Running Lighthouse Programmatically

In addition to the DevTools menu and the Chrome extension and the Lighthouse CLI, there are ways to use Lighthouse programmatically in ways that would be too time consuming to do with the lighthouse tools aready available.

The simplest example will run all Lighthouse reports for the given URL using the default options for Lighthouse.

```
function launchChromeAndRunLighthouse(url, opts, config = null) {
  return chromeLauncher.launch({chromeFlags: opts.chromeFlags})
    .then((chrome) => {
      opts.port = chrome.port;
      return lighthouse(url, opts, config)
        .then((results) => {
          return chrome.kill()
            .then(() => results.lhr)
        });
    });
}
```

The lighthouse configurations docs provides a detailes description of all the options that you can pass to Lighthouse.

There are sample configuration objects to show specific customizations of the default configuration.

This example will run the accessibility audits for a desktop-class device and produce an HTML report of the results.

```
const lighthouseOptions = {
  extends: 'lighthouse:default',
  settings: {
    onlyCategories: ['accessibility'],
    emulatedFormFactor:'desktop',
```

```
    output: ['html'],
  },
}

modules.export = lighthouseOptions;
```

While the next example runs the performance audits and throttles using Chrome DevTools.

```
const perfConfig = {
  extends: 'lighthouse:default',
  settings: {
    throttlingMethod: 'devtools',
    onlyCategories: ['performance'],
  },
};

module.exports = perfConfig;
```

You can have multiple functions to run each set of audits separately or you can run them all in one function. It's completely up to you.

Using Lighthouse programmatically offers a more complete example using the built-in fs module, Lighthouse, and Chrome Launcher

```
async function runLighthouse() {
  const chrome = await chromeLauncher.launch({chromeFlags: ['--headless']]
  const options = {
    logLevel: 'info',
    output: 'html',
    onlyCategories: ['performance'],
    port: chrome.port};
  const runnerRe'info' await lighthouse('https://publishing-project.riven

  'https://publishing-project.rivendellweb.net'// `.report` is the HTML re

  // `.lhr` is the Lighthouse Result as a JS ob'lhreport.html'// `.lhr` is
```

```
  console.log('Report is done for', runnerResult.lhr.finalUrl);
  console.log('Performance score was', runnerResult.lhr.categories.perfor

  await chrome.kill();
}

runLighthouse();
```

Chrome runs on headless mode by passing `--headless` to Chrome Layncher.

We then build an options object to run the performance audits, we write the results to an HTML file (`lhreport.html`) and log it to console.

The final task is to kill the headless Chrome instance we create to run Lighthouse.

This gives us a way to run repeated test against out sites; however it doesn't guarantee the same results every time.