

## Dossier Markdown Showcase

# Table of Contents

- 1      **About Dossier Markdown.....3**
- 2      **Writing Documents .....3**
  - 2.1    Writing simple text ..... 3
  - 2.2    Decorating text ..... 3
  - 2.3    Lists and enumerations ..... 4
  - 2.4    Definition lists..... 5
  - 2.5    Checklists ..... 5
  - 2.6    Adding structure..... 5
  - 2.7    Quoting text ..... 6
  - 2.8    Quoting code ..... 6

# 1 About Dossier Markdown

The “Dossier” documentation system uses a variant of the widely used “Markdown” markup language.

The markup language is specified in the [CommonMark Specification](#).

This showcase explains the use of the most prominent features of the markup language used by “Dossier”, and how they are rendered in the different output formats.

## 2 Writing Documents

The following sections demonstrate how to write documents in the “Dossier” markup language.

### 2.1 Writing simple text

Writing simple text is as simple as it can be: just type away the text. For example, the following markup

```
This is a simple paragraph containing normal text.
```

```
This is another simple paragraph containing normal text.
```

will be rendered as

```
This is a simple paragraph containing normal text.
```

```
This is another simple paragraph containing normal text.
```

### 2.2 Decorating text

Decorating text, e.g. to put emphasis on some terms, is not much more difficult. Those are the options:

```
This paragraph has _italic_ text.
```

```
This one has **bold** text.
```

will be rendered as

```
This paragraph has italic text.
```

```
This one has bold text.
```

## 2.3 Lists and enumerations

There are two basic types of lists: ordered lists, aka enumerations, and unordered lists. Both are easy to write:

Unordered list:

- \* first item
- \* second item
- \* third item

will be rendered as

Unordered list:

- first item
- second item
- third item

Ordered list:

1. first item
2. second item
3. third item

will be rendered as

Ordered list:

1. first item
2. second item
3. third item

Both types of lists can be nested:

1. first item
2. second item
  - \* first subitem
  - \* second subitem
  - \* third subitem
3. third item

will be rendered as

1. first item
2. second item
  - first subitem
  - second subitem

- ▶ third subitem
- 3. third item

## 2.4 Definition lists

Definition lists can be used, for example, to define terms:

Foofoo  
: This is a Foofoo

Killroy  
: This is a Killroy

will be rendered as

**Foofoo**  
This is a Foofoo

**Killroy**  
This is a Killroy

## 2.5 Checklists

Checklists can be added like this:

- \* [ ] do something
- \* [ ] save the world
- \* [x] read a book
- \* [ ] do something else

will be rendered as

- ☐ do something
- ☐ save the world
- ☒ read a book
- ☐ do something else

## 2.6 Adding structure

Structure can be added to a document by using section headers:

### Second level heading

### Another second level heading

#### And a third level

## 2.7 Quoting text

There are several options available to quote text. First the simple inline quotes:

This is a paragraph containing a `<q>html quote</q>`.

This is a paragraph containing a `'straight single quote'`.

This is a paragraph containing a `"straight double quote"`.

This is a paragraph containing a `'typographic single quote'`.

This is a paragraph containing a `"typographic double quote"`.

will be rendered as

```
This is a paragraph containing a 'straight single quote'.
This is a paragraph containing a "straight double quote".
This is a paragraph containing a 'typographic single quote'.
This is a paragraph containing a "typographic double quote".
```

Alternatively, sections of text can be quoted using a blockquote:

```
> This is a quoted section of text.
```

will be rendered as

```
This is a quoted section of text.
```

## 2.8 Quoting code

For quoting code or code sections, there are again two options:

1. inline code
2. code sections

Inline ``code`` has ``back-ticks around`` it.

will be rendered as

Inline `code` has `back-ticks` around it.

```
git checkout -b featureA
```

## Dossier Markdown Showcase

will be rendered as

```
git checkout -b featureA
```