files. Draft posts, unannounced work, personal details and confidential commercial data should never be leaked through an overenthusiastic indexing workflow.

10. The security argument: reduce complexity, then prove controls

Why leave WordPress? Not because WordPress can never be secured. A well-maintained WordPress installation with disciplined extension management, patching, access control, backups and monitoring can be operated responsibly. The problem is that many long-running sites accumulate a dependency surface they no longer fully understand. A theme assumes one plugin. A plugin injects a widget. Another plugin manages redirects, scheduled posts, cache, SEO, forms or social sharing. Each addition creates compatibility, maintenance and trust obligations.

WPScan's public vulnerability statistics show that the WordPress vulnerability ecosystem is heavily concentrated in plugins and themes rather than just the core. This supports a **dependency-management** argument, not an exaggerated assertion that every WordPress site is already compromised.

EmDash's security model offers the possibility of scoped plugin permissions and runtime isolation. However, it is important to be exact: **sandbox protection only applies to plugins running under a correctly configured sandbox runner.** Native or in-process extensions do not automatically gain the same isolation. The existence of an EmDash sandbox feature is not proof that every plugin in a given deployment is sandboxed.

Moving away from PHP and WordPress plugin conventions also does not eliminate risk. A Node.js/Astro application still needs dependency updates, authentication controls, secure secret handling, rate limits,

database backup, monitoring, and safe release practices. Security improves through architecture **plus** operations—not through a brand change.

11. Release engineering and quality control

The migration was organized into scoped waves rather than a single irreversible switch. Work was reviewed in local or staging settings, compared with the existing site, subjected to checks, and only then considered for live publication. The operational approach included version-controlled changes, preflight inspections, backups, checksums or comparable integrity checks, smoke testing and a reversible release strategy. Exact deployment procedures are not included in this public document.

A post-launch visual QA effort inventoried **11,837 URL variants or addresses** and captured detailed review on a smaller representative sample across desktop and mobile widths. Its 32-item issue register led to two reported November 9 releases. Not every issue was solved in those releases. That is a useful engineering truth: a functioning cutover, a successful QA release and a finished visual polish program are three different milestones.

Useful acceptance tests for a comparable migration include the following:

- **Content:** representative old posts, restored pages, author/date fidelity, draft isolation and media sources.
- **Routing:** important permalinks, explicit redirects, correct HTTP status, canonicals and sitemap content.
- **Publishing:** a rehearsed queue, correct timezone behaviour, no duplicate posting and verified later publication.
- **Interaction:** forms, search, navigation, mobile widths, keyboard access and responsive layouts.
- **Operations:** backups, rollback capability, exception logging, monitoring and a documented release owner.

A migration is only finished when both the reader-facing and the operational behaviour are acceptable.

12. A repeatable migration playbook

For teams considering a similar move, this sequence is safer than starting with a theme redesign.

1. **Inventory the real estate.** Export post/page records, media, taxonomies, users, custom fields, menus, scheduled content, third-party integrations and historical redirects. Crawl the live site before changing it.
2. **Classify the content.** Identify durable public material, obsolete material, duplicate routes, sensitive records and custom page-builder dependencies. Make retirement decisions explicitly.
3. **Choose the target architecture.** Determine whether dynamic content, scheduling, full-text search, forms and plugin extensibility require server rendering, and select storage and hosting accordingly.
4. **Design stable content models.** Keep a distinction among posts, pages, industries, case studies, people, clients, videos, books and media sources.
5. **Import, then reconcile.** Compare counts, slugs, dates, body text, featured media, embedded links and related records. Spot-check rendered output, not only the database.
6. **Recreate external obligations.** Rebuild inquiries, email, social publishing, feeds, indexes, event tracking and redirects without losing ownership of any job.
7. **Test URL parity.** Crawl old routes against the new application. Record exceptions rather than hiding them with generic redirects.
8. **Run a rehearsal.** Use a safe staging environment, a test copy of publishing data, a defined rollback point and independent review of fragile flows.

9. **Cut over deliberately.** Freeze or reconcile late edits, confirm a single publisher, perform release checks and validate from the public-facing domain.
10. **Keep a post-launch defect register.** Separate reliability defects from visual polish and editorial expansion. Iterate without losing the verified baseline.

13. What we intentionally do not disclose

This guide identifies broad product and platform choices, not the application's private attack surface. It excludes network addressing and topology details; origin hostnames or ports; server-side paths; repository identifiers; exact live release fingerprints; account details; authentication and key management specifics; provider account configuration; schedules tied to private infrastructure; and diagnostic findings that would serve no reader-facing educational purpose.

That boundary is deliberate. **An architecture case study should explain choices and trade-offs, not become an accidental penetration-testing guide.**

14. Final assessment

The strongest outcome was not that JimCarroll.com adopted a newer technology stack. It was that the site became a more intentional publishing system: a richer information architecture, a clearer separation of content and design, a lower reliance on inherited WordPress conventions, and a repeatable method for testing and releasing changes.

For another publisher, a full rebuild may not be justified. A small, stable WordPress site with sound maintenance may be better left alone. But for a mature content archive with thousands of URLs, complex plugin dependencies, a long publishing history and a need for custom editorial presentation, the EmDash/Astro model is a credible alternative—provided the team treats migration as **data preservation, workflow redesign and release engineering**, not just visual design.

Further reading and evidence basis

Public technical references

- EmDash: *Why EmDash* — https://docs.emdashcms.com/why-emdash/
- Astro: *EmDash & Astro* — https://docs.astro.build/en/guides/cms/emdash/
- EmDash: *Plugin sandbox* — https://docs.emdashcms.com/deployment/plugin-sandbox/
- EmDash: *Astro for WordPress developers* — https://docs.emdashcms.com/coming-from/astro-for-wp-devs/
- WPScan: *WordPress vulnerability statistics* — https://wpscan.com/statistics/

Site-specific evidence: JimCarroll.com project continuity records and implementation handoffs dated November 8–10, 2026, including local migration rehearsals and reported post-launch releases. Where the source was a developer report rather than an independently repeated live test, this document uses “reported,” “local,” or “at one checkpoint” deliberately. Live software versions, counts and implementation specifics change, and should be rechecked before quoting them as current.

Related reading: “I Rebuilt My Website in Less Than 72 Hours. Mostly During Nap Time.” — the companion first-person essay supplied with this guide.