PatNeedham / google-it Public

Code Issues 17 Pull requests 2 Actions Projects Security

v1.6.2

google-it / src / googleIt.js / <> Jump to

This commit does not belong to any branch on this repository, and may belong to a fork outside of the repository.

PatNeedham fix query selector change ✓ History

4 contributors

219 lines (203 sloc) | 5.88 KB

```
/* eslint-disable no-console */
/* eslint-disable array-callback-return */
const request = require('request');
const fs = require('fs');
const querystring = require('querystring');
const cheerio = require('cheerio');
require('colors');
const { exec } = require('child_process');

const {
  getDefaultRequestOptions,
  getTitleSelector,
  getLinkSelector,
  getSnippetSelector,
  getResultStatsSelector,
  getResultCursorSelector,
  logIt,
  saveToFile,
  saveResponse,
} = require('./utils');

export function errorTryingToOpen(error, stdout, stderr) {
  if (error) {
    console.log(`Error trying to open link in browser: ${error}`);
    console.log(`stdout: ${stdout}`);
    console.log(`stderr: ${stderr}`);
  }
}

export function openInBrowser(open, results) {
  if (open !== undefined) {
    // open is the first X number of links to open
    results.slice(0, open).forEach((result) => {
      exec(`open ${result.link}`, errorTryingToOpen);
    });
  }
}

export function getSnippet(elem) {
  // recursive function to get "all" the returned data from Google
  function findData(child) {
    if (!child.data) {
      return child.children.map((c) => c.data || findData(c));
    }
    return child.data;
  }

  // Issue with linter wanting "new" before "Array"
  // in this case, the casting is legit, we don't want a new array
  // eslint-disable-next-line unicorn/new-for-builtins
  return elem.children && elem.children.length > 0 ? elem.children.map((child) => Array(findData(child)).join('')).join('') : '';
}

export function display(results, disableConsole, onlyUrls) {
  logIt('\n', disableConsole);
  results.forEach((result) => {
    if (onlyUrls) {
      logIt(result.link.green, disableConsole);
    } else if (result.title) {
      logIt(result.title.blue, disableConsole);
      logIt(result.link.green, disableConsole);
      logIt(result.snippet, disableConsole);
      logIt('\n', disableConsole);
    } else {
      logIt('Result title is undefined.');
    }
  });
}

export const parseGoogleSearchResultUrl = (url) => {
  if (!url) {
    return undefined;
  }
  if (url.charAt(0) === '/') {
```

```javascript
    return querystring.parse(url).url;
  }
  return url;
};

export function getResults({
  data,
  noDisplay,
  disableConsole,
  onlyUrls,
  titleSelector,
  linkSelector,
  snippetSelector,
  resultStatsSelector,
  cursorSelector,
}) {
  const $ = cheerio.load(data);
  let results = [];

  const titles = $(getTitleSelector(titleSelector)).contents();
  titles.each((index, elem) => {
    if (elem.data) {
      results.push({ title: elem.data });
    } else {
      results.push({ title: elem.children[0].data });
    }
  });

  $(getLinkSelector(linkSelector)).map((index, elem) => {
    if (index < results.length) {
      results[index] = Object.assign(results[index], {
        link: parseGoogleSearchResultUrl(elem.attribs.href),
      });
    }
  });

  $(getSnippetSelector(snippetSelector)).map((index, elem) => {
    if (index < results.length) {
      results[index] = Object.assign(results[index], {
        snippet: getSnippet(elem),
      });
    }
  });

  if (onlyUrls) {
    results = results.map((r) => ({ link: r.link }));
  }
  if (!noDisplay) {
    display(results, disableConsole, onlyUrls);
  }

  const resultStats = $(getResultStatsSelector(resultStatsSelector)).html() || '';
  const approximateResults = ((resultStats.split(' results') || [''])[0].split('About ')[1] || '').replace(',', '');
  const seconds = parseFloat((resultStats.split(' (')[1] || '').split(' seconds')[0]);
  const cursor = $(getResultCursorSelector(cursorSelector)).html() || '';
  const page = parseInt(cursor.split('</span>')[1], 10);
  const stats = {
    page,
    approximateResults,
    seconds,
  };
  return { results, stats };
}

export function getResponse({
  fromFile: filePath,
  fromString,
  options,
  htmlFileOutputPath,
  query,
  limit,
  userAgent,
  start,
  includeSites,
  excludeSites,
}) {
  // eslint-disable-next-line consistent-return
  return new Promise((resolve, reject) => {
    if (filePath) {
      fs.readFile(filePath, (err, data) => {
        if (err) {
          return reject(new Error(`Erorr accessing file at ${filePath}: ${err}`));
        }
        return resolve({ body: data });
      });
    } else if (fromString) {
      return resolve({ body: fromString });
    }
    const defaultOptions = getDefaultRequestOptions({
      limit, query, userAgent, start, includeSites, excludeSites,
    });
    request({ ...defaultOptions, ...options }, (error, response, body) => {
      if (error) {
        return reject(new Error(`Error making web request: ${error}`));
      }
      saveResponse(response, htmlFileOutputPath);
      return resolve({ body, response });
    });
  });
}

function googleIt(config) {
  const {
    output,
    open,
    returnHtmlBody,
    titleSelector,
    linkSelector,
    snippetSelector,
    resultStatsSelector,
    cursorSelector,
    start,
    diagnostics,
  } = config;
  return new Promise((resolve, reject) => {
    getResponse(config).then(({ body, response }) => {
      const { results, stats } = getResults({
        data: body,
        noDisplay: config['no-display'],
        disableConsole: config.disableConsole,
        onlyUrls: config['only-urls'],
        titleSelector,
        linkSelector,
        snippetSelector,
        resultStatsSelector,
        cursorSelector,
        start,
      });
      const { statusCode } = response;
      if (results.length === 0 && statusCode !== 200 && !diagnostics) {
        reject(new Error(`Error in response: statusCode ${statusCode}. To see the raw response object, please include the 'diagnostics: true' as part of the options object (or -d i
      }
      saveToFile(output, results);
      openInBrowser(open, results);
      if (returnHtmlBody || diagnostics) {
        return resolve({
          results, body, response, stats,
        });
      }
      return resolve(results);
    }).catch(reject);
  });
}

export default googleIt;
```