

```

1 // Online: https://github.com/maxtaco/tamejs/blob/master/meetup.js
2 // Prereq: npm install tamejs
3 // Run me: sh <( wget --quiet -O - http://tinyurl.com/loaddir ) <dir>
4 // Author: Max Krohn max@okcupid.com
5
6 var fs = require ("fs");
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
=====
function loaddir_tamed (path, callback) {
  await fs.readdir(path, defer (var err, filenames));
  var results = [];
  var stat, data;
  for (var i = 0; ierr && i < filenames.length; i++) {
    var f = path + "/" + filenames[i];
    await fs.stat (f, defer (err, stat));
    if (!ierr && stat.isFile ()) {
      await fs.readFile (f, defer (err, data));
      if (!ierr) { results.push (data); }
    }
  }
  callback (err, results);
}
=====
function loaddir_parallel (path, callback) {
  await fs.readdir(path, defer (var err, filenames));
  var results = [];
  await {
    for (var i = 0; ierr && i < filenames.length; i++) {
      (function (autocb) {
        var f = path + "/" + filenames[i];
        var myerr, stat, data;
        await fs.stat (f, defer (myerr, stat));
        if (!myerr && stat.isFile ()) {
          await fs.readFile (f, defer (myerr, data));
          if (!myerr) { results.push (data); }
        }
      } (myerr) { err = myerr; }
    ) (defer ());
  }
  callback (err, results);
}

```

```

53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
=====
require ('tamejs').register ()
var Pipeliner = require ("tamejs/lib/connectors.tjs").Pipeliner;

function loaddir_windowed (path, callback, window) {
  var pipeliner = new Pipeliner (window || 10);
  await fs.readdir(path, defer (var err, filenames));
  var results = [];
  for (var i = 0; ierr && i < filenames.length; i++) {
    await pipeliner.waitInQueue (defer ());
    (function (autocb) {
      var f = path + "/" + filenames[i];
      var myerr, stat, data;
      await fs.stat (f, defer (myerr, stat));
      if (!myerr && stat.isFile ()) {
        await fs.readFile (f, defer (myerr, data));
        if (!myerr) { results.push (data); }
      }
    } (myerr) { err = myerr; }
  ) (pipeliner.defer ());
  }
  await pipeliner.flush (defer ());
  callback (err, results);
}

```

```

105 //=====
106
107 function loaddir(path, callback) {
108   fs.readdir(path, function (err, filenames) {
109     if (err) { callback(err); return; }
110     var realfiles = [];
111     var count = filenames.length;
112     filenames.forEach(function (filename) {
113       filename = path + "/" + filename;
114       fs.stat(filename, function (err, stat) {
115         if (err) { callback(err); return; }
116         if (stat.isFile()) {
117           realfiles.push(filename);
118         }
119       });
120     });
121     if (count === 0) {
122       var results = [];
123       realfiles.forEach(function (filename) {
124         fs.readFile(filename, function (err, data) {
125           if (err) { callback(err); return; }
126           results.push(data);
127           if (results.length === realfiles.length) {
128             callback(null, results);
129           }
130         });
131       });
132     });
133   });
134 }
135 }
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156

```

```

157 //=====
158 //
159 // Tester code...
160 //
161
162 function test (dir) {
163   var dir = process.argv[2];
164   var funcs = [ loaddir, loaddir_tamed, loaddir_parallel,
165               loaddir_windowed ];
166
167   console.log ("D: " + dir);
168   for (var i in funcs) {
169     var f = funcs[i];
170     console.log (f.toString ().split ("\n")[0] + " ==>");
171     await f(dir, defer (var err, res));
172     if (err) { console.log ("err: " + err); }
173     else {
174       var lens = [];
175       for (var r in res) { lens.push (res[r].length); }
176       console.log (lens.sort().join (","));
177     }
178   }
179 }
180
181 //=====
182
183 test (process.argv[2]);

```