PatNeedham fix query selector change ✓ History

2 contributors

287 lines (248 sloc) | 9.02 KB

```
"use strict";

Object.defineProperty(exports, "__esModule", {
  value: true
});
exports.errorTryingToOpen = errorTryingToOpen;
exports.openInBrowser = openInBrowser;
exports.getSnippet = getSnippet;
exports.display = display;
exports.getResults = getResults;
exports.getResponse = getResponse;
exports.default = exports.parseGoogleSearchResultUrl = void 0;

function ownKeys(object, enumerableOnly) { var keys = Object.keys(object); if (Object.getOwnPropertySymbols) { var symbols = Object.getOwnPropertySymbols(object); if (enumerableOn

function _objectSpread(target) { for (var i = 1; i < arguments.length; i++) { var source = arguments[i] != null ? arguments[i] : {}; if (i % 2) { ownKeys(Object(source), true).for

function _defineProperty(obj, key, value) { if (key in obj) { Object.defineProperty(obj, key, { value: value, enumerable: true, configurable: true, writable: true }); } else { obj

/* eslint-disable no-console */

/* eslint-disable array-callback-return */
var request = require('request');

var fs = require('fs');

var querystring = require('querystring');

var cheerio = require('cheerio');

require('colors');

var _require = require('child_process'),
    exec = _require.exec;

var _require2 = require('./utils'),
    getDefaultRequestOptions = _require2.getDefaultRequestOptions,
    getTitleSelector = _require2.getTitleSelector,
    getLinkSelector = _require2.getLinkSelector,
    getSnippetSelector = _require2.getSnippetSelector,
    getResultStatsSelector = _require2.getResultStatsSelector,
    getResultCursorSelector = _require2.getResultCursorSelector,
    logIt = _require2.logIt,
    saveToFile = _require2.saveToFile,
    saveResponse = _require2.saveResponse;

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

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

function getSnippet(elem) {
  // recursive function to get "all" the returned data from Google
  function findData(child) {
    if (!child.data) {
      return child.children.map(function (c) {
        return 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(function (child) {
    return Array(findData(child)).join('');
  }).join('') : '';
}

function display(results, disableConsole, onlyUrls) {
  logIt('\n', disableConsole);
  results.forEach(function (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.');
    }
  });
}

var parseGoogleSearchResultUrl = function parseGoogleSearchResultUrl(url) {
  if (!url) {
    return undefined;
  }

  if (url.charAt(0) === '/') {
    return querystring.parse(url).url;
  }

  return url;
};

exports.parseGoogleSearchResultUrl = parseGoogleSearchResultUrl;

function getResults(_ref) {
  var data = _ref.data,
      noDisplay = _ref.noDisplay,
      disableConsole = _ref.disableConsole,
      onlyUrls = _ref.onlyUrls,
      titleSelector = _ref.titleSelector,
      linkSelector = _ref.linkSelector,
      snippetSelector = _ref.snippetSelector,
      resultStatsSelector = _ref.resultStatsSelector,
      cursorSelector = _ref.cursorSelector;
  var $ = cheerio.load(data);
  var results = [];
  var titles = $(getTitleSelector(titleSelector)).contents();
  titles.each(function (index, elem) {
    if (elem.data) {
      results.push({
        title: elem.data
      });
    } else {
      results.push({
        title: elem.children[0].data
      });
    }
  });
  $(getLinkSelector(linkSelector)).map(function (index, elem) {
    if (index < results.length) {
      results[index] = Object.assign(results[index], {
        link: parseGoogleSearchResultUrl(elem.attribs.href)
      });
    }
  });
  $(getSnippetSelector(snippetSelector)).map(function (index, elem) {
    if (index < results.length) {
      results[index] = Object.assign(results[index], {
        snippet: getSnippet(elem)
      });
    }
  });

  if (onlyUrls) {
    results = results.map(function (r) {
      return {
        link: r.link
      };
    });
  }

  if (!noDisplay) {
    display(results, disableConsole, onlyUrls);
  }

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

```
    seconds: seconds
  };
  return {
    results: results,
    stats: stats
  };
}

function getResponse(_ref2) {
  var filePath = _ref2.fromFile,
      fromString = _ref2.fromString,
      options = _ref2.options,
      htmlFileOutputPath = _ref2.htmlFileOutputPath,
      query = _ref2.query,
      limit = _ref2.limit,
      userAgent = _ref2.userAgent,
      start = _ref2.start,
      includeSites = _ref2.includeSites,
      excludeSites = _ref2.excludeSites;
  // eslint-disable-next-line consistent-return
  return new Promise(function (resolve, reject) {
    if (filePath) {
      fs.readFile(filePath, function (err, data) {
        if (err) {
          return reject(new Error("Erorr accessing file at ".concat(filePath, ": ").concat(err)));
        }

        return resolve({
          body: data
        });
      });
    } else if (fromString) {
      return resolve({
        body: fromString
      });
    }

    var defaultOptions = getDefaultRequestOptions({
      limit: limit,
      query: query,
      userAgent: userAgent,
      start: start,
      includeSites: includeSites,
      excludeSites: excludeSites
    });
    request(_objectSpread(_objectSpread({}, defaultOptions), options), function (error, response, body) {
      if (error) {
        return reject(new Error("Error making web request: ".concat(error)));
      }

      saveResponse(response, htmlFileOutputPath);
      return resolve({
        body: body,
        response: response
      });
    });
  });
}

function googleIt(config) {
  var output = config.output,
      open = config.open,
      returnHtmlBody = config.returnHtmlBody,
      titleSelector = config.titleSelector,
      linkSelector = config.linkSelector,
      snippetSelector = config.snippetSelector,
      resultStatsSelector = config.resultStatsSelector,
      cursorSelector = config.cursorSelector,
      start = config.start,
      diagnostics = config.diagnostics;
  return new Promise(function (resolve, reject) {
    getResponse(config).then(function (_ref3) {
      var body = _ref3.body,
          response = _ref3.response;

      var _getResults = getResults({
        data: body,
        noDisplay: config['no-display'],
        disableConsole: config.disableConsole,
        onlyUrls: config['only-urls'],
        titleSelector: titleSelector,
        linkSelector: linkSelector,
        snippetSelector: snippetSelector,
        resultStatsSelector: resultStatsSelector,
        cursorSelector: cursorSelector,
        start: start
      }),
          results = _getResults.results,
          stats = _getResults.stats;

      var statusCode = response.statusCode;

      if (results.length === 0 && statusCode !== 200 && !diagnostics) {
        reject(new Error("Error in response: statusCode ".concat(statusCode, ". To see the raw response object, please include the 'diagnostics: true' as part of the options object
      }

      saveToFile(output, results);
      openInBrowser(open, results);
```

```
271
272       if (returnHtmlBody || diagnostics) {
273         return resolve({
274           results: results,
275           body: body,
276           response: response,
277           stats: stats
278         });
279       }
280
281       return resolve(results);
282     }).catch(reject);
283   });
284 }
285
286 var _default = googleIt;
287 exports.default = _default;
```