**morgan**

HTTP request logger middleware for node.js

Named after [Dexter](http://en.wikipedia.org/wiki/Dexter_Morgan), a show you should not watch until completion.

**API**

var morgan = require('morgan')

**morgan(format, options)**

Create a new morgan logger middleware function using the given format and options. The format argument may be a string of a predefined name (see below for the names), a string of a format string, or a function that will produce a log entry.

The format function will be called with three arguments tokens, req, and res, where tokens is object with all defined tokens, req is the HTTP request and res is the HTTP response. The function is expected to return a string that will be the log line, or undefined / null to skip logging.

**Using a predefined format string**

morgan('tiny')

**Using format string of predefined tokens**

morgan(':method :url :status :res[content-length] - :response-time ms')

**Using a custom format function**

morgan(function (tokens, req, res) {

return [

tokens.method(req, res),

tokens.url(req, res),

tokens.status(req, res),

tokens.res(req, res, 'content-length'), '-',

tokens['response-time'](req, res), 'ms'

].join(' ')

})

**Options**

Morgan accepts these properties in the options object.

**immediate**

Write log line on request instead of response. This means that a requests will be logged even if the server crashes, *but data from the response (like the response code, content length, etc.) cannot be logged*.

**skip**

Function to determine if logging is skipped, defaults to false. This function will be called as skip(req, res).

// EXAMPLE: only log error responses

morgan('combined', {

skip: function (req, res) { return res.statusCode < 400 }

})

**stream**

Output stream for writing log lines, defaults to process.stdout.

**Predefined Formats**

There are various pre-defined formats provided:

**combined**

Standard Apache combined log output.

:remote-addr - :remote-user [:date[clf]] ":method :url HTTP/:http-version" :status :res[content-length] ":referrer" ":user-agent"

**common**

Standard Apache common log output.

:remote-addr - :remote-user [:date[clf]] ":method :url HTTP/:http-version" :status :res[content-length]

**dev**

Concise output colored by response status for development use. The :status token will be colored red for server error codes, yellow for client error codes, cyan for redirection codes, and uncolored for all other codes.

:method :url :status :response-time ms - :res[content-length]

**short**

Shorter than default, also including response time.

:remote-addr :remote-user :method :url HTTP/:http-version :status :res[content-length] - :response-time ms

**tiny**

The minimal output.

:method :url :status :res[content-length] - :response-time ms

**Tokens**

**Creating new tokens**

To define a token, simply invoke morgan.token() with the name and a callback function. This callback function is expected to return a string value. The value returned is then available as ":type" in this case:

morgan.token('type', function (req, res) { return req.headers['content-type'] })

Calling morgan.token() using the same name as an existing token will overwrite that token definition.

The token function is expected to be called with the arguments req and res, representing the HTTP request and HTTP response. Additionally, the token can accept further arguments of it's choosing to customize behavior.

**:date[format]**

The current date and time in UTC. The available formats are:

* clf for the common log format ("10/Oct/2000:13:55:36 +0000")
* iso for the common ISO 8601 date time format (2000-10-10T13:55:36.000Z)
* web for the common RFC 1123 date time format (Tue, 10 Oct 2000 13:55:36 GMT)

If no format is given, then the default is web.

**:http-version**

The HTTP version of the request.

**:method**

The HTTP method of the request.

**:referrer**

The Referrer header of the request. This will use the standard mis-spelled Referer header if exists, otherwise Referrer.

**:remote-addr**

The remote address of the request. This will use req.ip, otherwise the standard req.connection.remoteAddress value (socket address).

**:remote-user**

The user authenticated as part of Basic auth for the request.

**:req[header]**

The given header of the request.

**:res[header]**

The given header of the response.

**:response-time[digits]**

The time between the request coming into morgan and when the response headers are written, in milliseconds.

The digits argument is a number that specifies the number of digits to include on the number, defaulting to 3, which provides microsecond precision.

**:status**

The status code of the response.

If the request/response cycle completes before a response was sent to the client (for example, the TCP socket closed prematurely by a client aborting the request), then the status will be empty (displayed as "-" in the log).

**:url**

The URL of the request. This will use req.originalUrl if exists, otherwise req.url.

**:user-agent**

The contents of the User-Agent header of the request.

**morgan.compile(format)**

Compile a format string into a format function for use by morgan. A format string is a string that represents a single log line and can utilize token syntax. Tokens are references by :token-name. If tokens accept arguments, they can be passed using [], for example: :token-name[pretty] would pass the string 'pretty' as an argument to the token token-name.

The function returned from morgan.compile takes three arguments tokens, req, and res, where tokens is object with all defined tokens, req is the HTTP request and res is the HTTP response. The function will return a string that will be the log line, or undefined / null to skip logging.

Normally formats are defined using morgan.format(name, format), but for certain advanced uses, this compile function is directly available.

**Examples**

**express/connect**

Simple app that will log all request in the Apache combined format to STDOUT

var express = require('express')

var morgan = require('morgan')

var app = express()

app.use(morgan('combined'))

app.get('/', function (req, res) {

res.send('hello, world!')

})

**vanilla http server**

Simple app that will log all request in the Apache combined format to STDOUT

var finalhandler = require('finalhandler')

var http = require('http')

var morgan = require('morgan')

// create "middleware"

var logger = morgan('combined')

http.createServer(function (req, res) {

var done = finalhandler(req, res)

logger(req, res, function (err) {

if (err) return done(err)

// respond to request

res.setHeader('content-type', 'text/plain')

res.end('hello, world!')

})

})

**write logs to a file**

**single file**

Simple app that will log all requests in the Apache combined format to the file access.log.

var express = require('express')

var fs = require('fs')

var morgan = require('morgan')

var path = require('path')

var app = express()

// create a write stream (in append mode)

var accessLogStream = fs.createWriteStream(path.join(\_\_dirname, 'access.log'), {flags: 'a'})

// setup the logger

app.use(morgan('combined', {stream: accessLogStream}))

app.get('/', function (req, res) {

res.send('hello, world!')

})

**log file rotation**

Simple app that will log all requests in the Apache combined format to one log file per day in the log/ directory using the[rotating-file-stream module](https://www.npmjs.com/package/rotating-file-stream).

var express = require('express')

var fs = require('fs')

var morgan = require('morgan')

var path = require('path')

var rfs = require('rotating-file-stream')

var app = express()

var logDirectory = path.join(\_\_dirname, 'log')

// ensure log directory exists

fs.existsSync(logDirectory) || fs.mkdirSync(logDirectory)

// create a rotating write stream

var accessLogStream = rfs('access.log', {

interval: '1d', // rotate daily

path: logDirectory

})

// setup the logger

app.use(morgan('combined', {stream: accessLogStream}))

app.get('/', function (req, res) {

res.send('hello, world!')

})

**use custom token formats**

Sample app that will use custom token formats. This adds an ID to all requests and displays it using the :id token.

var express = require('express')

var morgan = require('morgan')

var uuid = require('node-uuid')

morgan.token('id', function getId (req) {

return req.id

})

var app = express()

app.use(assignId)

app.use(morgan(':id :method :url :response-time'))

app.get('/', function (req, res) {

res.send('hello, world!')

})

function assignId (req, res, next) {

req.id = uuid.v4()

next()

}