META DATA

INTRODUCTION

Meta data goes at the top of documents, and is defined by any character that repeats 3 or more times. For example, `---` is the most common usage, but you can also use `###` or whatever repeats 3 or more times. By default, we parse the meta data with YAML but you can also use CSON by doing `--- cson` instead.

An example document that uses meta data will look like this:

```
---title: "Example Document"layout: "default"---My example document content
```

SPECIAL META DATA

FOR FILES & DOCUMENTS

`title`

The title for the document. Useful for headings.

`layout`

Tells DocPad what layout to use from the `layouts` folder. Layout files can use other layouts for advanced layout nesting.

is just a naming choice, rather than an implicit default.

`name`

Defaults to the `filename`. The name of the document. Useful for listings.

`date`

Defaults to `mtime`. Useful for setting a custom date via your documents meta data.

`slug`

Defaults to a slugified version of the `relativeBase`. Appears deprecated in favour of `url`.

`url`

The URL that you would like to use as the primary URL for the document. When a user accesses a document via a secondary URL, the user will be redirected to the primary URL automatically.

`urls`

The secondary URLs for a document. It can be a comma seperated values list, or an array of values.

`ignored`

Defaults to `false`. If set to `true`, the document will not be parsed. Useful for draft documents.

`standalone`

Defaults to `false`. If set to `true`, when a change is detected for the document, we will only regenerate this document and not anything else (e.g., documents with `referencesOthers` set to `true`).

FOR DOCUMENTS

`referencesOthers`

Defaults to `false`. If set to `true`, this document will be regenerated when a change occurs in another document. It is automatically set to `true` whenever a template helper is called that references another document. This makes so for instance on a blog listing page, when a blog post is changed, we will also regenereate the listing as well as the blog post.

`tags`

Defaults to `[]`. Tags can be a comma separated values list, or an array of values. While DocPad doesn't use tags for anything specifically, it is nice to have it handled uniformly across websites without you having to do it yourself.

`dynamic`

Defaults to `false`. If set to `true`, the document will be re-rendered on each request. This also adds the `req` object to the template data.

SPECIAL ATTRIBUTES

FOR FILES & DOCUMENTS

`id`

`relativePath`, we set the `id` to that instead.

`basename`

The file's name without the extension.

`extension`

The file's last extension (e.g., will be set to `eco` for the file `hello.md.eco`).

`outExtension`

The extension used for the output file. Same method as `extension` however it takes layouts into account as well.

`extensions`

The file's extensions as an array (e.g., will be set to `["md","eco"]` for the file `hello.md.eco`).

`filename`

The file's name with the extension.

`path`

The full path of our source file.

`outPath`

The full path of our output file.

`dirPath`

The full directory path of our source file.

`outDirPath`

The full directory path of our output file.

`outFilename`

The file's name with the output extension.

`relativePath`

The relative path of our source file.

`relativeOutPath`

The relative path of our output file.

`relativeDirPath`

The relative directory path of our source file.

`relativeOutDirPath`

The relative directory path of our output file.

`relativeBase`

The relative path of our source file without the file's extension.

`contentType`

The MIME content-type for the source file.

The MIME content-type for the output file.

`ctime`

The `Date` object for when this file was created.

`mtime`

The `Date` object for when this file was modified.

`encoding`

The encoding of the file. Either `binary` or `utf8`.

`source`

When `encoding` isnt `binary`, this is set to the raw contents of the file, stored as a string.

`content`

When `encoding` isnt `binary`, this is set to the contents of the file, stored as a string. This is used internally during the rendering process, **end-users should never use this property, instead they should either `source` or `contentRendered` depending on the use case.**

FOR DOCUMENTS

`write`

Defaults to `true`. Whether or not this document should be written to the

`render`

Defaults to `true`. Whether or not this document should be rendered.

`header`

The file meta data (header) in String format before it has been parsed.

`parser`

Defaults to `yaml`. The parser we used to parse the document's meta data header.

`body`

The file content (without the meta data header) before we've rendered it.

`rendered`

Defaults to `false`. Set to `true` once we have been rendered.

`contentRendered`

The rendered content (after is has been wrapped in the layouts).

`contentRenderedWithoutLayouts`

The rendered content (before being wrapped by the layouts).

METHODS

Refer to the Backbone Model Documentation

FOR FILES & DOCUMENTS

`toJSON()`

Same as the Backbone Model toJSON, but will also call `toJSON` on the original meta data to `meta` within the result.

`getMeta()`

Get the meta data Backbone Model for the file.

`setMeta(attrs)`

Same as the Backbone Model Set, but for the meta data Model.

`setDefaults(attrs)`

Same as the Backbone Model Set, but will only set attributes that haven't already been set to something.

`setMetaDefaults(attrs)`

Same as the Backbone Model Set, for the meta data but will only set the meta data Model attributes that haven't already been set to something.

`setData(data)`

Used for setting data of a virtual file (a file that does not have physical path).

`getData()`

Used for getting the data of a virtual file (a file that does not have physical

`setBuffer(buffer)`

Used for setting the source buffer.

`getBuffer()`

Used for getting the source buffer.

`setStat(stat)`

Used for setting the stat of the file.

`getStat()`

Used for getting the stat of the file.

`getContent()`

Used for getting the parsed source content or the buffer instance if it is a binary file.

`getOutContent()`

Used for getting the rendered content.

`isText()`

Is the file a text file?

`isBinary()`

Is the file a binary file?

`setUrl(url)`

Set the primary URL for the file.

`addUrl(url)`

Set a secondary URL for the file.

`removeUrl(url)`

Remove a URL for the file.

`getPath(relativePath, parentPath)`

Gets a path relative to the file.

FOR DOCUMENTS

`referencesOthers(flag?=true)`

Whether or not this document references another document. Sets the `referencesOthers` flag.