CHATGPT -> GITHUB PAGES

Direct Website Publishing Workflow

How to let ChatGPT inspect your static website repository, build a complete article page, update site indexes, commit the changes, and publish without opening Codex or PowerShell.

Configuration documented from the working brendonrcoleman.com workflow

Version 1.0 - August 8, 2026

Core idea: ChatGPT does not need unrestricted access to your computer. It can publish through a bounded GitHub write capability: read repository files, create or update specific files, commit them to the publishing branch, and let GitHub Pages serve the new state.

1. What happened in the working session

The publication did not run through Codex, PowerShell, WSL, or a local git command. ChatGPT used the connected GitHub capability available in the conversation to inspect the existing site structure, create a new article file, update the article index, and commit both changes directly to GitHub.

Observed publication path

```
ChatGPT conversation
      |
      v
connected GitHub app / tool
      |
      v
noblebrendon-cloud/noblebrendon-cloud.github.io
      |
      +--> read existing article template
      +--> read /articles/index.html
      +--> create /articles/categorical-collapse/index.html
      +--> update /articles/index.html
      |
      v
commit to main
      |
      v
GitHub Pages
      |
      v
https://brendonrcoleman.com/articles/categorical-collapse/
```

In the current repository, the default branch is main. The root CNAME file contains brendonrcoleman.com. The site uses static HTML, a shared stylesheet at /assets/style.css, and Google Analytics with measurement ID G-YY69T9F3G3. The article listing lives at /articles/index.html.

> **Important limitation:** The exact GitHub actions exposed inside ChatGPT can vary by plan, workspace, app version, and permissions. The workflow below distinguishes the ideal direct-write route from a fallback custom-action route.

2. The architecture in one sentence

The model decides what should be written; the tool boundary determines what may be changed; GitHub records what was changed; GitHub Pages publishes the resulting repository state.

```
MODEL REASONING
    |
    v
BOUNDED TOOL CALL
(fetch_file / create_file / update_file)
    |
    v
AUTHORIZED REPOSITORY CHANGE
    |
    v
GIT COMMIT
    |
```

```
    v
PUBLISHED SITE STATE
```

3. Prerequisites

☑️ A GitHub account with admin or push permission to the website repository.

☑️ A GitHub Pages site already publishing from a branch or from a GitHub Actions deployment workflow.

☑️ A stable publishing branch. In the documented site, this is main.

☑️ A known repository structure for articles, assets, navigation, and indexes.

☑️ A connected GitHub app/tool in ChatGPT with repository access.

☑️ For direct publishing, the connected app/tool must expose write actions for repository contents.

☑️ A human confirmation step before consequential external writes.

☑️ A verification method after commit: refetch the file, inspect the index, and check the public route.

Prerequisite matrix

Layer	Required state	Why it matters
ChatGPT	Apps/actions available in the current ChatGPT experience	Provides the external tool boundary.
GitHub authorization	Repository selected and authorized	Prevents broad account-level access.
GitHub permission	Contents: write for direct file publication	Required to create or replace repository files.
Website repository	Static site or deterministic build pipeline	Makes repository state map predictably to site state.
GitHub Pages	Publishing source configured	Turns commits into a public website.
Domain	Custom domain and DNS already working if used	Keeps canonical URLs stable.
Site metadata	Title, canonical, social metadata, schema, analytics conventions	Prevents technically published but malformed pages.

4. Connect GitHub to ChatGPT

OpenAI documents GitHub as a ChatGPT app. The normal connection flow starts from Settings > Apps, where GitHub can be connected through GitHub authorization and repository selection. Availability can differ by plan and experience.

☑️ Open ChatGPT Settings.

☑️ Open Apps (or the current Plugin/App directory surface).

☑️ Locate GitHub and choose Connect.

☑️ Complete the GitHub authorization flow.

☑️ Choose only the repositories ChatGPT actually needs.

☑️ Wait a few minutes if a newly authorized repository does not appear immediately.

☑️ Test a read operation first: ask ChatGPT to identify the default branch and fetch a known file.

> **Direct-write gate:** A connected GitHub app is not automatically a publishing app. Before relying on this workflow, confirm that the ChatGPT environment actually exposes repository write actions such as create_file and update_file, or an equivalent Contents write operation.

5. Permission design

GitHub Apps begin with no permissions by default, and GitHub recommends granting only the minimum permissions required. For repository file creation and replacement through the Contents API, the relevant repository permission is Contents: write. Editing files under .github/workflows can additionally require workflow-related write permission.

Permission	Recommended	Purpose
Metadata: read	Yes	Identify repository and branch information.
Contents: read	Yes	Inspect templates, indexes, CSS references, and current files.
Contents: write	Yes for direct publishing	Create article pages and update indexes.
Workflows: write	No unless needed	Only necessary if the workflow edits files under .github/workflows.
Issues / PRs	No for the basic workflow	Not needed for direct static-site publishing.
Administration	Avoid	Too broad for normal article publication.

For a personal static site, the safest default is repository-scoped access to the one website repository and no broader organization permissions.

6. GitHub Pages publishing prerequisite

GitHub Pages can publish from a branch or through a GitHub Actions workflow. For a simple static site, branch publishing is the shortest path: a commit to the configured branch becomes the source of the website.

Documented site contract

```
Repository: noblebrendon-cloud/noblebrendon-cloud.github.io
Default branch: main
Custom domain file: /CNAME
CNAME contents: brendonrcoleman.com
Article index: /articles/index.html
Article route convention: /articles/<slug>/index.html
Shared stylesheet: /assets/style.css
```

Do not accidentally destroy the domain mapping: When publishing from a branch, keep the root CNAME file intact. In the documented repository it contains only brendonrcoleman.com.

7. Article metadata contract

A page is not complete merely because the body text exists. The publishing workflow should treat metadata as part of the article artifact. The model should inspect one or more existing canonical pages first and preserve the site's conventions.

Required identity metadata

Field	Example / rule	Purpose
HTML <title>	When the Category Becomes the Reality | Brendon R. Coleman	Browser title and search result identity.
meta description	One concise summary of the article	Search and link preview description.
canonical URL	https://brendonrcoleman.com/articles/<slug>/	Declares the canonical public route.
article publish date	2026-08-08	Stable publication timestamp.
author	Brendon R. Coleman	Author identity.
route slug	categorical-collapse	Filesystem and URL identity.

Open Graph metadata

☑️ og:title

☑️ og:description

☑️ og:type = article

☑️ og:url

☑️ og:site_name

☑️ og:image when a canonical social image exists

Twitter/X card metadata

☑️ twitter:card (summary or summary_large_image)

☑️ twitter:title

☑️ twitter:description

☑️ twitter:image when a social image exists

Article structured data (JSON-LD)

Minimum JSON-LD pattern

```
{
 "@context": "https://schema.org",
 "@type": "Article",
```

```
  "headline": "<ARTICLE TITLE>",
  "description": "<ARTICLE DESCRIPTION>",
  "datePublished": "YYYY-MM-DD",
  "dateModified": "YYYY-MM-DD",
  "author": {
    "@type": "Person",
    "name": "Brendon R. Coleman",
    "url": "https://brendonrcoleman.com/"
  },
  "publisher": {
    "@type": "Person",
    "name": "Brendon R. Coleman"
  },
  "mainEntityOfPage": "https://brendonrcoleman.com/articles/<slug>/"
}
```

Site-level metadata and dependencies

Item	Current site convention
Stylesheet	/assets/style.css
Analytics loader	Google tag script
Analytics measurement ID	G-YY69T9F3G3
Primary nav	Home / Articles / Essays / Research / Books
Article index	/articles/index.html
Article body classes	contact-box essay-body
Footer	Shared site footer and current-year script

Rule: Never synthesize a fresh page template from memory when the repository already contains a canonical article template. Fetch the current template first, then adapt it.

8. Standard publication procedure

The sequence below is the repeatable procedure for publishing an article from a ChatGPT conversation directly to this static site.

1. Resolve the target repository - Confirm repository name, default branch, and write access before touching content.

2. Read the site contract - Fetch an existing article page, the article index, CNAME, and any shared style/navigation files needed to preserve conventions.

3. Define the article artifact - Resolve title, subtitle, slug, description, publish date, author, section classification, and article body.

4. Build the complete HTML document - Include metadata, Open Graph, Twitter/X card tags, JSON-LD, stylesheet, analytics, nav, body, and footer.

5. Check route collision - Fetch or search the intended path. If it already exists, do not create a duplicate; decide whether the task is an update.

6. Create the article file - Write /articles/<slug>/index.html with a clear commit message.

7. Refetch the created file - Verify that the write succeeded and the content matches the intended page.

8. Fetch the latest article index - Use the current blob SHA/state so the next write is based on the latest file.

9. Update the article index - Add the new article entry at the correct position and commit serially.

10. Verify repository state - Refetch both files and confirm CNAME remains intact.

11. Verify deployment - Check the GitHub Pages deployment path or public URL after propagation.

12. Record the receipt - Preserve commit SHAs, article path, canonical URL, publication date, and any verification result.

Why writes must be serial

GitHub's repository contents API requires care around concurrent modifications. A safe workflow creates or updates one path, receives the resulting state, then fetches the next path before updating it. This is especially important for index files that may have changed between reads.

```
READ article template
READ articles/index.html
READ CNAME

CREATE new article
  -> receive commit SHA

REFETCH articles/index.html
  -> receive current blob SHA

UPDATE articles/index.html
  -> receive commit SHA

VERIFY both files
```

9. The copy-ready ChatGPT publishing instruction

Use this as a bounded publishing prompt: The model still needs access to an authorized GitHub tool. The prompt does not create permissions by itself.

VERBATIM-SAFE PUBLISHING PROMPT

```
Publish the article from this conversation to my static website.

Repository:
noblebrendon-cloud/noblebrendon-cloud.github.io

Publishing branch:
main

Website:
https://brendonrcoleman.com/

Before writing:
1. Read one current article page to preserve the exact site structure.
2. Read /articles/index.html.
3. Read /CNAME and do not alter it.
4. Preserve /assets/style.css and the existing Google Analytics setup.
5. Resolve the article title, slug, description, date, author, canonical URL, Open Graph metadata, Twitter/X metadata, and Article JSON-LD.

Publishing rules:
- Create the article at /articles/<slug>/index.html.
- Do not overwrite an existing article unless I explicitly asked for an update.
- Update /articles/index.html only after the article file is successfully created.
- Perform repository writes serially.
- Keep source, draft, approval, publication, and resulting state conceptually distinct.
- Use a clear commit message for each write.
- After both commits, refetch the changed files and verify the canonical route, metadata, navigation, analytics tag, and article index entry.
- Return the commit SHA(s), repository paths, and public URL.
```

10. Worked example: the categorical-collapse publication

The August 8 article publication is a concrete example of the workflow. ChatGPT inspected an existing article template and the article index, then created the new article and updated the index.

Operation	Repository path	Result
Read template	/articles/probabilistic-intelligence-deterministic-systems/index.html	Recovered page structure and metadata conventions.
Read index	/articles/index.html	Recovered current article listing structure.
Create article	/articles/categorical-collapse/index.html	New canonical article page.
Update index	/articles/index.html	New article added as latest entry.
Preserve domain	/CNAME	brendonrcoleman.com unchanged.

Commit receipts from the working session:

```
Article creation commit:
b80efd28b07c6edfc2e49be368e091f7f0281614

Articles index update:
dcf5e7104cf233e6303bc58cead5033c97e0b145
```

11. Publication verification checklist

☑️ Article file exists at the intended repository path.
☑️ Canonical URL exactly matches the public route.
☑️ HTML title and H1 are correct.
☑️ Meta description is present.
☑️ Open Graph title, description, type, URL, and site name are present.
☑️ Twitter/X card fields are present.
☑️ JSON-LD parses as an Article object.
☑️ datePublished and dateModified are correct.
☑️ Author identity is correct.
☑️ Shared stylesheet path is unchanged.
☑️ Analytics loader and measurement ID are preserved.
☑️ Primary navigation still works.
☑️ Article index contains the new entry exactly once.
☑️ CNAME remains present and unchanged.
☑️ Commit SHA(s) are recorded.

☑️ Public route resolves after GitHub Pages propagation.

Verification should distinguish three states

1. REPOSITORY WRITE SUCCEEDED
   The file exists in GitHub.

2. DEPLOYMENT SUCCEEDED
   GitHub Pages has published the new repository state.

3. PUBLIC ROUTE VERIFIED
   The canonical URL resolves and the rendered page is correct.

Do not collapse these states: A successful create_file response is not the same thing as a verified public deployment. Keep the repository state, deployment state, and browser-visible state separate.

12. Common failure modes

Failure	Likely cause	Correct response
GitHub visible but no write actions	Current app/surface is read-only	Use a write-capable app/action or the fallback custom action described below.
Repository missing	Repo not authorized or recently connected	Review GitHub app repository selection; allow propagation time.
403 on create/update	Insufficient Contents write permission	Grant only the needed repository Contents write permission.
404 on a known private repo	App installation does not include that repo	Adjust repository access in GitHub app configuration.
409/422 while updating	Stale SHA or conflicting write	Refetch current file state and retry serially.
New page exists but site does not change	Pages source/build mismatch	Check GitHub Pages publishing source or deployment workflow.
Custom domain disappears	CNAME removed during branch publication	Restore root CNAME and verify Pages custom domain settings.
Page renders but previews are bad	Missing OG/Twitter metadata	Repair metadata before considering publication complete.
Article index overwritten	Model rewrote from stale copy	Always refetch index immediately before update.

13. If direct GitHub write actions are unavailable

OpenAI supports custom GPT Actions that connect a GPT to an external API using authentication plus an OpenAPI schema. If the standard GitHub connection in your ChatGPT surface is read-only, a narrowly scoped custom action can expose the GitHub repository contents endpoint instead.

Fallback A: Custom GPT Action

☑️ Create or edit a GPT.

☑️ Open the Actions section and create a new action.

☑️ Choose authentication: API key or OAuth, depending on your design.

☑️ Use a GitHub fine-grained token or GitHub App credential with repository Contents: write on the website repository only.

☑️ Define only the endpoints required for read/create/update. Do not expose broad account-management endpoints.

☑️ Test reads first, then a low-risk write to a test path.

☑️ Require user confirmation before consequential external writes.

> **Credential rule:** Never paste a GitHub token into the conversation, article body, repository file, or prompt. Store credentials only in the action/app authentication configuration.

Minimal action surface

```
GET  /repos/{owner}/{repo}/contents/{path}
PUT  /repos/{owner}/{repo}/contents/{path}

Required write capability:
Repository permission -> Contents: write

For UPDATE:
include the current blob SHA.

For CREATE:
do not send an existing-file SHA.
```

Fallback B: Custom MCP / ChatGPT app

For a larger governed system, a custom app built with MCP can expose higher-level tools such as publish_article instead of exposing GitHub's generic Contents API directly. This is safer because the tool can enforce repository, directory, metadata, and approval rules deterministically.

```
publish_article(
  approved_artifact_id,
  slug,
  repository="noblebrendon-cloud/noblebrendon-cloud.github.io",
  branch="main"
)

Tool implementation checks:
- artifact status == APPROVED
- slug is valid and unused
- destination is under /articles/
- metadata contract passes
- CNAME cannot be modified
- index update is serialized
- commit receipt is stored
```

14. How this becomes the importer-to-website workflow

The direct GitHub publishing pattern maps cleanly onto the governed content architecture. The importer should not hand a model unrestricted filesystem or GitHub authority. It should expose legitimate state transitions as tools.

```
IMPORTER CORPUS
    |
    v
search_corpus()
    |
    v
read_source()
    |
    v
create_candidate()
    |
    v
policy / provenance checks
    |
    v
human approval
    |
    v
APPROVED ARTIFACT
    |
    v
publish_article()
    |
    v
GitHub commit
    |
    v
GitHub Pages
    |
    v
PUBLIC RELEASE RECEIPT
```

Recommended tool boundaries

Tool	May do	Must not do
search_corpus	Return source references and metadata	Mutate source records
read_source	Read authorized source material	Change canonical originals
create_candidate	Create a draft artifact with provenance	Publish
request_review	Move candidate into review state	Self-approve
approve_artifact	Record explicit approval	Publish unless separately authorized
publish_article	Publish only an approved artifact	Publish raw model output
verify_release	Check repository/deployment/public	Rewrite history to make a failed

	route	release appear successful

State machine

```
DRAFT
  |
  v
CANDIDATE
  |
  v
REVIEW
  |
  v
APPROVED
  |
  v
PUBLISH_REQUESTED
  |
  v
COMMITTED
  |
  v
DEPLOYED
  |
  v
VERIFIED

No state is allowed to silently impersonate the next state.
```

15. Metadata as a deterministic gate

A future publish_article tool should reject an artifact before GitHub is touched if required metadata is incomplete. This prevents the model from treating page generation as equivalent to publication readiness.

```
REQUIRED BEFORE PUBLISH

identity:
  title
  slug
  author
  publish_date

web:
  canonical_url
  meta_description

social:
  og_title
  og_description
  og_type
  og_url
  og_site_name

structured_data:
```

```
  schema_type = Article
  headline
  description
  datePublished
  dateModified
  author
  mainEntityOfPage

site_contract:
  stylesheet
  analytics
  nav
  footer
  index_entry

governance:
  source_ids
  approval_status
  approval_actor
  policy_version
```

16. Security and governance rules

Least privilege: Authorize only the website repository, not every repository, unless there is a concrete need.

Directory bounds: A publishing tool should be able to write under /articles/ and the specific index files it owns, not arbitrary repository paths.

Protected files: Treat CNAME, workflow files, secrets, and infrastructure configuration as protected unless a separate change request explicitly targets them.

Serial writes: Do not perform create/update/delete calls against related files concurrently.

Human approval: Keep a confirmation boundary before public publication, especially when the article contains consequential claims.

Provenance: Record source IDs and the artifact state that was approved.

Receipts: Persist commit SHA, path, canonical URL, timestamp, and verification result.

Rollback: Know the prior commit SHA or file blob state so a bad publication can be reverted deterministically.

No credential leakage: Secrets belong in OAuth/app/action configuration, never in prompts or repository content.

17. Fast operating checklist

Preflight: Use this checklist before telling ChatGPT to publish.

☑️ GitHub app connected.

☑️ Correct repository authorized.

☑️ Write action available.

☑️ Repository and branch confirmed.

☑️ Existing article template fetched.

☑️ Article index fetched.

☑️ CNAME fetched and protected.

☑️ Title, slug, description, date, author resolved.

☑️ Canonical URL resolved.

☑️ OG / Twitter / JSON-LD metadata complete.

☑️ Article body approved.

☑️ Create article first.

☑️ Refetch index before updating.

☑️ Commit receipts recorded.

☑️ Public route verified.

18. What this workflow does not require

For the simple static-site publication path documented here, you do not need:

☑️ A local PowerShell session.

☑️ WSL.

☑️ A local git clone.

☑️ A Codex coding session.

☑️ A local development server.

☑️ Manual git add / commit / push commands.

Those tools remain useful for larger development work, local testing, build systems, multi-file refactors, runtime debugging, and repository-wide changes. The direct ChatGPT path is best when the website change is a bounded repository-content operation whose structure is already understood.

19. Version and capability note

> **As of August 8, 2026:** OpenAI's app model supports connected services that may expose write actions, but action availability depends on the app and configuration. The GitHub app is connected through ChatGPT Settings > Apps, while GitHub repository write operations require the relevant GitHub permissions. In the working session documented here, the available GitHub tool exposed create_file and update_file, which is why direct publication was possible.

20. Sources and implementation references

Product capabilities change. Recheck the current OpenAI and GitHub documentation before reproducing the setup in a new account, workspace, or repository.

OpenAI Help - Apps in ChatGPT
Connected apps can search, reference, sync, and in some configurations take write actions. App permissions and action controls determine when actions are available and when confirmation is required.
https://help.openai.com/en/articles/11487775-connectors-in

OpenAI Help - Connecting GitHub to ChatGPT
GitHub is connected from Settings > Apps; repository access is selected during GitHub authorization; availability varies by plan and experience.
https://help.openai.com/en/articles/11145903

OpenAI Help - Configuring actions in GPTs
Custom GPT Actions use authentication plus an OpenAPI schema to connect a GPT to an external API.
https://help.openai.com/en/articles/9442513

GitHub Docs - REST API endpoints for repository contents
The create/update contents endpoint can create or replace repository files. Fine-grained access requires repository Contents: write.
https://docs.github.com/en/rest/repos/contents

GitHub Docs - Choosing permissions for a GitHub App
GitHub Apps have granular repository permissions; minimum required permissions are recommended.
https://docs.github.com/en/apps/creating-github-apps/registering-a-github-app/choosing-permissions-for-a-github-app

GitHub Docs - Configuring a publishing source for GitHub Pages
Pages can publish from a branch or a GitHub Actions workflow.
https://docs.github.com/en/pages/getting-started-with-github-pages/configuring-a-publishing-source-for-your-github-pages-site

GitHub Docs - Troubleshooting custom domains and GitHub Pages
For branch publishing, the CNAME file in the publishing source must remain intact and correctly formatted.
https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/troubleshooting-custom-domains-and-github-pages

Repository facts documented from the current site

Fact	Current value
Repository	noblebrendon-cloud/noblebrendon-cloud.github.io
Default branch	main
Visibility	public
Custom domain	brendonrcoleman.com
CNAME path	/CNAME
Article index	/articles/index.html

New article example	/articles/categorical-collapse/index.html
Shared stylesheet	/assets/style.css
Analytics ID	G-YY69T9F3G3

END OF WORKFLOW GUIDE