markdown-pdf build passing (https://travis-ci.org/alanshaw/markdown-pdf) dependencies up to date (https://david-dm.org/alanshaw/markdown-pdf) coverage 88% (https://coveralls.io/r/alanshaw/markdown-pdf?branch=master)

Node module that converts Markdown files to PDFs.

The PDF looks great because it is styled by HTML5 Boilerplate. What? - Yes! Your Markdown is first converted to HTML, then pushed into the HTML5 Boilerplate `index.html`. Phantomjs renders the page and saves it to a PDF. You can even customise the style of the PDF by passing an optional path to your CSS *and* you can pre-process your markdown file before it is converted to a PDF by passing in a pre-processing function, for templating.

Getting started

```
npm install markdown-pdf
```

Example usage

```
var markdownpdf = require("markdown-pdf")
  , fs = require("fs")

fs.createReadStream("/path/to/document.md")
  .pipe(markdownpdf())
  .pipe(fs.createWriteStream("/path/to/document.pdf"))

// --- OR ---

markdownpdf().from("/path/to/document.md").to("/path/to/document.pdf", function
() {
  console.log("Done")
})
```

Options

Pass an options object (`markdownpdf({/* options */})`) to configure the output.

options.cwd

Type: `String`
Default value: `process.cwd()`

Current working directory.

options.phantomPath

Type: `String`
Default value: Path provided by phantomjs module

Path to the phantomjs binary.

options.cssPath

Type: `String`
Default value: `[module path]/markdown-pdf/css/pdf.css`

Path to custom CSS file, relative to the current directory.

options.highlightCssPath

Type: `String`
Default value: `[module path]/markdown-pdf/css/highlight.css`

Path to custom highlight CSS file (for code highlighting with highlight.js (https://highlightjs.org)), relative to the current directory.

options.paperFormat

Type: `String`
Default value: `A4`

'A3', 'A4', 'A5', 'Legal', 'Letter' or 'Tabloid'.

options.paperOrientation

Type: `String`
Default value: `portrait`

'portrait' or 'landscape'.

options.paperBorder

Type: `String`
Default value: `2cm`

Supported dimension units are: 'mm', 'cm', 'in', 'px'

options.runningsPath

Type: `String`
Default value: `runnings.js`

Path to CommonJS module which sets the page header and footer (see runnings.js (runnings.js)).

options.renderDelay

Type: `Number`
Default value: Time until `page.onLoadFinished` (http://phantomjs.org/api/webpage/handler/on-load-finished.html) event fired

Delay (in ms) before the PDF is rendered.

options.loadTimeout

Type: `Number`
Default value: `10000`

If `renderDelay` option isn't set, this is the timeout (in ms) before the page is rendered in case the `page.onLoadFinished` event doesn't fire.

options.preProcessMd

Type: `Function`
Default value: `function () { return through() }`

A function that returns a through2 stream (https://npmjs.org/package/through2) that transforms the markdown before it is converted to HTML.

options.preProcessHtml

Type: `Function`
Default value: `function () { return through() }`

A function that returns a through2 stream (https://npmjs.org/package/through2) that transforms the HTML before it is converted to PDF.

options.remarkable

Type: `object`
Default value: `{ html: true, breaks: true }`

A config object that is passed to remarkable (https://www.npmjs.com/package/remarkable#options), the underlying markdown parser.

options.remarkable.preset

Type: `String`
Default value: `default`

Use remarkable presets (https://www.npmjs.com/package/remarkable#presets) as a convenience to quickly enable/disable active syntax rules and options for common use cases.

Supported values are `default`, `commonmark` and `full`

options.remarkable.plugins

Type: `Array` of remarkable-plugin `Functions`
Default value: `[]`

An array of Remarkable plugin functions, that extend the markdown parser functionality.

options.remarkable.syntax

Type: `Array` of optional remarkable syntax `Stringss`
Default value: `[]`

An array of [optional Remarkable syntax extensions (https://github.com/jonschlinkert/remarkable#syntax-extensions)](https://github.com/jonschlinkert/remarkable#syntax-extensions), disabled by default, that extend the markdown parser functionality.

API

from.path(path, opts) / from(path, opts)

Create a readable stream from `path` and pipe to markdown-pdf. `path` can be a single path or array of paths.

from.string(string)

Create a readable stream from `string` and pipe to markdown-pdf. `string` can be a single string or array of strings.

concat.from.paths(paths, opts)

Create and concatinate readable streams from `paths` and pipe to markdown-pdf.

concat.from.strings(strings, opts)

Create and concatinate readable streams from `strings` and pipe to markdown-pdf.

to.path(path, cb) / to(path, cb)

Create a writeable stream to `path` and pipe output from markdown-pdf to it. `path` can be a single path, or array of output paths if you specified an array of inputs. The callback function `cb` will be invoked when data has finished being written.

to.buffer(opts, cb)

Create a [concat-stream (https://npmjs.org/package/concat-stream)](https://npmjs.org/package/concat-stream) and pipe output from markdown-pdf to it. The callback function `cb` will be invoked when the buffer has been created.

to.string(opts, cb)

Create a [concat-stream (https://npmjs.org/package/concat-stream)](https://npmjs.org/package/concat-stream) and pipe output from markdown-pdf to it. The callback function `cb` will be invoked when the string has been created.

More examples

From string to path

```
var markdownpdf = require("markdown-pdf")

var md = "foo===\n* bar\n* baz\n\nLorem ipsum dolor sit"
  , outputPath = "/path/to/doc.pdf"

markdownpdf().from.string(md).to(outputPath, function () {
  console.log("Created", outputPath)
})
```

From multiple paths to multiple paths

```
var markdownpdf = require("markdown-pdf")

var mdDocs = ["home.md", "about.md", "contact.md"]
  , pdfDocs = mdDocs.map(function (d) { return "out/" + d.replace(".md",
".pdf") })

markdownpdf().from(mdDocs).to(pdfDocs, function () {
  pdfDocs.forEach(function (d) { console.log("Created", d) })
})
```

Concat from multiple paths to single path

```
var markdownpdf = require("markdown-pdf")

var mdDocs = ["chapter1.md", "chapter2.md", "chapter3.md"]
  , bookPath = "/path/to/book.pdf"

markdownpdf().concat.from(mdDocs).to(bookPath, function () {
  console.log("Created", bookPath)
})
```

Transform markdown before conversion

```
var markdownpdf = require("markdown-pdf")
  , split = require("split")
  , through = require("through")
  , duplexer = require("duplexer")

function preProcessMd () {
  // Split the input stream by lines
  var splitter = split()

  // Replace occurences of "foo" with "bar"
  var replacer = through(function (data) {
    this.queue(data.replace(/foo/g, "bar") + "\n")
  })

  splitter.pipe(replacer)
  return duplexer(splitter, replacer)
}

markdownpdf({preProcessMd: preProcessMd})
  .from("/path/to/document.md")
  .to("/path/to/document.pdf", function () { console.log("Done") })
```

Remarkable options and plugins

Example using [remarkable-classy (https://www.npmjs.com/package/remarkable-classy)](https://www.npmjs.com/package/remarkable-classy) plugin:

```
var markdownpdf = require("markdown-pdf")

var options = {
    remarkable: {
        html: true,
        breaks: true,
        plugins: [ require('remarkable-classy') ],
        syntax: [ 'footnote', 'sup', 'sub' ]
    }
}

markdownpdf(options)
  .from("/path/to/document.md")
  .to("/path/to/document.pdf", function () { console.log("Done") })
```

CLI interface

Installation

To use markdown-pdf as a standalone program from the terminal run

```
npm install -g markdown-pdf
```

Usage

```
Usage: markdown-pdf [options] <markdown-file-path>

Options:

  -h, --help                                output usage information
  -V, --version                             output the version number
  <markdown-file-path>                      Path of the markdown file to convert
  -c, --cwd [path]                          Current working directory
  -p, --phantom-path [path]                 Path to phantom binary
  -h, --runnings-path [path]                Path to runnings (header, footer)
  -s, --css-path [path]                     Path to custom CSS file
  -z, --highlight-css-path [path]           Path to custom highlight-CSS file
  -m, --remarkable-options [json]           Options to pass to Remarkable
  -f, --paper-format [format]               'A3', 'A4', 'A5', 'Legal', 'Letter' or
'Tabloid'
  -r, --paper-orientation [orientation]     'portrait' or 'landscape'
  -b, --paper-border [measurement]          Supported dimension units are: 'mm',
'cm', 'in', 'px'
  -d, --render-delay [millis]               Delay before rendering the PDF
  -t, --load-timeout [millis]               Timeout before the page is rendered in
case `page.onLoadFinished` isn't fired
  -o, --out [path]                          Path of where to save the PDF
```