file-send

> A http file send
>
> 

Installation

```
$ npm install file-send
```

API

```
var http = require('http'),
  FileSend = require('file-send'),
  Send = FileSend('/', {
    etag: false,
    maxAge: '30d'
  });

http.createServer(function (req, res){
  Send.use(req, res) // Create a new send stream
    .transfer(); // Send stream to client
});
```

FileSend(root, [options])

Create a new `Send` for the given `root` path.

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).

The default value is `'ignore'`.

- `'allow'` No special treatment for dotfiles.
- `'deny'` Send a 403 for any request for a dotfile.
- `'ignore'` Pretend like the dotfile does not exist and 404.

etag

Enable or disable etag generation, defaults to true.

extensions

If a given file doesn't exist, try appending one of the given extensions, in the given order. By default, this is disabled (set to `false`). An example value that will serve extension-less HTML files: `['html', 'htm']`.
This is skipped if the requested file already has an extension.

index

By default send supports "index.html" files, 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 module.

Events

```
var stream = Send.use(req, res); // The send.use return a new send stream
```

The `stream` is an event emitter and will emit the following events:

- `error` an error occurred `(err)`
- `directory` a directory was requested
- `file` a file was requested `(path, stat)`
- `headers` the headers are about to be set on a file `(res, path, stat)`
- `stream` file streaming has started `(stream)`
- `end` streaming has completed

stream.transfer()

The `transfer` method is used to pipe the response into the Node.js HTTP response object, typically `send.use(req, res).transfer()`.

stream.redirect(url)

redirect url, if header already send, do nothing.

stream.error(status, [error])

emit http error, if header already send will end the response with error message and status.

stream.send(path, stat)

The basic interface, send a file stream to response no filter.
If it is not necessary to do not use.

Error-handling

By default when no `error` listeners are present an automatic response will be made, otherwise you have full control over the response, aka you may show a 5xx page etc.

Caching

It does *not* perform internal caching, you should use a reverse proxy cache such as Varnish for this, or those fancy things called CDNs. If your application is small enough that it would benefit from single-node memory caching, it's small enough that it does not need caching at all ;).

Debugging

To enable `debug()` instrumentation:

```
$ node app -v
```

or:

```
$ node app -verbose
```

Running tests

```
$ npm install
$ npm test
```

Examples

Serving from a root directory with custom error-handling:

```
var http = require('http'),
  FileSend = require('file-send'),
  Send = FileSend('/www/example.com/public'); // Set root

var app = http.createServer(function(req, res){
  // Your custom error-handling logic:
  function error(err) {
    res.statusCode = err.status || 500;
    res.end(err.message);
  }

  // Your custom headers
  function headers(res, path, stat) {
    // serve all files for download
    res.setHeader('Content-Disposition', 'attachment');
  }

  // Your custom directory handling logic:
  function directory(path, stat) {
    // TODO You can do something here
    // Like displays the current directory file list
  }

  // Transfer arbitrary files from within /www/example.com/public/*
  Send.use(req, res)
    .on('error', error)
    .on('directory', directory)
    .on('headers', headers)
    .transfer();
}).listen(3000);
```

License

MIT