

node-errno

*Better [libuv/Node.js/io.js](#) error handling & reporting.
Available in npm as *errno*.*

[Build Status](#)

[errno exposed](#)

[Custom errors](#)

errno exposed

Ever find yourself needing more details about Node.js errors?
Me too, so *node-errno* contains the *errno* mappings direct from
libuv so you can use them in your code.

By *errno*:

By code:

```
// "description": "directory not
// empty"
}
```

Make your errors more descriptive:

```
var errno = require('errno')

function errmsg(err) {
  var str = 'Error: '
  // if it's a libuv error then get the
  description from errno
  if (errno.errno[err.errno])
    str +=
      errno.errno[err.errno].description
  else
    str += err.message

  // if it's a `fs` error then it'll
  have a 'path' property
  if (err.path)
    str += ' [' + err.path + ']'

  return str
}

var fs = require('fs')

fs.readFile('thisisnotarealfile.txt',
  function (err, data) {
    if (err)
      console.log(errmsg(err))
  })
```

TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Use as a command line tool:

```
~ $ errno 53
{
  "errno": 53,
  "code": "ENOTEMPTY",
  "description": "directory not empty"
}
~ $ errno EROFS
{
  "errno": 56,
  "code": "EROFS",
  "description": "read-only file system"
}
~ $ errno foo
No such errno/code: "foo"
```

Supply no arguments for the full list. Error codes are processed case-insensitive.

You will need to install with `npm install errno -g` if you want the `errno` command to be available without supplying a full path to the `node_modules` installation.

Custom errors

Use `errno.custom.createError()` to create custom Error objects to throw around in your Node.js library. Create error hierarchies so `instanceof` becomes a useful tool in tracking errors. Call-stack is correctly captured at the time you create an instance of the error object, plus a `cause` property will make available the original error object if you pass one in to the constructor.

```

var create =
  require('errno').custom.createError
  var MyError = create('MyError') //
    inherits from Error
  var SpecificError =
    create('SpecificError',
      MyError) // inherits from
    MyError
  var OtherError = create('OtherError',
    MyError)

// use them!
if (condition) throw new
  SpecificError('Eek! Something
    bad happened')
if (err) return callback(new
  OtherError(err))

```

Also available is a `errno.custom.FileSystemError` with in-built access to `errno` properties:

The resulting error object passed through the callback will have the following properties: `code`, `errno`, `path` and `message` will contain a descriptive human-readable message.

- ## Contributors
- [bahamas10](#) (Dave Eddy) - Added CLI
 - [ralphtheninja](#) (Lars-Magnus Skog)

Copyright & Licence

Copyright (c) 2012-2015 [Rod Vagg \(@rvagg\)](#)
 Made available under the MIT licence:

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,