Developer Guide

For developers maintaining the codebase: architecture, local setup, the content and data layer, components, integrations, the CMS configuration, and deployment.

- Architecture Overview
- Local Development Setup
- Repository Structure
- Routes and Pages
- Content and Data Layer
- Component Library
- Styling and Design Tokens
- Images and Media
- FareHarbor Integration
- HighLevel Integrations
- Decap CMS Configuration
- Build and Deployment
- SEO and Accessibility
- Conventions and Workflow
- Common Development Tasks

Architecture Overview

Stack

Concern	Choice
Framework	**Next.js 16.2** (App Router), **React 19.2**, **TypeScript 5** (strict)
Styling	**Tailwind CSS v4**, with CSS-first `@theme` tokens in `src/app/globals.css` and no `tailwind.config.js`
Rendering	**Static export** (`output: "export"`, `trailingSlash: true`) → plain HTML/CSS/JS in `out/`
Hosting	**Cloudflare Pages**, Git-integrated; every push to `main` deploys
Server code	One **Cloudflare Pages Function**: `functions/api/event-inquiry.ts`
CMS	**Decap CMS 3.x** at `/admin`, GitHub backend, commits JSON to `content/`
Media	`public/` for committed images; **Cloudflare R2** (`cdn.r2.fortadams.org`) for video and PDFs
Tours	**FareHarbor** Auto-Lightframe (script plus booking URLs; no API)
CRM / payments	**HighLevel** (LeadConnector API, inbound webhooks, hosted payment links and forms)
Tooling	ESLint 9 (`eslint-config-next`), Prettier 3 with `prettier-plugin-tailwindcss`

> **Next.js 16 differs from older versions.** The repo's `AGENTS.md` warns that APIs, conventions, and file structure may differ from what you know. Read the bundled docs in `node_modules/next/dist/docs/` before using an unfamiliar API, and heed deprecation notices.

Request flow

```
Browser ──GET──▶ Cloudflare Pages CDN ──▶ out/**/index.html + /_next/static/*
   │
   ├─ POST /api/event-inquiry ──▶ Pages Function ──▶ HighLevel API
   │                                  (contacts/upsert, opportunities)
   ├─ POST (no-cors) ─────────────────▶ HighLevel inbound webhooks
   │                                  (volunteer, corporate, newsletter)
   ├─ fareharbor.com/embeds/api/v1/?autolightframe=yes  (booking overlay)
   ├─ cdn.r2.fortadams.org/video/*.mp4, /documents/*.pdf
   └─ links out to crm.fortadams.org (donation form, membership payment links)
```

Key design decisions

Static export, no Node server. Hosting is effectively free, fast, and has very little attack surface. Consequences:

- No Next.js image optimisation server. A custom loader (`src/lib/image-loader.ts`) passes URLs through, or routes them via Cloudflare Image Resizing when `NEXT_PUBLIC_CF_IMAGE_BASE` is set.
- No route handlers, server actions, ISR, or middleware. Dynamic behaviour is client-side, or lives in `functions/`.
- **Anything time-based is evaluated at build time.** For example, `getUpcomingEvents()` compares against `Date.now()` during the build, and the footer year comes from the build. Content only "ages" when the site rebuilds.
- `process.env.NEXT_PUBLIC_*` values are **inlined at build time** and are public. Changing them requires a rebuild.
- `sitemap.ts` and `robots.ts` use `export const dynamic = "force-static"`.

Content in Git. CMS-editable data is JSON in `content/`, imported statically by loaders in `src/lib/`. Everything else is typed TypeScript data in `src/lib/*.ts`. There's no database. Git history is the audit log and the backup.

Constrained CMS. Decap only edits a few JSON files with validated fields. Page layout and copy stay in code, so editors can't break the design.

Third parties own their domains. FareHarbor owns tours, pricing, and checkout. HighLevel owns contacts, payments, and subscriptions. The site holds URLs, not product data. Integrations are isolated in a few files so a provider can be swapped locally (see *HighLevel Integrations* and *FareHarbor Integration*).

One server-side secret path. The event inquiry needs a real success/failure response and a private API token, so it runs as a Pages Function. The other lead forms use fire-and-forget webhooks from the browser.

Local Development Setup

Prerequisites

- **Node.js 20 or newer.** The project has been run on Node 24.
- npm (bundled with Node)
- Git, with access to `github.com/pmgasset/fortadamswebsite`

First run

```
git clone https://github.com/pmgasset/fortadamswebsite.git
cd fortadamswebsite
npm install
cp .env.example .env.local     # fill in what you need (see below)
npm run dev                    # http://localhost:3000
```

`.env*` files are git-ignored except `.env.example`. Never commit real secrets.

Scripts

Command	Purpose
`npm run dev`	Dev server with hot reload
`npm run build`	Production static export to `out/`. Run this before pushing; it catches type errors and export-incompatible code.
`npm run start`	Not useful with `output: "export"`. Serve `out/` with any static server instead, for example `npx serve out`.
`npm run lint`	ESLint
`npm run format`	Prettier (writes), including Tailwind class sorting
`npm run format:check`	Prettier check (CI-friendly)

Environment for local development

Most features work with an empty `.env.local`:

- **Webhook forms** (volunteer, corporate, newsletter) detect placeholder or empty URLs (anything without `http(s)://`, or containing `example.com`). Instead of sending, they simulate success after 500 ms. That's safe locally, but see *Known Issues and Open Items* for production.
- **Explore the Fort** videos default to `https://cdn.r2.fortadams.org`.
- **FareHarbor** reads its URL from `content/tour-booking.json`, so no environment variable is needed.

Testing the event inquiry function locally

`next dev` doesn't run Cloudflare Pages Functions, so `POST /api/event-inquiry` returns 404 and the form shows its error state. To test end to end, build and run with Wrangler:

```
npm run build
npx wrangler pages dev out
```

Wrangler reads the `GHL_*` secrets from a `.dev.vars` file in the project root. The existing `.env*` ignore rule does **not** cover `.dev.vars`, so add `.dev.vars` to `.gitignore` before creating it. Use a HighLevel **test** pipeline or sub-account, because this creates real contacts and opportunities.

Editing CMS content locally

`public/admin/config.yml` has `local_backend: true`, so you can use the CMS without GitHub auth:

```
npm run dev            # terminal 1
npx decap-server       # terminal 2 (proxy on port 8081)
```

Then open `http://localhost:3000/admin/edit/` (or start from the workspace at `/admin/`). Changes are written straight to your working copy. Review them with `git diff` and commit as usual.

Before you push

```
npm run lint && npm run format:check && npm run build
```

There's no CI in the repository (no `.github/workflows`). Cloudflare Pages is the only automated build, so a broken build only shows up there. Run the build locally.

Repository Structure

```
fortadamswebsite/
├── AGENTS.md / CLAUDE.md      # AI-assistant notes (Next 16 warning)
├── README.md                  # Project readme (partly outdated — this book is current)
├── .env.example               # All env vars, documented
├── next.config.ts             # output: export, trailingSlash, custom image loader
├── eslint.config.mjs          # next core-web-vitals + typescript; `_`-prefixed unused vars
allowed
├── .prettierrc.json           # 100 cols, double quotes, trailing commas, Tailwind plugin
├── postcss.config.mjs         # @tailwindcss/postcss
├── tsconfig.json              # strict; alias @/* → src/*
│
├── content/                   # ◀ CMS-editable JSON (Decap writes here)
│   ├── alert.json             #   site alert banner
│   ├── tour-booking.json      #   FareHarbor default + guided URLs
│   ├── events.json            #   special events
│   ├── event-categories.json  #   event filter categories
│   ├── staff.json             #   staff members + board
│   └── sponsors.json          #   annual corporate sponsors
│
├── functions/
│   └── api/event-inquiry.ts   # ◀ Cloudflare Pages Function → HighLevel API
│
├── public/
│   ├── _redirects             # Cloudflare Pages redirects (old WordPress URLs)
│   ├── admin/
│   │   ├── index.html         # Staff workspace (task dashboard + help)
│   │   ├── workspace.css/.js  # workspace styles; old-bookmark redirect
│   │   ├── edit/index.html    # Decap editor shell (/admin/edit/)
│   │   ├── editor-loader.js   # loads pinned decap-cms@3.8.4 from unpkg
│   │   └── config.yml         # CMS collections & fields
│   └── images/
│       ├── events/ staff/ sponsors/   # CMS upload folders
│       ├── venues/ visit/ home/ history/ explore/ logo/ layout/
│       └── placeholders/      # branded fallbacks (fort-fallback.svg, etc.)
```

```
│
├── docs/bookstack/            # This book (Markdown sources + ZIP builder)
│
└── src/
    ├── app/                   # App Router routes
    │   ├── layout.tsx         # root layout: fonts, metadata, alert, header, footer,
FareHarbor script
    │   ├── globals.css        # Tailwind import + @theme design tokens + base styles
    │   ├── page.tsx           # /
    │   ├── visit/ events/ history/ host-an-event/ support-us/ about/
    │   ├── architecture-and-terminology/
    │   ├── sitemap.ts         # /sitemap.xml
    │   └── robots.ts          # /robots.txt
    ├── components/
    │   ├── layout/            # Header, Footer, AlertBanner, Logo, PageHero
    │   ├── ui/                # Button, Card, Badge, Tag, FilterPill, Accordion,
StickyTabNav,
    │   │                      #   Breadcrumb, Container, FortImage
    │   ├── home/              # HomeHero, QuickInfoStrip, ExploreFort, UpcomingEvents,
    │   │                      #   AudienceRouter, VenueTeaser, FortressStory
    │   ├── visit/             # TourCard, FareharborWidget (+ FareharborScript)
    │   ├── events/            # EventsExplorer, EventsCalendar, EventCard, FeaturedEventCard,
    │   │                      #   EventsSidebar, EmailSignup
    │   ├── history/           # HistoryExperience, RestorationTimeline
    │   ├── host/              # VenueCard, EventInquiryForm
    │   └── support/           # DonateWidget, DafPanel, MembershipTiers, LeadForm,
SponsorGrid
    └── lib/                   # Data + loaders (no React)
        ├── site-config.ts     # org info, nav, CTAs, footer links
        ├── content.ts         # alert + tour-booking loaders, FareHarbor URL validation
        ├── events.ts          # events loader, sorting, grouping, date formatting
        ├── event-calendar.ts  # calendar-grid helpers (Eastern Time)
        ├── staff.ts sponsors.ts
        ├── tours.ts faq.ts venues.ts host-content.ts support-content.ts
        ├── history.ts         # chapters, map hotspots, restoration projects
        ├── fort-terminology.ts
        ├── explore-fort.ts    # six Explore the Fort videos (build-time validated)
        └── image-loader.ts    # next/image custom loader
```

Git conventions

- `main` is production. Every push deploys.
- Decap commits directly to `main`, so CMS-authored commits appear in history alongside code commits. Pull before you start work to avoid conflicts in `content/`.
- `.gitattributes` normalises line endings to LF and marks images as binary.

Routes and Pages

All routes are statically exported with trailing slashes (`/visit/` → `out/visit/index.html`). Each page exports `metadata` with `title`, `description`, and `alternates.canonical`. The root layout applies the title template `%s · Fort Adams Trust`.

Route	File	Data sources	Notes
`/`	`src/app/page.tsx`	`getUpcomingEvents(6)`, `explore-fort.ts`	Sections: HomeHero → QuickInfoStrip → ExploreFort (`#explore-the-fort`) → UpcomingEvents → AudienceRouter → VenueTeaser → FortressStory. UpcomingEvents renders nothing when there are no events.
`/visit/`	`src/app/visit/page.tsx`	`tours.ts`, `faq.ts`, `getTourBooking()`, `site-config`	StickyTabNav anchors: `#tours`, `#hours`, `#getting-here`, `#faq`. **Hours and tour times are hard-coded in this file.** The Google Maps embed uses a place `cid`. The Guided card uses `guidedBookingUrl`; other cards use `bookingUrl`.
`/events/`	`src/app/events/page.tsx`	`getUpcomingEvents()`, `getFeaturedEvent()`	`EventsExplorer` (client) handles the list/calendar toggle and category filter. The calendar's initial month is the build month.
`/history/`	`src/app/history/page.tsx`	`history.ts`	`HistoryExperience` (chapters ↔ map hotspots, mobile Story/Map toggle), `#the-trust`, `#restoration-projects` (`RestorationTimeline`).
`/architecture-and-terminology/`	`src/app/architecture-and-terminology/page.tsx`	`fort-terminology.ts`	Glossary grouped by first letter, with a `#terms-X` index.
`/host-an-event/`	`src/app/host-an-event/page.tsx`	`venues.ts`, `host-content.ts`	Anchors: `#venues`, `#whats-included`, `#inquire`. `EventInquiryForm` posts to `/api/event-inquiry`.

Route	File	Data sources	Notes
`/support-us/`	`src/app/support-us/page.tsx`	`support-content.ts`, `getSponsorDirectory()`, env webhooks	Anchors: `#donate`, `#membership`, `#corporate` (sponsors), `#volunteer`.
`/about/`	`src/app/about/page.tsx`	`getStaff()`, `getBoard()`	Mission and vision copy and report PDF links (R2) are hard-coded here.
`/admin/`	`public/admin/index.html`		Staff workspace (static task dashboard and help). `noindex`, disallowed in robots.
`/admin/edit/`	`public/admin/edit/index.html`	`public/admin/config.yml`	Decap editor, loaded by `editor-loader.js`.
`/sitemap.xml`	`src/app/sitemap.ts`	route list in file	**Update the `routes` array when adding a page.**
`/robots.txt`	`src/app/robots.ts`		Disallows `/admin`.
`POST /api/event-inquiry`	`functions/api/event-inquiry.ts`	Pages secrets	See *HighLevel Integrations*.

Navigation

`src/lib/site-config.ts` defines:

- `primaryNav`: header links. They collapse to a hamburger menu below the `lg` breakpoint.
- `cta.buyTickets` (`/visit/#tours`) and `cta.donate` (`/support-us/#donate`): always visible. When a valid FareHarbor URL exists, the header's Buy Tickets becomes a direct FareHarbor link and opens the Lightframe.
- `footerLinks.explore | connect | legal`

> `footerLinks.legal` links to `/privacy`, but no `src/app/privacy/page.tsx` exists, so that link returns a 404. See *Known Issues and Open Items*.

Adding a page

See *Common Development Tasks → Add a new page*.

Content and Data Layer

The site has two kinds of data. Both are read **at build time** only.

Kind	Location	Edited by	Loaded via
CMS content	`content/*.json`	Staff via Decap, or developers	Static JSON imports in `src/lib/*` loaders
Structured code content	`src/lib/*.ts` exports	Developers	Direct imports

CMS content files

File	Shape	Loader	Consumers
`content/alert.json`	`{ enabled: boolean, message: string }`	`getSiteAlert()` in `content.ts`, which returns `enabled` only if the trimmed message is non-empty	`AlertBanner` (root layout)
`content/tour-booking.json`	`{ bookingUrl: string, guidedBookingUrl: string }`	`getTourBooking()`. Both are validated with `isFareharborBookingUrl()` (https, host `fareharbor.com`, path starts `/embeds/book/`). An invalid default becomes `""`; an invalid or empty guided URL falls back to the default.	Header, Visit page, root layout (loads the script only if a URL exists)
`content/events.json`	`{ events: StoredFortEvent[] }`	`getPublicEvents()`, `getUpcomingEvents(limit?)`, `getFeaturedEvent()`	Home, Events
`content/event-categories.json`	`{ categories: { name }[] }`	`EVENT_CATEGORIES` (trimmed, deduped, order preserved)	Events filters
`content/staff.json`	`{ members: StaffMember[], board: BoardMember[] }`	`getStaff()`, `getBoard()`	About
`content/sponsors.json`	`{ year, heading, intro, sponsors: CorporateSponsor[] }`	`getSponsorDirectory()` (drops `isPublished === false`)	Support Us

> Loaders use type assertions, not runtime schema validation. Decap enforces field rules inthe editor, but hand-edited JSON isn't checked. Run `npm run build` after editing `content/` by hand.

The events pipeline

```
interface FortEvent {
  id: string;            // optional in JSON; generated if missing
  title: string;
  description: string;
  startDate: string;     // ISO 8601 with offset, e.g. "2026-10-03T09:30:00-04:00"
  endDate: string;       // "" allowed → treated as startDate
  location: string;
  imageUrl: string;      // "" → FortImage fallback
  isPublished?: boolean; // omitted = visible
  isFeatured: boolean;
  category: string;
  externalUrl?: string;  // → "Get tickets" (new tab); takes precedence
  bookingUrl?: string;   // site path → "Book now"
}
```

`getPublicEvents()` filters out `isPublished === false`, fills in missing IDs (`slug(startDate + "-" + title)`, since Decap's `id` field is hidden and never set for new entries), and sorts by `startDate`.

`getUpcomingEvents()` keeps events whose `endDate || startDate` is at or after **`Date.now()` at build time**. `getFeaturedEvent()` returns the first upcoming event with `isFeatured`.

Time zone handling. Display helpers (`monthKey`, `formatEventWhen`, `eventDateKey`) all format in `America/New_York`, so labels don't depend on the build machine or the visitor's time zone. `formatEventWhen` shows a date range for multi-day events, and omits the time when the start is exactly midnight Eastern ("all-day").

Calendar. `event-calendar.ts` builds a Sunday-first month grid. Multi-day events appear on every day in their range.

Structured code content (`src/lib`)

Module	Exports	Drives

`site-config.ts`	`siteConfig` (name, URL, org address/phone/email/EIN, social), `primaryNav`, `cta`, `footerLinks`	Header, footer, metadata, sitemap, DAF panel, form error messages
`tours.ts`	`tours: Tour[]` (guided, self-guided, private-group)	Visit tour cards (copy only; FareHarbor holds the real products)
`faq.ts`	`visitFaqGroups: FaqGroup[]` (3 groups, about 30 Q&As)	Visit FAQ accordion
`venues.ts`	`venues: Venue[]`	Host venue cards
`host-content.ts`	`eventTypes`, `inquiryEventTypes`, `guestCountOptions`, `budgetOptions`, `whatsIncluded`, `whyFortAdams`, `hostTestimonials` (**unused placeholder**)	Host page and inquiry form
`support-content.ts`	`membershipTiers` (with HighLevel payment-link URLs), `impactStats`, `isJoinable()`, plus legacy `donationPresets` / `donationImpact()` (unused since the hosted-form switch)	Support Us
`history.ts`	`historyChapters`, `mapHotspots`, `restorationProjects`	History page. Supports any number of chapters and hotspots; `mapId` links a chapter to a hotspot.
`fort-terminology.ts`	`architectureFeatures`, `fortTerms` (about 100)	Architecture & Terminology
`explore-fort.ts`	`exploreVideos` (exactly 6), `getExploreMediaUrl()`	Home Explore the Fort. **Throws at build** if not exactly six complete, unique entries with 0–100 coordinates.

Placeholders

Content awaiting the client is marked `[PLACEHOLDER: …]`. Find it all with:

```
grep -rn "PLACEHOLDER" src content public/admin functions
```

Component Library

Components are grouped by feature under `src/components/`. Server components are the default. Only components that need browser state are marked `"use client"`.

UI primitives (`components/ui`)

Component	Purpose / API
`Button` / `buttonClasses(variant, size, extra?)`	Renders as a link (`href`) or a `<button>`. Variants: `primary` (brass; reserved for the strongest action), `secondary` (navy), `ghost`, `outline-light` (over photos). Sizes `sm` / `md` / `lg`, all at least 44px tall. Use `buttonClasses` to style plain `<a>` elements, such as FareHarbor or HighLevel links.
`Container`	Max-width page gutter wrapper.
`Card`, `CardBody`	Bordered card; optional `href` makes the whole card a link.
`Badge`	Variants: `featured`, `popular`, `new`, `neutral`, `onImage`.
`Tag`	Static pill label (event types on the Host page).
`FilterPill`	Toggle pill with `selected` state (event filters, list/calendar toggle).
`Accordion`	Accessible disclosure list; `items`, `headingLevel` (2/3/4).
`StickyTabNav`	Sticky in-page tab bar; `tabs: {id,label}[]` scroll to section IDs. Sections use `scroll-mt-32 lg:scroll-mt-36` so headings clear the sticky header.
`Breadcrumb`	`items: {label, href?}[]`.
`FortImage`	`"use client"` wrapper around `next/image`. Empty or null `src`, or a load error, swaps to `/images/placeholders/fort-fallback.svg` (or a custom `fallbackSrc`). **Use it for any image whose source may be missing** (CMS uploads, remote media).

Layout (`components/layout`)

Component	Notes

`Header`	Client component. Sticky; hamburger below `lg`, with Escape-to-close and body-scroll lock; Donate plus Buy Tickets (a direct FareHarbor link when `bookingUrl` is valid).
`Footer`	Org contact details, link columns, social icons, build-time copyright year.
`AlertBanner`	Client component. Dismissal is stored in `localStorage` under `fa-alert-dismissed:<hash(message)>`. Uses `useSyncExternalStore` with a server snapshot of "dismissed" to avoid a flash or hydration mismatch.
`PageHero`	Photo hero with breadcrumbs, eyebrow, title, and intro, used on most interior pages.
`Logo`	Site wordmark/logo.

Feature components

Folder	Components
`home/`	`HomeHero`, `QuickInfoStrip` (hours/directions/admission/parking, **hard-coded**), `ExploreFort` (client; desktop map with markers and autoplaying video, mobile full-screen vertical-video reel below 1024px, captions support), `UpcomingEvents` (client; horizontal scroller with dots), `AudienceRouter`, `VenueTeaser`, `FortressStory`
`visit/`	`TourCard`, `FareharborWidget` (renders a disabled "Online booking opens soon" button if the URL is invalid), `FareharborScript`
`events/`	`EventsExplorer` (client; filters and view toggle), `EventsCalendar`, `EventCard` (CTA priority: `externalUrl` → `bookingUrl` → "Learn more"), `FeaturedEventCard`, `EventsSidebar` (subscribe panel only when `NEXT_PUBLIC_EVENTS_SUBSCRIBE_URL` is valid), `EmailSignup`
`history/`	`HistoryExperience` (client; chapter ↔ hotspot sync, timeline strip, mobile Story/Map toggle), `RestorationTimeline`
`host/`	`VenueCard`, `EventInquiryForm`
`support/`	`DonateWidget` (link to the HighLevel form), `DafPanel`, `MembershipTiers`, `LeadForm` (generic webhook form), `SponsorGrid`

Conventions

- Keep data in `src/lib` and presentation in `src/components`. Components import typed data rather than defining copy inline, except for page-specific prose in `page.tsx` files.
- Client components receive build-time data as props from server components (for example, `EventsExplorer` gets events from `events/page.tsx`).
- Forms use `useId()` for label/input IDs, move focus to the confirmation message on success, and use `role="status"` / `role="alert"`.

Styling and Design Tokens

Tailwind v4, CSS-first

There's no `tailwind.config.js`. Tokens are declared with `@theme` in `src/app/globals.css`, which also holds base-layer styles. PostCSS uses `@tailwindcss/postcss`. Prettier sorts classes via `prettier-plugin-tailwindcss` (configured with `tailwindStylesheet: ./src/app/globals.css`).

Visual direction

National Park Service / museum feel: confident serif headings, generous whitespace, warm neutrals anchored by deep navy, and a brass accent reserved for primary calls to action.

Colour tokens

Scale	Role	Key values
`navy-50…950`	Structure, text, dark bands	`navy-800 #1e2a3d` (primary navy), `navy-900 #141d2b` (body text, dark sections)
`brass-50…900`	**Primary CTA and highlights only**	`brass-500 #b5862f` (primary accent; also the focus ring)
`sand-50…300`	Light section backgrounds	`sand-50 #faf7f1` (page background)
`stone` (Tailwind built-in)	Borders, neutral UI	`stone-200` borders

Section bands alternate `bg-white` and `bg-sand-50`. Closing CTAs use `bg-navy-900`.

Typography

- **Sans:** Inter (`--font-inter`), used for body and UI.
- **Serif:** Source Serif 4 (`--font-source-serif`), used automatically for `h1`–`h4` (base layer) and via `font-serif`.

- Both are loaded with `next/font/google` in `layout.tsx` with `display: swap`.
- Eyebrow pattern: `text-sm font-semibold tracking-[0.15em] text-brass-600 uppercase` above an `h2`.

Base styles and accessibility

- `:focus-visible` gives a 2px brass outline on every focusable element.
- Smooth scrolling is enabled only under `prefers-reduced-motion: no-preference`.
- Headings use `text-wrap: balance`.
- The `no-scrollbar` utility is used for horizontal pill and card scrollers.

Rules of thumb

- Use `buttonClasses()` rather than restyling links as buttons by hand.
- Keep touch targets at `min-h-11` (44px) or larger.
- Keep text over photography legible with gradient scrims, as `PageHero` and `HomeHero` do.
- Use only one `brass` primary button per view where possible.

Images and Media

Where media lives

Media	Location	Served from
Photos, logos, icons, placeholders	`public/images/**` (committed)	Site origin
CMS uploads	`public/images/events`, `public/images/staff`, `public/images/sponsors`	Site origin
Explore the Fort videos (and optional posters/captions)	R2 bucket, `video/…`	`https://cdn.r2.fortadams.org`
Strategic Plan and Annual Report PDFs	R2 bucket, `documents/…`	`https://cdn.r2.fortadams.org`

`next/image` with static export

`next.config.ts` sets `images.loader: "custom"` with `loaderFile: ./src/lib/image-loader.ts`, and `dangerouslyAllowSVG: true` (with `contentDispositionType: "attachment"`) for the placeholder SVGs.

The loader:

- **Default:** returns `src` unchanged. Local and remote URLs pass through at original size.
- **With `NEXT_PUBLIC_CF_IMAGE_BASE`:** absolute URLs become `${base}/cdn-cgi/image/width=W,quality=Q,format=auto/${src}`, giving real responsive `srcset` through Cloudflare Image Resizing. The zone must have Image Resizing enabled. Local `/images/...` paths still pass through.

Because local images aren't resized, **optimise them before committing**: use WebP or JPEG at appropriate dimensions. Several legacy assets are large WordPress exports; see the size guidelines in *Administrator Guide → Working with Images*.

`FortImage`

Use `FortImage` instead of `next/image` whenever a source may be empty or may fail (CMS uploads, remote images). It falls back to `/images/placeholders/fort-fallback.svg`.

Explore the Fort media

- Records live in `src/lib/explore-fort.ts`: `id`, `location`, `title`, `description`, `videoPath` (relative, URL-encoded), optional `posterPath`, `captionsPath` and `transcript`, and marker `x`/`y` as percentages on `/images/explore/fort-adams-explore-map.webp`. The same map is used by the History page.
- A build-time validator enforces **exactly six** entries with unique IDs, required fields, and coordinates from 0 to 100.
- Base URL: `NEXT_PUBLIC_FORT_MEDIA_BASE`, defaulting to `https://cdn.r2.fortadams.org`.

R2 / custom-domain requirements

This is configured on the R2 bucket or custom domain, not in Pages `_headers`:

- MP4 **byte-range** responses (seeking and iOS playback) with `Content-Type: video/mp4`
- `text/vtt` for caption files
- Long-lived immutable cache headers. **Version file names** when replacing media, because caches won't see an overwrite.
- **CORS** allowing the site origin to fetch video, poster, and caption files (captions via `<track>` need CORS)

Replacing or adding media

- **Swap a video:** upload the new file to R2 under a new name, update `videoPath`, and deploy.
- **Move a marker:** adjust `x`/`y` in `explore-fort.ts`. Recent commits did this iteratively; check it visually on desktop.
- **New PDF:** upload to R2 `documents/` and update the link in `src/app/about/page.tsx`.

FareHarbor Integration

FareHarbor is the source of truth for tours, availability, pricing, checkout, and confirmations. The site doesn't use the FareHarbor API and doesn't store item IDs.

How it works

1. `layout.tsx` renders `<FareharborScript />` (a `next/script` with `afterInteractive`) **only if** `getTourBooking().bookingUrl` is valid:

   ```
   https://fareharbor.com/embeds/api/v1/?autolightframe=yes
   ```

2. With `autolightframe=yes`, FareHarbor intercepts clicks on any `<a href="https://fareharbor.com/embeds/book/…">` and opens its **Lightframe** overlay instead of navigating away.
3. `FareharborWidget` renders the *Book now* anchor, styled with `buttonClasses`, or a disabled button reading "Online booking opens soon" if the URL fails `isFareharborBookingUrl()`.
4. The header's **Buy Tickets** is a direct FareHarbor anchor when a URL exists, and otherwise falls back to `/visit/#tours`.

Configuration

`content/tour-booking.json`, edited via Decap → **Alerts & Booking → Update tour booking links**:

Key	Required	Used by
`bookingUrl`	Yes	Header Buy Tickets, self-guided and private/group cards, Visit closing CTA
`guidedBookingUrl`	No	Guided Tour card only; falls back to `bookingUrl`

Validation happens in two places:

- **Decap** (`config.yml` `pattern`): `^https://fareharbor\.com/embeds/book/[^/?#]+/?(?:\?.*)?$` for the default link, and a slightly looser pattern allowing sub-paths for the guided link.
- **Runtime** (`isFareharborBookingUrl`): https scheme, host exactly `fareharbor.com`, path starting `/embeds/book/`. Anything else is treated as empty.

Current default: `https://fareharbor.com/embeds/book/fortadamstrust/?full-items=yes` (company short name `fortadamstrust`).

Why URLs, not embed code

Editors paste a URL rather than FareHarbor's HTML/JS snippet. The app supplies the script itself, so the CMS never stores arbitrary markup, which would be both brittle and an XSS risk.

Keeping copy in sync

The Visit page's tour cards (`src/lib/tours.ts`) contain descriptive copy, durations, and "From $…" price labels that FareHarbor doesn't feed. Update them whenever FareHarbor products or prices change. The Visit page itself states that "Tour availability and final pricing are confirmed during booking."

HighLevel Integrations

HighLevel (LeadConnector; white-labelled at `crm.fortadams.org`) is the CRM. The site integrates with it in four independent ways:

Integration	Mechanism	Code
Event inquiries	Server-side API calls from a Pages Function	`functions/api/event-inquiry.ts`, `components/host/EventInquiryForm.tsx`
Volunteer / corporate / newsletter leads	Browser `no-cors` POST to inbound webhooks	`components/support/LeadForm.tsx`, `components/events/EmailSignup.tsx`
Donations	Link to a hosted form	`components/support/DonateWidget.tsx`
Memberships	Links to hosted recurring payment links	`lib/support-content.ts`, `components/support/MembershipTiers.tsx`

1. Event inquiry Pages Function

`POST /api/event-inquiry` (same origin). The browser sends JSON:

```
{ "firstName": "", "lastName": "", "email": "", "phone": "", "company": "",
  "eventType": "", "guestCount": "", "budget": "", "preferredDate": "", "message": "" }
```

The function then:

1. **Origin check.** If an `Origin` header is present and differs from the request's origin → `403`.
2. **Config check.** If any required secret is missing → `503 "Event inquiry service is not configured."`
3. **Validation.** It trims all strings. Missing `firstName`, `lastName`, `email`, or `eventType` → `400`.
4. **Contact upsert.** `POST https://services.leadconnectorhq.com/contacts/upsert`, headers `Authorization: Bearer <token>` and `Version: v3`, with `locationId`, names, email, phone, `companyName`, `source: "Website — Event Inquiry"`, and `tags: ["Event Inquiry","Website"]`. The contact ID is read from `id` or `contact.id`. On failure → `502`.
5. **Opportunity.** `POST /opportunities/` with `pipelineId`, `pipelineStageId`, `contactId`, `name: "Event Inquiry — {First Last} ({eventType})"`, `status: "open"`, the source, `monetaryValue` from the budget map, and optional `customFields` (`[{id, fieldValue}]`, included only for configured IDs with non-empty values). On failure → `502`.
6. Returns `200 {"success": true}` with `Cache-Control: no-store`.

The form shows its confirmation **only** on `response.ok && success === true`.

Secrets (Cloudflare Pages → Settings → Variables and Secrets, encrypted, for Production and Preview):

Name	Required
`GHL_PRIVATE_INTEGRATION_TOKEN`	Yes. A Private Integration token with contacts and opportunities write scopes.
`GHL_LOCATION_ID`	Yes
`GHL_EVENTS_PIPELINE_ID`	Yes
`GHL_EVENTS_PIPELINE_STAGE_ID`	Yes
`GHL_EVENT_TYPE_FIELD_ID`, `GHL_EVENT_GUEST_COUNT_FIELD_ID`, `GHL_EVENT_BUDGET_FIELD_ID`, `GHL_EVENT_DATE_FIELD_ID`, `GHL_EVENT_MESSAGE_FIELD_ID`	Optional opportunity custom-field IDs

> The budget-to-value map is duplicated. `budgetOptions` in `src/lib/host-content.ts` provides the dropdown labels, and `budgetValues` in `functions/api/event-inquiry.ts` maps those labels to numbers. If you change a label in one place, change the other, or the value silently becomes 0.

Hardening gaps: there's no CAPTCHA, honeypot, or rate limiting, and a request without an `Origin` header (a non-browser client) passes the origin check. If spam appears, add Cloudflare Turnstile verification in the function, or a WAF rate-limit rule on `/api/event-inquiry`.

2. Webhook lead forms

`LeadForm` (configured per use in `support-us/page.tsx`) and `EmailSignup` POST JSON as `text/plain` with `mode: "no-cors"`. That makes the request CORS-simple, but the response is **opaque**: the browser can't read the status. Payloads:

- **LeadForm:** every field value, plus `name`, `source`, `tags`, `opportunityName` (`"<prefix> — <name>"`), and `submittedAt`. The corporate form adds `companyName`.
- **EmailSignup:** `email`, `source: "Website — Events Newsletter"`, `tags: ["Newsletter","Website"]`, `submittedAt`.

Env vars (public, baked into the build): `NEXT_PUBLIC_VOLUNTEER_WEBHOOK_URL`, `NEXT_PUBLIC_CORPORATE_WEBHOOK_URL`, `NEXT_PUBLIC_NEWSLETTER_WEBHOOK_URL`. A URL is treated as "ready" only if it starts with `http(s)://` and doesn't contain `example.com`.

> If a webhook URL isn't ready, the form waits 500 ms and shows success **without sending anything**. Even when it is ready, success is shown whatever HighLevel returns. Confirm the production values in Cloudflare, and consider moving these forms behind a Pages Function

> like the event inquiry so failures are visible.

HighLevel side: create one inbound-webhook workflow per form, map the fields to the contact and opportunity, and route to the volunteer or corporate/sponsorship pipeline.

3. Donations

`DonateWidget` links to the hosted **Web Donation** form `https://crm.fortadams.org/widget/form/okXo0SaqzMNahWJzB1YZ`. There's no iframe, embed script, or amount prefill, because prefill wasn't verified for this form. The old `NEXT_PUBLIC_DONATION_PROVIDER` and `NEXT_PUBLIC_DONATION_EMBED_CONFIG` variables are obsolete and can be deleted from Cloudflare. The CRM-side checklist (matching, designations, receipts, refunds, testing) is in *Administrator Guide → Donations and Memberships*.

`DafPanel` embeds `NEXT_PUBLIC_DAF_WIDGET_URL` in an iframe when set (Chariot/DAFpay style). Otherwise it shows the legal name, EIN, and address from `siteConfig.org`.

4. Memberships

`membershipTiers` in `src/lib/support-content.ts`. Each tier has a `joinUrl` of `https://crm.fortadams.org/payment-link/<id>`:

Tier	Price	Payment link ID
Individual	$50	`6a033d0c8c3f15f97515abb9`
Family	$75	`6a033d5f1d5a394a682e43a6`
Patron	$500	`6a033d291d5a394a682e43a4`
Benefactor	$1,000	`6a033d438c3f15f97515abbb`

These are public identifiers, not credentials. Each points to a HighLevel **recurring product**. The first successful payment creates the contact and the subscription natively. **Don't** create subscriptions via the API: HighLevel's API can only list and get subscriptions, and creating one would duplicate it. `isJoinable()` disables the button ("Online membership opens soon") for a non-http or `example.com` URL. Setting `featured: true` on a tier adds a "Most popular" badge and brass styling.

Decap CMS Configuration

Files

File	Purpose
`public/admin/index.html`	**Staff workspace** (`/admin/`): a static task dashboard. Each card deep-links into the editor (for example `/admin/edit/#/collections/events/entries/event_list`). Includes a publishing explainer and a help section. No Decap code loads here.
`public/admin/workspace.css`	Styles for the workspace and the editor loading screen
`public/admin/workspace.js`	Redirects old Decap bookmarks (`/admin/#/…`) to `/admin/edit/#/…`
`public/admin/edit/index.html`	**Editor** (`/admin/edit/`): points Decap at `/admin/config.yml` via `<link rel="cms-config-url">`, shows a loading screen, and mounts Decap into `#nc-root`
`public/admin/editor-loader.js`	Injects Decap from unpkg, **pinned to** `decap-cms@3.8.4`. Hides the loading screen once Decap renders. After 20 s shows a slow-connection message, or an error message if the script fails to load.
`public/admin/config.yml`	Backend, media folders, and collections

All admin pages are `noindex, nofollow`, and robots disallows `/admin`.

> To upgrade Decap, change the pinned version in `editor-loader.js`, then test sign-in and every collection locally (`npx decap-server`) before deploying. An unpkg outage affects only the editor, never the public site.

> The workspace cards hard-code collection and file `name`s in their URLs. If you rename a collection or file entry in `config.yml`, update the matching card link in `index.html`.

Backend and authentication

```
backend:
  name: github
```

```
  repo: pmgasset/fortadamswebsite
  branch: main
  base_url: https://decap-oauth.fortadams.org   # OAuth proxy
local_backend: true
site_url: https://www.fortadams.org
```

- There's **no** `publish_mode: editorial_workflow`, so every Publish commits straight to `main` and deploys.
- The GitHub backend needs an **OAuth proxy**, because GitHub's OAuth flow requires a client secret. Setup:
  1. Create a **GitHub OAuth App** (org or user Settings → Developer settings → OAuth Apps). Set the callback URL to the proxy's callback endpoint.
  2. Deploy a proxy, for example the Cloudflare Worker `sterlingwes/decap-proxy` or `i40west/netlify-cms-oauth`, with `GITHUB_OAUTH_CLIENT_ID` and `GITHUB_OAUTH_CLIENT_SECRET`, on `decap-oauth.fortadams.org`.
  3. Keep `backend.base_url` pointing at it, and confirm the proxy is live before launch.
- Editors need write access to the repo.

Collections

The header comment in `config.yml` says it: **keep collection, file, and field `name`s stable**, because they map to existing JSON keys and paths. Labels can change freely.

Collection label (`name`)	File entry label (`name`)	Target	Media folder
Special Events (`events`)	Add or update events (`event_list`)	`content/events.json`	`public/images/events`
	Event Categories (`event_categories`)	`content/event-categories.json`	
Alerts & Booking (`settings`)	Update the announcement banner (`alert`)	`content/alert.json`	global: `public/images/staff`
	Update tour booking links (`tour_booking`)	`content/tour-booking.json`	
Staff & Board (`staff`)	Update staff & board (`directory`)	`content/staff.json`	global: `public/images/staff`
Sponsors & Partners (`sponsors`)	Update sponsor recognition (`annual_sponsors`)	`content/sponsors.json`	`public/images/sponsors`

All collections are **file collections** (one JSON file each, with list widgets) and set `delete: false`, so editors can't delete the underlying files. Global options: `editor.preview: false` (no preview pane), `logo_url` (the Trust logo on the login screen), and `display_url` pointing at `/admin/`. Lists are `collapsed: true`, `required: false`, `default: []`.

Notable field configuration:

- **Alert message:** `pattern: ^.{0,120}$`.
- **Tour URLs:** regex patterns (see *FareHarbor Integration*).
- **Event `isPublished`:** `default: false`, so **new events start hidden** until the editor switches them on.
- **Event list:** `add_to_top: true`, `allow_reorder: false` (the site sorts by date).
- **Event category:** a `relation` widget into `event_categories` (`value_field: categories.*.name`). It stores the **name string**, so renames don't cascade.
- **Event dates:** `datetime`, `format: YYYY-MM-DDTHH:mm:ssZ`, `picker_utc: false`, displayed as `MMM D, YYYY` / `h:mm A`. The stored offset comes from the **editor's device time zone**, hence the "set your device to Eastern Time" hint.
- **Event id:** a `hidden` widget, never set by Decap for new entries. `getPublicEvents()` generates one.
- **Event `externalUrl`:** `^$|^https?://[^\s/?#]+[^\s]*$`. **`bookingUrl`:** `^$|^/(?!/)[^\s]*$` (site paths only, no protocol-relative `//`).
- **Staff `email`:** `^$|^[^\s@]+@[^\s@]+\.[^\s@]+$`.
- **Sponsor `year`:** `^[0-9]{4}$`. **Sponsor `website`:** same pattern as event `externalUrl`. `add_to_top: true`.

Adding a CMS-editable field or area

1. Add the field to the JSON file in `content/` with a sensible default, so existing data stays valid.
2. Update the TypeScript interface and loader in `src/lib/`, treating the field as optional for older data.
3. Add the widget to `config.yml`. Mirror the constraints with `pattern`, `required`, and `hint`.
4. Render it in the component.
5. Test locally with `npx decap-server` (see *Local Development Setup*), then `npm run build`.

Keep new areas constrained: validated strings, booleans, lists, and images. Avoid rich-text or HTML fields that let editors break layouts or inject markup.

Build and Deployment

Cloudflare Pages project settings

Setting	Value
Source	GitHub `pmgasset/fortadamswebsite`, production branch `main`
Framework preset	None / Next.js (Static HTML Export)
Build command	`npm run build`
Build output directory	`out`
Node version	20+ (set `NODE_VERSION` if the default is older)
Functions	Auto-detected from the root `functions/` directory

- **Production deploys:** every push to `main`, including every Decap publish.
- **Preview deploys:** every other branch and PR gets a unique `*.pages.dev` URL. Preview uses the *Preview* environment's variables. Set the `GHL_*` secrets there too, ideally pointing at a test pipeline.

Environment variables

Configure them under **Workers & Pages → project → Settings → Variables and Secrets**. `NEXT_PUBLIC_*` values are compiled into the JavaScript bundle, so **changing one requires a redeploy**. The `GHL_*` secrets are read by the Function at request time. Full list: *Reference → Environment Variables*.

Deploying

- **Normal:** merge or push to `main`, then watch *Deployments*.
- **Rebuild without code changes** (for example, to drop past events or pick up a changed env var): *Deployments → latest production deployment → ⋯ → Retry deployment*.
- **Roll back:** *Deployments → a previous successful deployment → ⋯ → Rollback to this deployment*. This is instant. Remember that the next push to `main` deploys the current `main` again, so also revert the bad commit.

Scheduled rebuilds (recommended)

Because "upcoming" events are computed at build time, schedule a daily rebuild:

1. **Pages → Settings → Builds → Deploy hooks**: create a hook for `main` and copy its URL. Treat it as a secret.
2. Call it daily just after midnight Eastern, from a Cloudflare Worker Cron Trigger (`curl -X POST <hook>`) or a GitHub Actions `schedule` workflow.

This hasn't been set up yet; it's tracked in *Known Issues and Open Items*.

Redirects and headers

- `public/_redirects` is copied to `out/_redirects` and applied by Pages. It currently has one rule (`/preservation/restoration-projects/` → `/history/#restoration-projects`, 301) plus commented, **guessed** examples. Populate it from a crawl of the old WordPress site before DNS cutover.
- There's no `_headers` file. Security headers (CSP, HSTS, etc.) aren't configured at the Pages level. If you add a CSP, allow `fareharbor.com`, `cdn.r2.fortadams.org`, `crm.fortadams.org`, the HighLevel webhook hosts, `www.google.com` (map iframe), and `unpkg.com` (for `/admin` only).

Domains

The production canonical URL is `siteConfig.url` (`https://www.fortadams.org`), used for `metadataBase`, the sitemap, and robots. Attach both apex and `www` under **Pages → Custom domains** and redirect one to the other. Confirm the final domain before launch; it's still marked `[PLACEHOLDER]`.

SEO and Accessibility

SEO

- **Metadata:** every page exports `title`, `description`, and `alternates.canonical`. The root layout sets `metadataBase` from `siteConfig.url`, the title template `%s · Fort Adams Trust`, Open Graph defaults, and `twitter.card: summary_large_image`.
- **OG image:** currently `/images/og-default.svg` (placeholder). Many social platforms don't render SVG OG images. Add a real **1200×630 JPG/PNG** and update `layout.tsx`.
- **Sitemap:** `src/app/sitemap.ts` lists routes explicitly. Events get `changeFrequency: weekly`, others monthly; home has priority 1. **Add new routes by hand.**
- **Robots:** `src/app/robots.ts` allows everything except `/admin`, and links the sitemap.
- **Redirects:** `public/_redirects`. Map every indexed WordPress URL before launch to keep search rankings.
- **Headings:** one `h1` per page, with sections labelled by `aria-labelledby` headings.

Accessibility (target: WCAG 2.1 AA)

Implemented patterns to keep when adding features:

Area	Pattern
Landmarks	A skip link to `#main-content` (first focusable element), plus `header`/`main`/`footer`; the alert banner is `role="region"` with a label
Focus	Global brass `:focus-visible` outline; focus moves to confirmation messages after form success; Escape closes the mobile menu
Targets	Buttons and inputs at least 44px (`min-h-11`)
Images	Meaningful `alt` on all content images; `alt=""` for decorative; sponsor logos use "<Name> logo"
Forms	Visible labels bound with `useId`, required markers, `role="alert"` errors, `role="status"` success
Widgets	ARIA on the accordion, sticky tabs, filter pill groups (`role="group"` + `aria-label`), menu, and carousel dots
Media	Explore videos support WebVTT captions (`captionsPath`) and transcripts. **Add captions for all six videos.**

Area	Pattern
Motion	Smooth scroll only when `prefers-reduced-motion: no-preference`
Contrast	Gradient scrims behind text on photos; brass-on-navy and navy-on-sand palettes
Embeds	Titled iframes (Google Map, DAF widget)

When adding UI, check it with keyboard only, with a screen reader (VoiceOver or NVDA), and with Lighthouse/axe.

Conventions and Workflow

Code style

- **TypeScript strict.** Import via the `@/` alias (`@/lib/...`, `@/components/...`).
- **Prettier:** 100-character lines, double quotes, semicolons, trailing commas, Tailwind class sorting. Run `npm run format`.
- **ESLint:** `eslint-config-next` core-web-vitals + TypeScript. Unused bindings prefixed with `_` are allowed.
- **Comments:** explain *why*, especially around integrations and static-export constraints. Match the density of the surrounding file.
- **Placeholders:** mark content awaiting the client with `[PLACEHOLDER: …]` so it can be found with grep.

Architecture rules

- **Server components by default.** Add `"use client"` only for state, effects, or browser APIs.
- **Data in `src/lib`, presentation in `src/components`.** Page-specific prose can live in `page.tsx`.
- **Nothing that needs a server at runtime** except in `functions/`. No route handlers, server actions, `headers()`/`cookies()`, or ISR. They break `output: "export"`.
- **Never expose secrets through `NEXT_PUBLIC_*`.** Anything needing a secret goes in a Pages Function.
- **Keep third-party specifics isolated**: FareHarbor in `content.ts` and `FareharborWidget`; HighLevel in the Function, `LeadForm`, `EmailSignup`, `DonateWidget`, and `support-content.ts`.
- **Payments never touch the site.** Link to provider-hosted checkouts.
- **Validate at the edge of CMS data.** Loaders should tolerate missing optional fields and bad URLs (see `getTourBooking`).
- **Before using a Next.js API, check `node_modules/next/dist/docs/`** (per `AGENTS.md`). Next 16 has breaking changes.

Workflow

1. `git pull` first. Decap commits to `main` frequently.

2. Work on a branch for anything non-trivial. Pushing it gives you a Cloudflare preview URL to share for review.
3. `npm run lint && npm run format:check && npm run build` before merging.
4. Merge to `main`, then verify on production.
5. If a change affects editors or admins (a new CMS field, changed behaviour), update this book (see *Maintaining This Book*).

Commit messages in this repo are short imperative summaries, for example "Link donations to HighLevel hosted checkout" or "Add annual corporate sponsors directory".

Common Development Tasks

Short recipes for the most frequent requests. After each one, run `npm run build` and check the page locally before pushing.

Update hours, season, or tour times

- **Visit page:** the `dl` rows in the *Hours & Season* section of `src/app/visit/page.tsx`, and the intro sentence in `PageHero`.
- **Home quick facts:** `src/components/home/QuickInfoStrip.tsx` ("Open daily, Apr–Nov").
- **FAQ:** "What are the hours of operation?" in `src/lib/faq.ts`.

Update tour cards or prices

Edit `src/lib/tours.ts` (`priceLabel`, `duration`, `highlights`, `badges`). The admission note ("Admission starts at $10...") is in `src/app/visit/page.tsx`, and "Admission from $10" is in `QuickInfoStrip.tsx`.

Add or edit an FAQ

Add an object `{ question, answer }` to the right group in `visitFaqGroups` (`src/lib/faq.ts`). The accordion picks it up automatically.

Change a membership tier

Edit the tier in `membershipTiers` (`src/lib/support-content.ts`): `price`, `benefits`, `tagline`, and `joinUrl` (the HighLevel payment-link ID). Coordinate with the HighLevel product so the price

matches the checkout. Set `featured: true` to highlight a tier.

Change the donation form

Update `DONATION_CHECKOUT_URL` in `src/components/support/DonateWidget.tsx`.

Update contact details or social links

`siteConfig.org` and `siteConfig.social` in `src/lib/site-config.ts`. The footer, DAF panel, and form error messages update automatically.

Add a venue

Add a `Venue` to `src/lib/venues.ts`, and put optimised images in `public/images/venues/`. Venues render in array order.

Add a history chapter, hotspot, or restoration project

Append to `historyChapters`, `mapHotspots`, or `restorationProjects` in `src/lib/history.ts`. Link a chapter to a hotspot with `mapId` = hotspot `id`. Hotspot `x`/`y` are percentages on the fort map. Mark the present-day chapter with `isPresent: true`.

Add a glossary term

Add `{ term, definition }` to `fortTerms` in `src/lib/fort-terminology.ts`. Keep the list alphabetical, because the page groups terms by first letter in array order.

Add a new page

1. Create `src/app/<slug>/page.tsx` with a `metadata` export (`title`, `description`, `alternates.canonical`).
2. Use `PageHero` (or `Breadcrumb` plus a header band), `Container`, and the section and heading patterns from existing pages.
3. Add the route to `routes` in `src/app/sitemap.ts`.
4. Add it to `primaryNav` and/or `footerLinks` in `site-config.ts` if it needs a link.
5. If it replaces an old WordPress URL, add a 301 in `public/_redirects`.

Add a redirect

Append `<old-path> <new-path> 301` to `public/_redirects`. Use trailing slashes on internal targets (the site uses `trailingSlash: true`).

Replace an Explore the Fort video

Upload the new MP4 to R2 under a **new file name**, update `videoPath` in `src/lib/explore-fort.ts`, and optionally add `captionsPath` (VTT) and `posterPath`. The list must stay at exactly six entries.

Add a new webhook-backed form

Reuse `LeadForm` with a new `NEXT_PUBLIC_*_WEBHOOK_URL`, and set `source`, `tags`, and `opportunityPrefix`. Better still, when delivery must be confirmed, add a Pages Function modelled on `event-inquiry.ts`.

Expose something new in the CMS

See *Decap CMS Configuration → Adding a CMS-editable field or area*.