Lighthouse Web page quality audits

START HERE

`lighthouse <url>`
Audit a page, write HTML report

`lighthouse <url> --view`
Run and open the report

`lighthouse <url> --output=json --output-path=./report.json`
Save machine-readable result

`lighthouse <url> --only-categories=performance,seo`
Limit to some categories

`lighthouse <url> --preset=desktop`
Desktop scoring and emulation

`lighthouse <url> --quiet --chrome-flags="--headless"`
Headless, no progress logs

`lighthouse <url> --throttling-method=provided`
Measure without simulated throttling

`lighthouse --list-all-audits`
List every audit id

Gather load page in Chrome, collect artifacts → `artifacts` → **Audit** run audits on artifacts → `scores` → **Categories** performance, a11y, SEO ... → `reporters` → **Report** html / json / csv

CONCEPTS

Default run is mobile emulation with simulated slow 4G and a slowed CPU, so scores are lower than what a fast laptop feels. Use --preset=desktop for desktop.

Scores vary between runs. Close other tabs, run several times and compare medians before trusting a small change.

Needs Node 22+ and an installed Chrome or Chromium. Point CHROME_PATH at a specific binary.

RUNNING

`lighthouse <url>`
Audit a URL, write an HTML report to the cwd

`lighthouse <url> --view`
Open the HTML report when done

`lighthouse --url-file=<file> --output-path=<dir>`
Audit every URL listed in a file

`lighthouse <url> --fail-fast`
Stop remaining URLs after the first failure
Use with --url-file.

`lighthouse <url> --max-wait-for-load <ms>`
Cap how long to wait for page load

`lighthouse <url> --ignore-status-code`
Warn instead of fail on HTTP error status

`lighthouse <url> --verbose`
Show debug logging

`lighthouse <url> --quiet`
Hide progress, debug logs and errors

OUTPUT

`--output=<html|json|csv>`
Pick the report format

`--output=json --output=html`
Write several formats in one run

OUTPUT (cont.)

`--output-path=<file|dir|stdout>`
Where to write the report
With several outputs the path gets the standard extension per format. Directories must already exist.

`--view`
Open the HTML report in a browser

`--save-assets`
Save trace and devtools log to disk

`--disable-full-page-screenshot`
Skip the large full-page screenshot

`--locale <code>`
Format the report in another language

JSON output goes to stdout by default; HTML and CSV go to a file named from the URL and date.

CATEGORIES & AUDITS

`--only-categories=<a,b>`
Run only these categories
performance, accessibility, best-practices, seo, agentic-browsing.

`--only-audits <id> ...`
Run only the named audits

`--skip-audits <id> ...`
Run everything except these audits

`--list-all-audits`
Print all audit ids and exit

`--preset=<name>`
Built-in config: perf, desktop, experimental
Ignored when --config-path is given.

`--config-path=<file>`
Use your own config (custom audits, categories)

`--plugins <name> ...`
Run Lighthouse plugins

`--blocked-url-patterns <pattern> ...`
Block matching network requests

`--extra-headers <json|file>`
Send extra HTTP headers (cookie, auth)

CATEGORIES & AUDITS (cont.)

`--disable-storage-reset`
Keep cache and storage from before the run

`--cli-flags-path=<file>`
Read flags from a JSON file

EMULATION & THROTTLING

`--preset=desktop`
Desktop viewport, desktop metrics scoring

`--form-factor=<mobile|desktop>`
Set scoring form factor only

`--throttling-method=<method>`
Throttling: simulate, devtools, provided
simulate is the default; provided applies none.

`--throttling.cpuSlowdownMultiplier=<n>`
Set CPU slowdown

`--throttling.rttMs=<ms>`
Set simulated network round-trip time

`--throttling.throughputKbps=<kbps>`
Set simulated network throughput

`--throttling.requestLatencyMs=<ms>`
Emulated request latency (devtools method)

`--throttling.downloadThroughputKbps=<kbps>`
Emulated download speed (devtools method)

`--throttling.uploadThroughputKbps=<kbps>`
Emulated upload speed (devtools method)

`--screenEmulation.disabled`
Turn off screen emulation

CHROME

`--chrome-flags="<flags>"`
Pass flags to the launched Chrome

`--chrome-flags="--headless"`
Run without a visible window

CHROME (cont.)

`--chrome-flags="--no-sandbox"`
Needed in many containers and CI
Only where the sandbox cannot run, e.g. root in Docker.

`--port <n>`
Debugging port; 0 picks a random one

`CHROME_PATH=<path> lighthouse <url>`
Use a specific Chrome binary

GATHER & AUDIT SEPARATELY

`lighthouse <url> -G`
Gather only, save artifacts to ./latest-run

`lighthouse <url> -A`
Audit saved artifacts without a browser

For CI budgets and history across commits, use Lighthouse CI (GoogleChrome/lighthouse-ci) instead of scripting the CLI.

RECIPES

Score as JSON, extract the performance score

```
lighthouse <url> --output=json --output-path=stdout --quiet --chrome-flags="--headless" \
  | jq '.categories.performance.score'
```

HTML and JSON side by side

```
lighthouse <url> --output=html --output=json --output-path=./reports/home
# writes reports/home.report.html and reports/home.report.json
```

Desktop run in a container

```
lighthouse <url> --preset=desktop --only-categories=performance,accessibility \
  --chrome-flags="--headless --no-sandbox" --output=json --output-path=./out.json
```

Authenticated page without throttling

```
lighthouse <url> --extra-headers=./headers.json \
  --throttling-method=provided --screenEmulation.disabled --no-emulatedUserAgent
```