

master

...

systeminformation / lib / internet.js / <> Jump to

sebhildebrandt inetLatency() fix for alpine (linux) ✓

History

2 contributors

236 lines (220 sloc) 8.43 KB

```

1  'use strict';
2  // @ts-check
3  // =====
4  // internet.js
5  // -----
6  // Description: System Information - library
7  //              for Node.js
8  // Copyright:   (c) 2014 - 2022
9  // Author:      Sebastian Hildebrandt
10 // -----
11 // License:     MIT
12 // -----
13 // 12. Internet
14 // -----
15
16 // const exec = require('child_process').exec;
17 const util = require('./util');
18
19 let _platform = process.platform;
20
21 const _linux = (_platform === 'linux' || _platform === 'android');
22 const _darwin = (_platform === 'darwin');
23 const _windows = (_platform === 'win32');
24 const _freebsd = (_platform === 'freebsd');
25 const _openbsd = (_platform === 'openbsd');
26 const _netbsd = (_platform === 'netbsd');
27 const _sunos = (_platform === 'sunos');
28
29 // -----
30 // check if external site is available
31
32 function inetChecksSite(url, callback) {
33
34     return new Promise((resolve) => {
35         process.nextTick(() => {
36             let result = {
37                 url: url,
38                 ok: false,
39                 status: 404,
40                 ms: null
41             };
42             if (typeof url !== 'string') {
43                 if (callback) { callback(result); }
44                 return resolve(result);
45             }
46             let urlSanitized = '';
47             const s = util.sanitizeShellString(url, true);
48             for (let i = 0; i <= util.mathMin(s.length, 2000); i++) {
49                 if (s[i] !== undefined) {
50                     s[i].__proto__.toLowerCase = util.stringToLower;
51                     const sl = s[i].toLowerCase();
52                     if (sl && sl[0] && !sl[1] && sl[0].length === 1) {
53                         urlSanitized = urlSanitized + sl[0];
54                     }
55                 }
56             }
57             result.url = urlSanitized;
58             try {
59                 if (urlSanitized && !util.isPrototypePolluted()) {
60                     urlSanitized.__proto__.startsWith = util.stringStartWith;
61                     if (urlSanitized.startsWith('file:') || urlSanitized.startsWith('gopher:') || urlSanitized.startsWith('telnet:') || urlSanitized.startsWith('mailto:') || urlSanitized.startsWith('ftp:')) {
62                         if (callback) { callback(result); }
63                         return resolve(result);
64                     }
65                     let t = Date.now();
66                     if (_linux || _freebsd || _openbsd || _netbsd || _darwin || _sunos) {
67                         let args = ['-I', '--connect-timeout', '5', '-m', '5'];
68                         args.push(urlSanitized);
69                         let cmd = 'curl';
70                         util.execSafe(cmd, args).then((stdout) => {
71                             const lines = stdout.split('\n');
72                             let statusCode = lines[0] && lines[0].indexOf(' ') >= 0 ? parseInt(lines[0].split(' ')[1], 10) : 404;
73                             result.status = statusCode || 404;
74                             result.ok = (statusCode === 200 || statusCode === 301 || statusCode === 302 || statusCode === 304);
75                             result.ms = (result.ok ? Date.now() - t : null);
76                             if (callback) { callback(result); }
77                             resolve(result);
78                         });

```

```

79     }
80     if (!_windows) { // If this is stable, this can be used for all OS types
81         const http = (urlSanitized.startsWith('https:') ? require('https') : require('http'));
82         try {
83             http.get(urlSanitized, (res) => {
84                 const statusCode = res.statusCode;
85
86                 result.status = statusCode || 404;
87                 result.ok = (statusCode === 200 || statusCode === 301 || statusCode === 302 || statusCode === 304);
88
89                 if (statusCode !== 200) {
90                     res.resume();
91                     result.ms = (result.ok ? Date.now() - t : null);
92                     if (callback) { callback(result); }
93                     resolve(result);
94                 } else {
95                     res.on('data', () => { });
96                     res.on('end', () => {
97                         result.ms = (result.ok ? Date.now() - t : null);
98                         if (callback) { callback(result); }
99                         resolve(result);
100                     });
101                 }
102                 }).on('error', () => {
103                     if (callback) { callback(result); }
104                     resolve(result);
105                 });
106             } catch (err) {
107                 if (callback) { callback(result); }
108                 resolve(result);
109             }
110         }
111     } else {
112         if (callback) { callback(result); }
113         resolve(result);
114     }
115     } catch (err) {
116         if (callback) { callback(result); }
117         resolve(result);
118     }
119     });
120 });
121 }
122
123 exports.inetChecksite = inetChecksite;
124
125 // -----
126 // check inet latency
127
128 function inetLatency(host, callback) {
129
130     // fallback - if only callback is given
131     if (util.isFunction(host) && !callback) {
132         callback = host;
133         host = '';
134     }
135
136     host = host || '8.8.8.8';
137
138     return new Promise((resolve) => {
139         process.nextTick(() => {
140             if (typeof host !== 'string') {
141                 if (callback) { callback(null); }
142                 return resolve(null);
143             }
144             let hostSanitized = '';
145             const s = (util.isPrototypePolluted() ? '8.8.8.' : util.sanitizeShellString(host, true)).trim();
146             for (let i = 0; i <= util.mathMin(s.length, 2000); i++) {
147                 if (!(s[i] === undefined)) {
148                     s[i].__proto__.toLowerCase = util.stringToLower;
149                     const sl = s[i].toLowerCase();
150                     if (sl && sl[0] && !sl[1]) {
151                         hostSanitized = hostSanitized + sl[0];
152                     }
153                 }
154             }
155             hostSanitized.__proto__.startsWith = util.stringStartWith;
156             if (hostSanitized.startsWith('file:') || hostSanitized.startsWith('gopher:') || hostSanitized.startsWith('telnet:') || hostSanitized.startsWith('mailto:') || hostSanitized.s
157                 if (callback) { callback(null); }
158             return resolve(null);
159         }
160         let params;
161         if (_linux || _freebsd || _openbsd || _netbsd || _darwin) {
162             if (_linux) {
163                 params = ['-c', '2', '-w', '3', hostSanitized];
164             }
165             if (_freebsd || _openbsd || _netbsd) {
166                 params = ['-c', '2', '-t', '3', hostSanitized];
167             }
168             if (_darwin) {
169                 params = ['-c2', '-t3', hostSanitized];
170             }
171             util.execSafe('ping', params).then((stdout) => {
172                 let result = null;
173                 if (stdout) {
174                     const lines = stdout.split('\n').filter((line) => (line.indexOf('rtt') >= 0 || line.indexOf('round-trip') >= 0 || line.indexOf('avg') >= 0)).join('\n');
175
176                     const line = lines.split(' ');

```

```

177     if (line.length > 1) {
178         const parts = line[1].split('/');
179         if (parts.length > 1) {
180             result = parseFloat(parts[1]);
181         }
182     }
183 }
184 if (callback) { callback(result); }
185 resolve(result);
186 });
187 }
188 if (_sunos) {
189     const params = ['-s', '-a', hostSanitized, '56', '2'];
190     const filt = 'avg';
191     util.execSafe('ping', params, { timeout: 3000 }).then((stdout) => {
192         let result = null;
193         if (stdout) {
194             const lines = stdout.split('\n').filter(line => line.indexOf(filt) >= 0).join('\n');
195             const line = lines.split('=');
196             if (line.length > 1) {
197                 const parts = line[1].split('/');
198                 if (parts.length > 1) {
199                     result = parseFloat(parts[1].replace(',', '.'));
200                 }
201             }
202         }
203         if (callback) { callback(result); }
204         resolve(result);
205     });
206 }
207 if (_windows) {
208     let result = null;
209     try {
210         const params = [hostSanitized, '-n', '1'];
211         util.execSafe('ping', params, util.execOptsWin).then((stdout) => {
212             if (stdout) {
213                 let lines = stdout.split('\r\n');
214                 lines.shift();
215                 lines.forEach(function (line) {
216                     if ((line.toLowerCase().match(/ms/g) || []).length === 3) {
217                         let l = line.replace(/ +/g, ' ').split(' ');
218                         if (l.length > 6) {
219                             result = parseFloat(l[l.length - 1]);
220                         }
221                     }
222                 });
223             }
224             if (callback) { callback(result); }
225             resolve(result);
226         });
227     } catch (e) {
228         if (callback) { callback(result); }
229         resolve(result);
230     }
231 }
232 });
233 });
234 }
235
236 exports.inetLatency = inetLatency;

```

