**Updated:** 5/2/2020 11:04 PM

# Overview

* Provides a command line interface for the Google Lighthouse engine to run quick accessibility tests using rulesets from a version of the axe-core library
* Core requirement is an installation of the Google Chrome desktop browser

# Software required

## Lighthouse

* Node.js: [Long time support version](https://github.com/nodejs/Release) (Status = Active LTS)
  + <https://nodejs.org/en/>
* Google Chrome browser for Desktop
  + <https://www.google.com/chrome/>
* Global installation of Lighthouse
  + <https://www.npmjs.com/package/lighthouse>

## Lighthouse-batch

* Node v10.13+.
  + <https://nodejs.org/en/>
* Google Chrome browser for Desktop
  + <https://www.google.com/chrome/>
* Global installation of Lighthouse-batch
  + <https://www.npmjs.com/package/lighthouse-batch>

# Installation

## Lighthouse

* Long Time support version of Node.js (see above: software required) version 10+
* Google Chrome for Desktop (see above)
* Install Lighthouse globally:
  + npm install -g lighthouse

## Lighthouse-batch

* Node 10.13+
* Google Chrome for Desktop (see above)
* Install lighthouse-batch globally:
  + npm install lighthouse-batch -g

# Usage and Syntax

## Lighthouse

lighthouse <url>

Logging:

--verbose Displays verbose logging [boolean]

--quiet Displays no progress, debug logs or errors [boolean]

Configuration:

--save-assets Save the trace & devtools log to disk [boolean]

--list-all-audits Prints a list of all available audits and exits [boolean]

--list-trace-categories Prints a list of all required trace categories and exits [boolean]

--print-config Print the normalized config for the given config and options, then exit. [boolean]

--additional-trace-categories Additional categories to capture with the trace (comma-delimited).

--config-path The path to the config JSON.

An example config file: lighthouse-core/config/lr-desktop-config.js

--chrome-flags Custom flags to pass to Chrome (space-delimited). For a full list of flags, see

http://peter.sh/experiments/chromium-command-line-switches/.

Environment variables:

CHROME\_PATH: Explicit path of intended Chrome binary. If set must point to an executable of a build of

Chromium version 66.0 or later. By default, any detected Chrome Canary or Chrome (stable) will be launched.

[default: ""]

--port The port to use for the debugging protocol. Use 0 for a random port [default: 0]

--preset Use a built-in configuration. [choices: "full", "perf", "mixed-content"]

WARNING: If the --config-path flag is provided, this preset will be ignored.

--hostname The hostname to use for the debugging protocol. [default: "localhost"]

--max-wait-for-load The timeout (in milliseconds) to wait before the page is considered done loading and the run should continue.

WARNING: Very high values can lead to large traces and instability [default: 45000]

--emulated-form-factor Controls the emulated device form factor (mobile vs. desktop) if not disabled [choices: "mobile", "desktop", "none"] [default: "mobile"]

--enable-error-reporting Enables error reporting, overriding any saved preference. --no-enable-error-reporting will do the opposite. More:

https://git.io/vFFTO

--gather-mode, -G Collect artifacts from a connected browser and save to disk. If audit-mode is not also enabled, the run will quit

early. [boolean]

--audit-mode, -A Process saved artifacts from disk [boolean]

Output:

--output Reporter for the results, supports multiple values [choices: "json", "html", "csv"] [default: "html"]

--output-path The file path to output the results. Use 'stdout' to write to stdout.

If using JSON output, default is stdout.

If using HTML or CSV output, default is a file in the working directory with a name based on the test URL and date.

If using multiple outputs, --output-path is appended with the standard extension for each output type. "reports/my-run" -> "reports/my-run.report.html", "reports/my-run.report.json", etc.

Example: --output-path=./lighthouse-results.html

--view Open HTML report in your browser [boolean]

Options:

--help Show help [boolean]

--version Show version number [boolean]

--cli-flags-path The path to a JSON file that contains the desired CLI flags to apply.

Flags specified at the command line will still override the file-based ones.

--blocked-url-patterns Block any network requests to the specified URL patterns [array]

--disable-storage-reset Disable clearing the browser cache and other storage APIs before a run [boolean]

--throttling-method Controls throttling method [choices: "devtools", "provided", "simulate"]

--throttling.rttMs Controls simulated network RTT (TCP layer)

--throttling.throughputKbps Controls simulated network download throughput

--throttling.requestLatencyMs Controls emulated network RTT (HTTP layer)

--throttling.downloadThroughputKbps Controls emulated network download throughput

--throttling.uploadThroughputKbps Controls emulated network upload throughput

--throttling.cpuSlowdownMultiplier Controls simulated + emulated CPU throttling

--extra-headers Set extra HTTP Headers to pass with request [string]

Examples:

lighthouse <url> --view Opens the HTML report in a browser after the run completes

lighthouse <url> --config-path=./myconfig.js Runs Lighthouse with your own configuration: custom audits, report

generation, etc.

lighthouse <url> --output=json --output-path=./report.json --save-assets Save trace, devtoolslog, and named JSON report.

lighthouse <url> --emulated-form-factor=none Disable device emulation and all throttling.

--throttling-method=provided

lighthouse <url> --chrome-flags="--window-size=412,660" Launch Chrome with a specific window size

lighthouse <url> --quiet --chrome-flags="--headless" Launch Headless Chrome, turn off logging

lighthouse <url> --extra-headers "{\"Cookie\":\"monster=blue\"}" Stringify\'d JSON HTTP Header key/value pairs to send in requests

lighthouse <url> --extra-headers=./path/to/file.json Path to JSON file of HTTP Header key/value pairs to send in requests

lighthouse <url> --only-categories=performance,pwa Only run the specified categories. Available categories: accessibility,

best-practices, performance, pwa, seo.

For more information on Lighthouse, see https://developers.google.com/web/tools/lighthouse/.

Output Examples

lighthouse

# saves `./<HOST>\_<DATE>.report.html`

lighthouse --output json

# json output sent to stdout

lighthouse --output html --output-path ./report.html

# saves `./report.html`

# NOTE: specifying an output path with multiple formats ignores your specified extension for \*ALL\* formats

lighthouse --output json --output html --output-path ./myfile.json

# saves `./myfile.report.json` and `./myfile.report.html`

lighthouse --output json --output html

# saves `./<HOST>\_<DATE>.report.json` and `./<HOST>\_<DATE>.report.html`

lighthouse --output-path=~/mydir/foo.out --save-assets

# saves `~/mydir/foo.report.html`

# saves `~/mydir/foo-0.trace.json` and `~/mydir/foo-0.devtoolslog.json`

lighthouse --output-path=./report.json --output json

# saves `./report.json`

Lifecycle Examples

You can run a subset of Lighthouse's lifecycle if desired via the --gather-mode (-G) and --audit-mode (-A) CLI flags.

lighthouse http://example.com -G

# launches browser, collects artifacts, saves them to disk (in `./latest-run/`) and quits

lighthouse http://example.com -A

# skips browser interaction, loads artifacts from disk (in `./latest-run/`), runs audits on them, generates report

lighthouse http://example.com -GA

# Normal gather + audit run, but also saves collected artifacts to disk for subsequent -A runs.

# You can optionally provide a custom folder destination to -G/-A/-GA. Without a value, the default will be `$PWD/latest-run`.

lighthouse -GA=./gmailartifacts https://gmail.com

## Lighthouse-batch

**lighthouse-batch [options]**

**Options:**

**-s, --sites [sites] a comma delimited list of site urls to analyze with Lighthouse**

**-f, --file [path] an input file with a site url per-line to analyze with Lighthouse**

**-p, --params <params> extra parameters to pass to lighthouse cli for each execution e.g. -p "--perf --quiet"**

**-h, --html generate an html report alongside the json report**

**-o, --out [out] the output folder to place reports, defaults to './report/lighthouse'**

**-g, --use-global use a global lighthouse install instead of the dependency version**

**-v, --verbose enable verbose logging**

**--help output usage information**

**Example usage**

**lighthouse-batch -s https://www.bbc.com,https://housing.com**

**This will generate the following files under the /report/lighthouse folder.**

**www\_bbc\_com.report.json // Full results for bbc.com**

**housing\_com.report.json // Full results from housing.com**

**summary.json // Summary of results with scores out of 100**

**Example summary.json**

**NOTE:**

**Chrome is run with the following flags to support the widest set of execution environments, including docker containers --chrome-flags="--no-sandbox --headless --disable-gpu".**

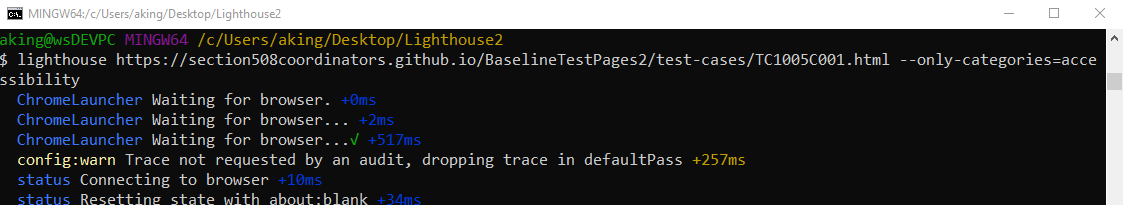
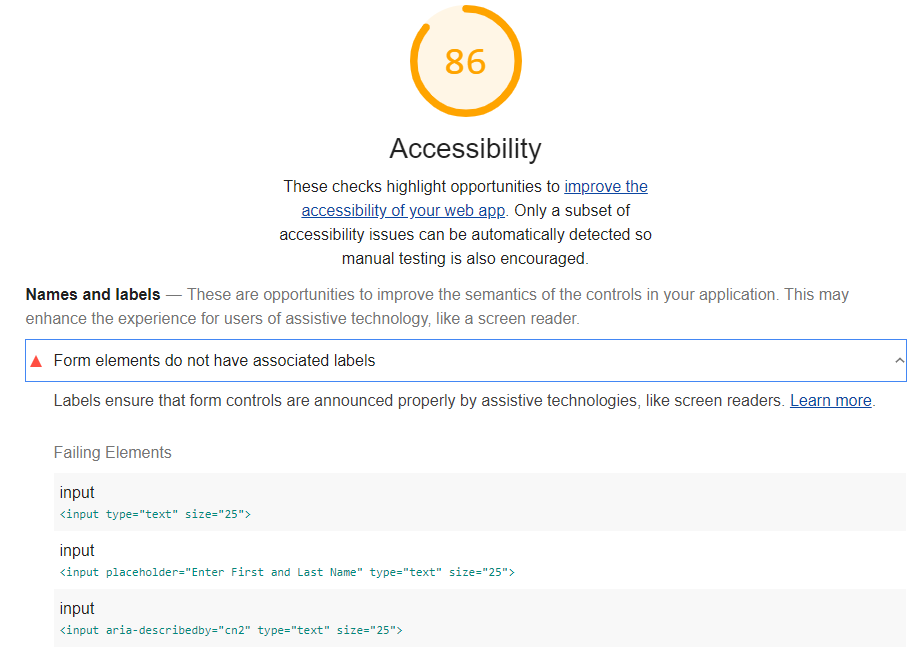
**You can replace these with your own by passing --chrome-flags as extra parameters. e.g.**

**--params "--chrome-flags=\"--no-sandbox --disable-gpu\""**

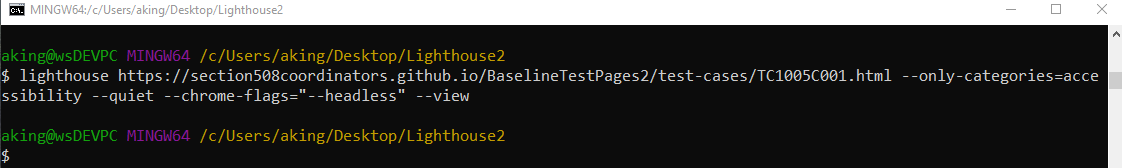
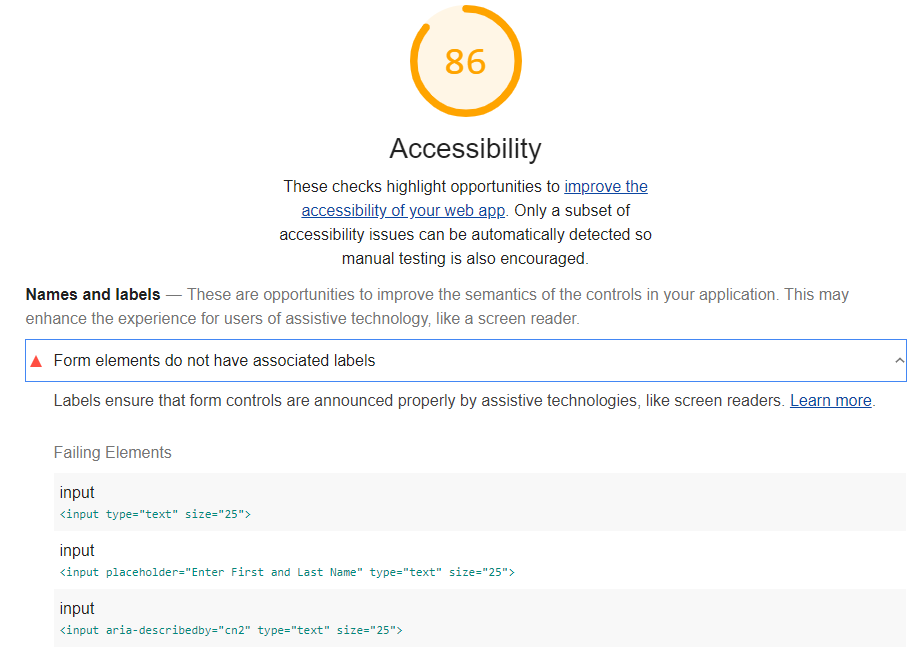
# Examples

## Lighthouse-CLI-Command-Ex-01

Runs only the audit (test) for accessibility only against one URL, launches a visual instance of Chrome briefly while testing the page then closes chrome and writes a report to your local machine file system. All the while the command window scrolls progress log of all processing going on as it happens:

lighthouse https://section508coordinators.github.io/BaselineTestPages2/test-cases/TC1005C001.html --only-categories=accessibility   
***Console window output:*** ***Report that is written to file system:***

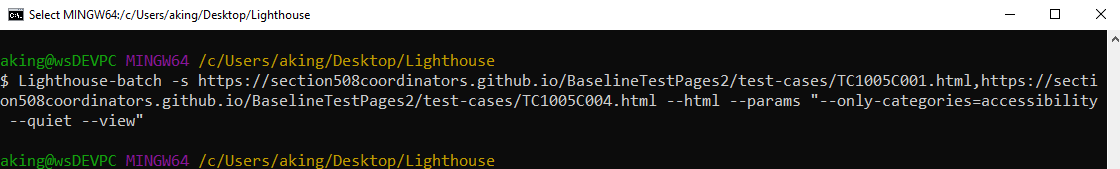
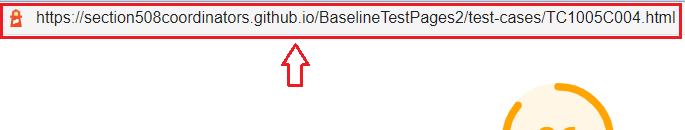
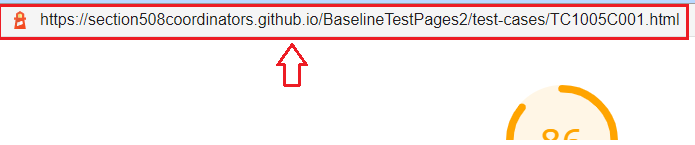
## Lighthouse-CLI-Command-Ex-02

Runs only the audits (tests) for accessibility against one URL, runs a headless instance of Chrome, sends no processing feedback to command window, and automatically opens the report written to the file system in your browser when done:  
lighthouse https://section508coordinators.github.io/BaselineTestPages2/test-cases/TC1005C001.html --only-categories=accessibility --quiet --chrome-flags="--headless" --view   
***Console window output:*** ***Report that opens automatically in browser:***

## Lighthouse-batch-Command-Ex-03

Runs only the audits (tests) for accessibility against multiple URLs, runs a headless instance of Chrome, sends no processing feedback to command window, and automatically opens all reports written to the file system in your browser as they finish:

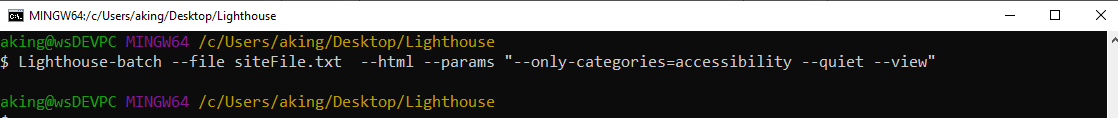
Lighthouse-batch -s https://section508coordinators.github.io/BaselineTestPages2/test-cases/TC1005C001.html,https://section508coordinators.github.io/BaselineTestPages2/test-cases/TC1005C004.html --html --params "--only-categories=accessibility --quiet --view"

***Console window output:***  
***Subsequent HTML results files with URLs in top left to indicate which site URL it is the results for:***   
  


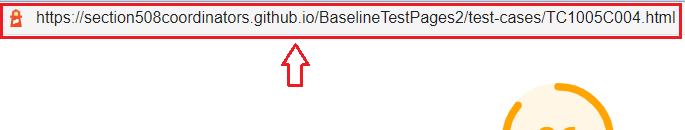
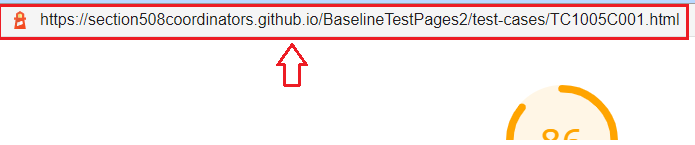
## Lighthouse-CLI-Command-Ex-04

Runs a batch of multiple URLs that are pulled from a separate ***siteFile.txt*** file of URLs, only the audits (tests) for accessibility, runs a headless instance of Chrome, sends no processing feedback to command window, and automatically opens all reports written to the file system in your browser as they finish:

Lighthouse-batch --file siteFile.txt --html --params "--only-categories=accessibility --quiet --view"

***Console window output:***  
 ***Contents of siteFile.txt (5 URLs):***

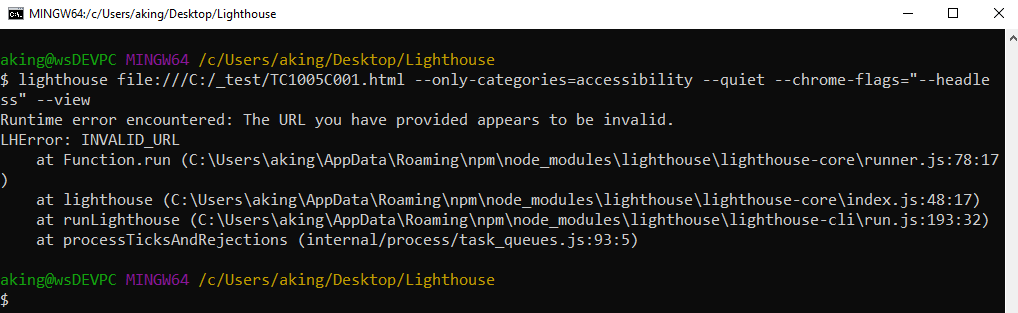
https://section508coordinators.github.io/BaselineTestPages2/test-cases/TC1112A001.html  
https://section508coordinators.github.io/BaselineTestPages2/test-cases/TC1112A002.html  
https://section508coordinators.github.io/BaselineTestPages2/test-cases/TC0813C001.html  
https://section508coordinators.github.io/BaselineTestPages2/test-cases/TC1005C004.html  
https://section508coordinators.github.io/BaselineTestPages2/test-cases/TC1005C005.html

***Subsequent HTML results files with URLs in top left to indicate which site URL it is the results for:***  ******  


## Lighthouse-batch-Command-Ex-05

The example below shows an attempt to test with Lighthouse against files located on a local PC file system. The job fails. Google Lighthouse can only process against files hosted on a web server via HTTP/HTTPS protocols:

lighthouse file:///C:/\_test/TC1005C001.html --only-categories=accessibility --quiet --chrome-flags="--headless" --view

***Console window output:***  
  
***Error Message that displays using Lighthouse within the DevTools Interface in the Chrome Browser:***   
