sebhildebrandt inetLatency() fix for alpine (linux) ✓ History

2 contributors

236 lines (220 sloc) | 8.43 KB

```javascript
'use strict';
// @ts-check
// ==================================================================================
// internet.js
// ----------------------------------------------------------------------------------
// Description:   System Information - library
//                for Node.js
// Copyright:     (c) 2014 - 2022
// Author:        Sebastian Hildebrandt
// ----------------------------------------------------------------------------------
// License:       MIT
// ==================================================================================
// 12. Internet
// ----------------------------------------------------------------------------------

// const exec = require('child_process').exec;
const util = require('./util');

let _platform = process.platform;

const _linux = (_platform === 'linux' || _platform === 'android');
const _darwin = (_platform === 'darwin');
const _windows = (_platform === 'win32');
const _freebsd = (_platform === 'freebsd');
const _openbsd = (_platform === 'openbsd');
const _netbsd = (_platform === 'netbsd');
const _sunos = (_platform === 'sunos');

// --------------------------
// check if external site is available

function inetChecksite(url, callback) {

  return new Promise((resolve) => {
    process.nextTick(() => {
      let result = {
        url: url,
        ok: false,
        status: 404,
        ms: null
      };
      if (typeof url !== 'string') {
        if (callback) { callback(result); }
        return resolve(result);
      }
      let urlSanitized = '';
      const s = util.sanitizeShellString(url, true);
      for (let i = 0; i <= util.mathMin(s.length, 2000); i++) {
        if (s[i] !== undefined) {
          s[i].__proto__.toLowerCase = util.stringToLower;
          const sl = s[i].toLowerCase();
          if (sl && sl[0] && !sl[1] && sl[0].length === 1) {
            urlSanitized = urlSanitized + sl[0];
          }
        }
      }
      result.url = urlSanitized;
      try {
        if (urlSanitized && !util.isPrototypePolluted()) {
          urlSanitized.__proto__.startsWith = util.stringStartWith;
          if (urlSanitized.startsWith('file:') || urlSanitized.startsWith('gopher:') || urlSanitized.startsWith('telnet:') || urlSanitized.startsWith('mailto:') || urlSanitized.st
            if (callback) { callback(result); }
            return resolve(result);
          }
          let t = Date.now();
          if (_linux || _freebsd || _openbsd || _netbsd || _darwin || _sunos) {
            let args = ['-I', '--connect-timeout', '5', '-m', '5'];
            args.push(urlSanitized);
            let cmd = 'curl';
            util.execSafe(cmd, args).then((stdout) => {
              const lines = stdout.split('\n');
              let statusCode = lines[0] && lines[0].indexOf(' ') >= 0 ? parseInt(lines[0].split(' ')[1], 10) : 404;
              result.status = statusCode || 404;
              result.ok = (statusCode === 200 || statusCode === 301 || statusCode === 302 || statusCode === 304);
              result.ms = (result.ok ? Date.now() - t : null);
              if (callback) { callback(result); }
              resolve(result);
            });
          }
          if (_windows) {   // if this is stable, this can be used for all OS types
            const http = (urlSanitized.startsWith('https:') ? require('https') : require('http'));
            try {
              http.get(urlSanitized, (res) => {
                const statusCode = res.statusCode;

                result.status = statusCode || 404;
                result.ok = (statusCode === 200 || statusCode === 301 || statusCode === 302 || statusCode === 304);

                if (statusCode !== 200) {
                  res.resume();
                  result.ms = (result.ok ? Date.now() - t : null);
                  if (callback) { callback(result); }
                  resolve(result);
                } else {
                  res.on('data', () => { });
                  res.on('end', () => {
                    result.ms = (result.ok ? Date.now() - t : null);
                    if (callback) { callback(result); }
                    resolve(result);
                  });
                }
              }).on('error', () => {
                if (callback) { callback(result); }
                resolve(result);
              });
            } catch (err) {
              if (callback) { callback(result); }
              resolve(result);
            }
          }
        } else {
          if (callback) { callback(result); }
          resolve(result);
        }
      } catch (err) {
        if (callback) { callback(result); }
        resolve(result);
      }
    });
  });
}

exports.inetChecksite = inetChecksite;

// --------------------------
// check inet latency

function inetLatency(host, callback) {

  // fallback - if only callback is given
  if (util.isFunction(host) && !callback) {
    callback = host;
    host = '';
  }

  host = host || '8.8.8.8';

  return new Promise((resolve) => {
    process.nextTick(() => {
      if (typeof host !== 'string') {
        if (callback) { callback(null); }
        return resolve(null);
      }
      let hostSanitized = '';
      const s = (util.isPrototypePolluted() ? '8.8.8.8' : util.sanitizeShellString(host, true)).trim();
      for (let i = 0; i <= util.mathMin(s.length, 2000); i++) {
        if (!(s[i] === undefined)) {
          s[i].__proto__.toLowerCase = util.stringToLower;
          const sl = s[i].toLowerCase();
          if (sl && sl[0] && !sl[1]) {
            hostSanitized = hostSanitized + sl[0];
          }
        }
      }
      hostSanitized.__proto__.startsWith = util.stringStartWith;
      if (hostSanitized.startsWith('file:') || hostSanitized.startsWith('gopher:') || hostSanitized.startsWith('telnet:') || hostSanitized.startsWith('mailto:') || hostSanitized.s
        if (callback) { callback(null); }
        return resolve(null);
      }
      let params;
      if (_linux || _freebsd || _openbsd || _netbsd || _darwin) {
        if (_linux) {
          params = ['-c', '2', '-w', '3', hostSanitized];
        }
        if (_freebsd || _openbsd || _netbsd) {
          params = ['-c', '2', '-t', '3', hostSanitized];
        }
        if (_darwin) {
          params = ['-c2', '-t3', hostSanitized];
        }
        util.execSafe('ping', params).then((stdout) => {
          let result = null;
          if (stdout) {
            const lines = stdout.split('\n').filter((line) => (line.indexOf('rtt') >= 0 || line.indexOf('round-trip') >= 0 || line.indexOf('avg') >= 0)).join('\n');

            const line = lines.split('=');
            if (line.length > 1) {
              const parts = line[1].split('/');
              if (parts.length > 1) {
                result = parseFloat(parts[1]);
              }
            }
          }
          if (callback) { callback(result); }
          resolve(result);
        });
      }
      if (_sunos) {
        const params = ['-s', '-a', hostSanitized, '56', '2'];
        const filt = 'avg';
        util.execSafe('ping', params, { timeout: 3000 }).then((stdout) => {
          let result = null;
          if (stdout) {
            const lines = stdout.split('\n').filter(line => line.indexOf(filt) >= 0).join('\n');
            const line = lines.split('=');
            if (line.length > 1) {
              const parts = line[1].split('/');
              if (parts.length > 1) {
                result = parseFloat(parts[1].replace(',', '.'));
              }
            }
          }
          if (callback) { callback(result); }
          resolve(result);
        });
      }
      if (_windows) {
        let result = null;
        try {
          const params = [hostSanitized, '-n', '1'];
          util.execSafe('ping', params, util.execOptsWin).then((stdout) => {
            if (stdout) {
              let lines = stdout.split('\r\n');
              lines.shift();
              lines.forEach(function (line) {
                if ((line.toLowerCase().match(/ms/g) || []).length === 3) {
                  let l = line.replace(/ +/g, ' ').split(' ');
                  if (l.length > 6) {
                    result = parseFloat(l[l.length - 1]);
                  }
                }
              });
            }
            if (callback) { callback(result); }
            resolve(result);
          });
        } catch (e) {
          if (callback) { callback(result); }
          resolve(result);
        }
      }
    });
  });
}

exports.inetLatency = inetLatency;
```