

Accessibility Content QA - Guide

Product: Accessibility Content QA, by Mergify Digital

What it is: A local, offline, static-HTML accessibility and content checker. You give it an HTML file (or paste HTML into its local browser UI), and it returns a deterministic 0-100 score, a pass/fail gate, counts, and a list of findings as JSON.

What it is not: It is not a WCAG certification tool, not an ADA/legal compliance certification, and it does not certify any page or site. It checks a useful, defined subset of static-HTML issues only. Copy, reports, and conversations about this product must never claim certification.

Who it's for

- Freelancers and small agencies doing a pre-publish check on static or CMS-exported pages
- Website owners and content managers who want a quick, private check before a page goes live
- Developers who want a scriptable gate in a build / CI step (exit code 0 = pass, 2 = fail)

Its wedge is privacy and workflow: it runs entirely on your machine, makes no network calls, and uploads nothing - useful for client staging pages you cannot send to an online scanner.

Requirements and install

- Python 3.12 or newer. No third-party packages - Python standard library only.
- No installation step. Unzip the package, open a terminal / PowerShell in the unzipped folder, and run the commands below.
- Files in this package: app.py (CLI checker), webui_private_v11.py (local browser UI), test_app.py, test_webui_private_v11.py, README.md, GUIDE.md, LICENSE.txt, MANIFEST.json (file list with SHA-256 hashes).

Verify your copy (optional): compare the SHA-256 hashes in MANIFEST.json against the files in the folder.

Run - CLI

Check a file and pretty-print the JSON report:

```
python app.py .\page.html --pretty
```

```
python3 app.py ./page.html --pretty
```

Read HTML from standard input instead:

```
Get-Content .\page.html -Raw | python app.py - --pretty
```

```
cat ./page.html | python3 app.py - --pretty
```

Exit codes:

- 0 - the gate passes (score ≥ 80 and zero high-severity findings)
- 2 - the gate fails

That makes it usable as a pre-publish / CI gate: fail the step when the tool exits with code 2.

Run - WebUI

The WebUI is a loopback-only paste-HTML interface. It does not crawl URLs, does not execute the HTML you paste, does not save scans, and does not upload anything.

```
python webui_private_v11.py --port 8797
```

Then open, on the same computer only:

http://127.0.0.1:8797/

Paste your HTML source (maximum 1 MiB), submit, and read the result table plus the JSON block on the page. Stop the UI with Ctrl+C in its terminal. It binds to 127.0.0.1 only and rejects non-local Host/Origin requests; there is no public or network endpoint.

WebUI canary state (checked 2026-10-27 on the build machine): the canary package was sealed 2026-10-26 with result PRIVATE_BROWSABLE_PLUGIN_PRODUCT_CANARY_VERIFIED, 13 total unit tests, browser render sample PASS, external distribution false at that time. On 2026-10-27 the UI process was also found listening on 127.0.0.1:8797 and returned HTTP 200. That running instance was an on-demand process on the build machine, not part of this distribution - buyers run the UI themselves with the command above.

What the checks cover - precisely

The CLI and WebUI use the same engine (app.py: analyze_html). The rules, with their codes and severities:

- | Code
- | Severity
- | What it flags
- | html_lang_missing
- | high
- | <html> is missing a non-empty lang attribute
- | image_alt_missing
- | high
- | An has no alt attribute at all
- | form_control_unlabeled
- | high
- | A form control has no label - see label handling below
- | title_missing
- | medium
- | Document <title> is missing or empty
- | heading_first_not_h1
- | medium
- | The first heading in the document is not an H1
- | heading_level_jump
- | medium
- | A heading jumps more than one level (e.g. H2 -> H4)
- | heading_empty
- | medium
- | A heading element has no text
- | link_text_missing
- | medium
- | A link has no text content
- | meta_description_missing
- | low
- | Meta description is missing or empty

| image_alt_empty_non_decorative

| low

| An image has empty alt but is not explicitly marked decorative

| link_text_generic

| low

| Link text is generic: "click here", "here", "read more", "more", "learn more", or "link"

Form-control labelling is checked properly, not just by looking for a <label> nearby. A control counts as labelled if any of these is present and non-empty: a wrapping <label>, a <label for="?"> matching the control's id, aria-label, aria-labelledby (where every referenced element has text), or a title attribute. Hidden, submit, button, reset, and image inputs are excluded.

Decorative images: an empty alt is allowed when the image is explicitly decorative - role="presentation" or role="none", or aria-hidden="true".

What it does NOT check

- Color con

- ARIA validity beyond the label handling described above
- Keyboard navigation, focus order, or any interactive behaviour
- JavaScript-rendered DOM - it analyses the static HTML source you give it only
- Anything on a live URL - it does not fetch, crawl, or scan websites

Reading the report

Example JSON shape:

```
{
  "score": 82,
  "pass": true,
  "summary": {
    "issues": 3,
    "by_severity": { "high": 0, "medium": 1, "low": 2 },
    "images": 4,
    "links": 12,
    "form_controls": 2
  },
  "issues": [
    { "code": "meta_description_missing", "severity": "low", "message": "Meta description is missing or empty." }
  ]
}
```

- score - Starts at 100. Each finding subtracts a fixed penalty: high = 18, medium = 10, low = 4. The score never goes below 0. The same HTML always produces the same score.

- pass - true only when score >= 80 and there are zero high-severity findings. A page can score above 80 and still fail if it has a high-severity finding.

- summary - Total issue count, counts by severity, and how many images, links, and form controls were seen.

- issues - One entry per finding: a stable code (use this in your own tracking), its severity, and a plain-English message.

Fix high-severity findings first, re-run, and use the exit code in your publish checklist.

Troubleshooting

- python not found / wrong version - Install Python 3.12+ and re-open the terminal. On Windows, `py app.py .\page.html --pretty` also works.
- Exit code 2 with no visible error - That is the tool working as designed: the gate failed. Read the issues array in the JSON output.
- WebUI port already in use - Pick another port, e.g. `python webui_private_v11.py --port 8798`, and open that port's address instead. Valid ports are 1024-65535.
- WebUI says "Local origin required" / 403 - Open it exactly as `http://127.0.0.1:<port>/` on the same machine. It deliberately rejects other hosts and origins.
- WebUI rejects the paste (413 / oversized) - Input is capped at 1 MiB. Split large pages or check the exported single-page HTML instead.
- A page that "looks fine" scores low - The tool reads static source only. Content injected later by JavaScript is not in the source it sees, so check the exported / view-source HTML of the page.
- Tests - Run them yourself any time: `python -m unittest -v test_app test_webui_private_v11` - expected: 13 tests, OK (re-verified 2026-10-27: test_app 3/3, test_webui_private_v11 10/10).

Scope and honesty note

This tool covers a useful subset of accessibility and content checks on static HTML. It does not provide complete WCAG conformance testing and it is not legal or accessibility certification. For a full audit - rendered-DOM scanning, contrast, keyboard testing, and manual review with assistive technology - you still need those tools and that review. This product's job is the fast, private, repeatable pre-publish gate in front of them.