# http-errors

[NPM Version](https://npmjs.org/package/http-errors) [NPM Downloads](https://nodejs.org/en/download) [Node.js Version](https://nodejs.org/en/download) [Build Status](https://github.com/jshttp/http-errors/actions?query=workflow%3Aci) [Test Coverage](https://coveralls.io/r/jshttp/http-errors?branch=master)

Create HTTP errors for Express, Koa, Connect, etc. with ease.

## Install

This is a [Node.js](https://nodejs.org/en/) module available through the [npm registry](https://www.npmjs.com/). Installation is done using the [npm install command](https://docs.npmjs.com/getting-started/installing-npm-packages-locally):

$ npm install http-errors

## Example

var createError = require('http-errors')

var express = require('express')

var app = express()

app.use(function (req, res, next) {

if (!req.user) return next(createError(401, 'Please login to view this page.'))

next()

})

## API

This is the current API, currently extracted from Koa and subject to change.

### Error Properties

* expose - can be used to signal if message should be sent to the client, defaulting to false when status >= 500
* headers - can be an object of header names to values to be sent to the client, defaulting to undefined. When defined, the key names should all be lower-cased
* message - the traditional error message, which should be kept short and all single line
* status - the status code of the error, mirroring statusCode for general compatibility
* statusCode - the status code of the error, defaulting to 500

### createError([status], [message], [properties])

Create a new error object with the given message msg. The error object inherits from createError.HttpError.

var err = createError(404, 'This video does not exist!')

* status: 500 - the status code as a number
* message - the message of the error, defaulting to node's text for that status code.
* properties - custom properties to attach to the object

### createError([status], [error], [properties])

Extend the given error object with createError.HttpError properties. This will not alter the inheritance of the given error object, and the modified error object is the return value.

fs.readFile('foo.txt', function (err, buf) {

if (err) {

if (err.code === 'ENOENT') {

var httpError = createError(404, err, { expose: false })

} else {

var httpError = createError(500, err)

}

}

})

* status - the status code as a number
* error - the error object to extend
* properties - custom properties to attach to the object

### createError.isHttpError(val)

Determine if the provided val is an HttpError. This will return true if the error inherits from the HttpError constructor of this module or matches the "duck type" for an error this module creates. All outputs from the createError factory will return true for this function, including if an non-HttpError was passed into the factory.

### new createError[code || name]([msg]))

Create a new error object with the given message msg. The error object inherits from createError.HttpError.

var err = new createError.NotFound()

* code - the status code as a number
* name - the name of the error as a "bumpy case", i.e. NotFound or InternalServerError.

#### List of all constructors

| **Status Code** | **Constructor Name** |
| --- | --- |
| 400 | BadRequest |
| 401 | Unauthorized |
| 402 | PaymentRequired |
| 403 | Forbidden |
| 404 | NotFound |
| 405 | MethodNotAllowed |
| 406 | NotAcceptable |
| 407 | ProxyAuthenticationRequired |
| 408 | RequestTimeout |
| 409 | Conflict |
| 410 | Gone |
| 411 | LengthRequired |
| 412 | PreconditionFailed |
| 413 | PayloadTooLarge |
| 414 | URITooLong |
| 415 | UnsupportedMediaType |
| 416 | RangeNotSatisfiable |
| 417 | ExpectationFailed |
| 418 | ImATeapot |
| 421 | MisdirectedRequest |
| 422 | UnprocessableEntity |
| 423 | Locked |
| 424 | FailedDependency |
| 425 | TooEarly |
| 426 | UpgradeRequired |
| 428 | PreconditionRequired |
| 429 | TooManyRequests |
| 431 | RequestHeaderFieldsTooLarge |
| 451 | UnavailableForLegalReasons |
| 500 | InternalServerError |
| 501 | NotImplemented |
| 502 | BadGateway |
| 503 | ServiceUnavailable |
| 504 | GatewayTimeout |
| 505 | HTTPVersionNotSupported |
| 506 | VariantAlsoNegotiates |
| 507 | InsufficientStorage |
| 508 | LoopDetected |
| 509 | BandwidthLimitExceeded |
| 510 | NotExtended |
| 511 | NetworkAuthenticationRequired |

## License

[MIT](http://license)