thlorenz removing find-parent-dir legacy code and dep History

1 contributor

213 lines (175 sloc) | 7.5 KB ...

```
1  'use strict';
2
3  var path             = require('path')
4    , fs               = require('fs')
5    , util             = require('util')
6    , parseInlineShims = require('./parse-inline-shims')
7    , mothership       = require('mothership')
8    , format           = require('util').format
9
10 var shimsCache =  {}
11   , shimsByPath     =  {};
12
13 function inspect(obj, depth) {
14   return util.inspect(obj, false, depth || 5, true);
15 }
16
17 function isPath(s) {
18   return (/^[.]{0,2}[/\\]/).test(s);
19 }
20
21 function validate(key, config, dir) {
22   var msg
23     , details = 'When evaluating shim "' + key + '": ' + inspect(config) + '\ninside ' + dir + '\n
24
25   if (!config.hasOwnProperty('exports')) {
26     msg = 'browserify-shim needs at least a path and exports to do its job, you are missing the ex
27           '\nIf this module has no exports, specify exports as null.'
28     throw new Error(details + msg);
29   }
```

```
}

function updateCache(packageDir, pack, resolvedShims, exposeGlobals) {
  shimsCache[packageDir] = { pack: pack, shims: resolvedShims, exposeGlobals: exposeGlobals };
  Object.keys(resolvedShims).forEach(function(fullPath) {
    var shim = resolvedShims[fullPath];
    validate(fullPath, shim, packageDir);
    shimsByPath[fullPath] = shim;
  });
}

function resolveDependsRelativeTo(dir, browser, depends, packDeps, messages) {
  var resolved;

  if (!depends) return undefined;

  return Object.keys(depends).reduce(function (acc, k) {
    if (browser[k]){
      acc[k] = depends[k];
      messages.push(format('Found depends "%s" exposed in browser field', k));
    } else if (!isPath(k)) {
      acc[k] = depends[k];
      if (packDeps[k]) {
        messages.push(format('Found depends "%s" as an installed dependency of the package', k));
      } else {
        messages.push(format('WARNING, depends "%s" is not a path, nor is it exposed in the browse
      }
    } else {
      // otherwise resolve the path
      resolved = path.resolve(dir, k);
      acc[resolved] = depends[k];
      messages.push(format('Depends "%s" was resolved to be at [%s]', k, resolved));
    }

    return acc;
  }, {})
}

function resolvePaths (packageDir, shimFileDir, browser, shims, packDeps, messages) {
  return Object.keys(shims)
    .reduce(function (acc, relPath) {
      var shim = shims[relPath];
      var exposed = browser[relPath];
      var shimPath;

      if (exposed) {
        // lib exposed under different name/path in package.json's browser field
        // and it is referred to by this alias in the shims (either external or in package.json)
        // i.e.: 'non-cjs': { ... } -> browser: { 'non-cjs': './vendor/non-cjs.js }
```

```
        shimPath = path.resolve(packageDir, exposed);
        messages.push(format('Found "%s" in browser field referencing "%s" and resolved it to "%s"
      } else if (shimFileDir) {
        // specified via relative path to shim file inside shim file
        // i.e. './vendor/non-cjs': { exports: .. }
        shimPath = path.resolve(shimFileDir, relPath);
        messages.push(format('Resolved "%s" found in shim file to "%s"', relPath, shimPath));
      } else {
        // specified via relative path in package.json browserify-shim config
        // i.e. 'browserify-shim': { './vendor/non-cjs': 'noncjs' }
        shimPath = path.resolve(packageDir, relPath);
        messages.push(format('Resolved "%s" found in package.json to "%s"', relPath, shimPath));
      }
      var depends = resolveDependsRelativeTo(shimFileDir || packageDir, browser, shim.depends, pac

      acc[shimPath] = { exports: shim.exports, depends: depends };
      return acc;
    }, {});
}

function mapifyExposeGlobals(exposeGlobals) {
  return Object.keys(exposeGlobals)
    .reduce(function (acc, k) {

      var val = exposeGlobals[k];
      var parts = val.split(':');

      if (parts.length < 2 || !parts[1].length) {
        throw new Error(
            'Expose Globals need to have the format "global:expression.\n"'
          + inspect({ key: k, value: val }) + 'does not.'
        );
      }

      // this also handle unlikely cases of 'global:_.someFunc(':')' with a `:` in the actual glob
      parts.shift();
      acc[k] = parts.join(':');

      return acc;
    }, {});
}

function separateExposeGlobals(shims) {
  var onlyShims = {}
    , exposeGlobals = {};

  Object.keys(shims).forEach(function (k) {
    var val = shims[k]
      , exp = val && val.exports;
```

```
128
129     if (exp && /^global\:/.test(exp)) {
130       exposeGlobals[k] = exp;
131     } else {
132       onlyShims[k] = val;
133     }
134   });
135
136   return { shims: onlyShims, exposeGlobals: mapifyExposeGlobals(exposeGlobals) };
137 }
138
139 function resolveFromShimFile(packageDir, pack, shimField, messages) {
140   var shimFile =  path.join(packageDir, shimField)
141     , shimFileDir = path.dirname(shimFile);
142
143   var allShims = require(shimFile);
144   var separated = separateExposeGlobals(allShims);
145
146   var resolvedShims = resolvePaths(packageDir, shimFileDir, pack.browser || {}, separated.shims, p
147   return { shims: resolvedShims, exposeGlobals: separated.exposeGlobals };
148 }
149
150 function resolveInlineShims(packageDir, pack, shimField, messages) {
151   var allShims = parseInlineShims(shimField);
152   var separated = separateExposeGlobals(allShims);
153
154   var resolvedShims = resolvePaths(packageDir, null, pack.browser || {}, separated.shims, pack.dep
155   return { shims: resolvedShims, exposeGlobals: separated.exposeGlobals };
156 }
157
158 var resolve = module.exports = function resolveShims (file, messages, cb) {
159   // find the package.json that defines browserify-shim config for this file
160   mothership(file, function (pack) { return !! pack['browserify-shim'] }, function (err, res) {
161     if (err) return cb(err);
162
163     if (!res || !res.pack) return cb(new Error('Unable to find a browserify-shim config section in
164
165     var pack       = res.pack;
166     var packFile   = res.path;
167     var packageDir = path.dirname(packFile);
168
169     // we cached this before which means it was also grouped by file
170     var cached = shimsCache[packageDir];
171     // if it was cached, that means any package fixes were applied as well
172     if (cached) {
173       return cb(null, {
174           package_json       :  packFile
175         , packageDir         :  packageDir
176         , resolvedPreviously :  true
```

```
177         , shim               :  shimsByPath[file]
178         , exposeGlobals      :  cached.exposeGlobals
179         , browser            :  pack.browser
180         , 'browserify-shim'  :  pack['browserify-shim']
181         , dependencies       :  pack.dependencies
182       });
183     }
184
185       try {
186         pack = require(packFile);
187
188         var shimField = pack['browserify-shim'];
189         if (!shimField) return cb(null, { package_json: packFile, shim: undefined });
190
191         var resolved = typeof shimField === 'string'
192           ? resolveFromShimFile(packageDir, pack, shimField, messages)
193           : resolveInlineShims(packageDir, pack, shimField, messages);
194
195         messages.push({ resolved: resolved.shims });
196         updateCache(packageDir, pack, resolved.shims, resolved.exposeGlobals);
197
198         cb(null, {
199             package_json      :  packFile
200           , packageDir        :  packageDir
201           , shim              :  shimsByPath[file]
202           , exposeGlobals     :  resolved.exposeGlobals
203           , browser           :  pack.browser
204           , 'browserify-shim' :  pack['browserify-shim']
205           , dependencies      :  pack.dependencies
206         });
207
208       } catch (err) {
209         console.trace();
210         return cb(err);
211       }
212     });
213 }
```