# serve-static

[NPM Version](https://npmjs.org/package/serve-static) [NPM Downloads](https://npmjs.org/package/serve-static) [Linux Build](https://github.com/expressjs/serve-static/actions/workflows/ci.yml) [Windows Build](https://ci.appveyor.com/project/dougwilson/serve-static) [Test Coverage](https://coveralls.io/r/expressjs/serve-static?branch=master)

## 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 serve-static

## API

var serveStatic = require('serve-static')

### serveStatic(root, options)

Create a new middleware function to serve files from within a given root directory. The file to serve will be determined by combining req.url with the provided root directory. When a file is not found, instead of sending a 404 response, this module will instead call next() to move on to the next middleware, allowing for stacking and fall-backs.

#### Options

##### acceptRanges

Enable or disable accepting ranged requests, defaults to true. Disabling this will not send Accept-Ranges and ignore the contents of the Range request header.

##### cacheControl

Enable or disable setting Cache-Control response header, defaults to true. Disabling this will ignore the immutable and maxAge options.

##### dotfiles

Set how "dotfiles" are treated when encountered. A dotfile is a file or directory that begins with a dot ("."). Note this check is done on the path itself without checking if the path actually exists on the disk. If root is specified, only the dotfiles above the root are checked (i.e. the root itself can be within a dotfile when set to "deny").

* 'allow' No special treatment for dotfiles.
* 'deny' Deny a request for a dotfile and 403/next().
* 'ignore' Pretend like the dotfile does not exist and 404/next().

The default value is similar to 'ignore', with the exception that this default will not ignore the files within a directory that begins with a dot.

##### etag

Enable or disable etag generation, defaults to true.

##### extensions

Set file extension fallbacks. When set, if a file is not found, the given extensions will be added to the file name and search for. The first that exists will be served. Example: ['html', 'htm'].

The default value is false.

##### fallthrough

Set the middleware to have client errors fall-through as just unhandled requests, otherwise forward a client error. The difference is that client errors like a bad request or a request to a non-existent file will cause this middleware to simply next() to your next middleware when this value is true. When this value is false, these errors (even 404s), will invoke next(err).

Typically true is desired such that multiple physical directories can be mapped to the same web address or for routes to fill in non-existent files.

The value false can be used if this middleware is mounted at a path that is designed to be strictly a single file system directory, which allows for short-circuiting 404s for less overhead. This middleware will also reply to all methods.

The default value is true.

##### immutable

Enable or disable the immutable directive in the Cache-Control response header, defaults to false. If set to true, the maxAge option should also be specified to enable caching. The immutable directive will prevent supported clients from making conditional requests during the life of the maxAge option to check if the file has changed.

##### index

By default this module will send "index.html" files in response to a request on a directory. To disable this set false or to supply a new index pass a string or an array in preferred order.

##### lastModified

Enable or disable Last-Modified header, defaults to true. Uses the file system's last modified value.

##### maxAge

Provide a max-age in milliseconds for http caching, defaults to 0. This can also be a string accepted by the [ms](https://www.npmjs.org/package/ms#readme) module.

##### redirect

Redirect to trailing "/" when the pathname is a dir. Defaults to true.

##### setHeaders

Function to set custom headers on response. Alterations to the headers need to occur synchronously. The function is called as fn(res, path, stat), where the arguments are:

* res the response object
* path the file path that is being sent
* stat the stat object of the file that is being sent

## Examples

### Serve files with vanilla node.js http server

var finalhandler = require('finalhandler')

var http = require('http')

var serveStatic = require('serve-static')

// Serve up public/ftp folder

var serve = serveStatic('public/ftp', { index: ['index.html', 'index.htm'] })

// Create server

var server = http.createServer(function onRequest (req, res) {

serve(req, res, finalhandler(req, res))

})

// Listen

server.listen(3000)

### Serve all files as downloads

var contentDisposition = require('content-disposition')

var finalhandler = require('finalhandler')

var http = require('http')

var serveStatic = require('serve-static')

// Serve up public/ftp folder

var serve = serveStatic('public/ftp', {

index: false,

setHeaders: setHeaders

})

// Set header to force download

function setHeaders (res, path) {

res.setHeader('Content-Disposition', contentDisposition(path))

}

// Create server

var server = http.createServer(function onRequest (req, res) {

serve(req, res, finalhandler(req, res))

})

// Listen

server.listen(3000)

### Serving using express

#### Simple

This is a simple example of using Express.

var express = require('express')

var serveStatic = require('serve-static')

var app = express()

app.use(serveStatic('public/ftp', { index: ['default.html', 'default.htm'] }))

app.listen(3000)

#### Multiple roots

This example shows a simple way to search through multiple directories. Files are searched for in public-optimized/ first, then public/ second as a fallback.

var express = require('express')

var path = require('path')

var serveStatic = require('serve-static')

var app = express()

app.use(serveStatic(path.join(\_\_dirname, 'public-optimized')))

app.use(serveStatic(path.join(\_\_dirname, 'public')))

app.listen(3000)

#### Different settings for paths

This example shows how to set a different max age depending on the served file type. In this example, HTML files are not cached, while everything else is for 1 day.

var express = require('express')

var path = require('path')

var serveStatic = require('serve-static')

var app = express()

app.use(serveStatic(path.join(\_\_dirname, 'public'), {

maxAge: '1d',

setHeaders: setCustomCacheControl

}))

app.listen(3000)

function setCustomCacheControl (res, path) {

if (serveStatic.mime.lookup(path) === 'text/html') {

// Custom Cache-Control for HTML files

res.setHeader('Cache-Control', 'public, max-age=0')

}

}

## License

[MIT](http://license)