Org Mode

The Emacs of plain-text formats

Timothy

2020-07-21

Plain text formats

Known uses

When I say “plain text”, what comes to mind?

Known uses

When I say “plain text”, what comes to mind?

- Source code

Known uses

When I say “plain text”, what comes to mind?

- Source code
- `README` files

Known uses

When I say “plain text”, what comes to mind?

- Source code
- `README` files
- LaTeX
- `HTML`

Known uses

When I say “plain text”, what comes to mind?

- Source code
- `README` files
- $\LaTeX$
- `HTML`

All of these essentially consist of content and markup/syntax.

Markdown

The most common plaintext markup syntax, created in 2004.

```markdown
# This is markdown

For **bold** we use twice as many asterics as neccesary, then use the same
charachter for *italic*. But wait there's another way to do _italics_, it just
seems like it should be underline. Thankfully there's only one way to do
[links](https://tecosaur.com), or is it (a link)[https://tecosaur.com]? Every so
often I forget the order, and then it's a pain. At least `inline-code` is simple.

By the way, did I mention there are about 40 functionally different markdown
↪  implementations?
(see [here](https://github.com/commonmark/commonmark-spec/wiki/markdown-flavors))
```

Org-mode

A format developed within Emacs, created in 2003. Other systems have begun work on supporting the format.

```
* This is Org-mode

We have a single-emphasis-charachter *bold*, and sensible /italics/, which
makes a nice change.
There's only one way of doing [[*This is Org-mode][links]],
but unfortunately I now need to press shift for ~inline code~.

We also get =verbatim= though, and there's only one implementation of this,
so no fragmentation.
```

One moment. . .

From our first slide, “Content and markup”.

One moment. . .

From our first slide, “Content and markup”.

That applies to far more that the examples we listed. Consider:

One moment...

From our first slide, "Content and markup".

That applies to far more that the examples we listed. Consider:

- Presentations

One moment. . .

From our first slide, “Content and markup”.

That applies to far more that the examples we listed. Consider:

- Presentations
- Spreadsheets

One moment. . .

From our first slide, “Content and markup”.

That applies to far more that the examples we listed. Consider:

- Presentations
- Spreadsheets
- (Uni) notes, PKBs
- Reports/Papers

One moment. . .

From our first slide, “Content and markup”.

That applies to far more that the examples we listed. Consider:

- Presentations
- Spreadsheets
- (Uni) notes, PKBs
- Reports/Papers
- Interactive notebooks

One moment. . .

From our first slide, “Content and markup”.

That applies to far more that the examples we listed. Consider:

- Presentations
- Spreadsheets
- (Uni) notes, PKBs
- Reports/Papers
- Interactive notebooks
- Books

One moment. . .

From our first slide, “Content and markup”.

That applies to far more that the examples we listed. Consider:

- Presentations
- Spreadsheets
- (Uni) notes, PKBs
- Reports/Papers
- Interactive notebooks
- Books
- Todo lists / task management

One moment. . .

From our first slide, “Content and markup”.

That applies to far more that the examples we listed. Consider:

- Presentations
- Spreadsheets
- (Uni) notes, PKBs
- Reports/Papers
- Interactive notebooks
- Books
- Todo lists / task management

That’s . . . a lot

Consider the overlap

For almost all of the different use cases just mentioned, we use a different app for almost each use. In each app though, we spend most of our time just working with text.

Consider the overlap

For almost all of the different use cases just mentioned, we use a different app for almost each use. In each app though, we spend most of our time just working with text.

For a single task, we'll often use multiple apps, then struggle to weave a workflow together involving them all. Most apps are well aware of this, and so have put a good deal of effort into integrations (e.g. the Microsoft Office series).

Consider the overlap

For almost all of the different use cases just mentioned, we use a different app for almost each use. In each app though, we spend most of our time just working with text.

For a single task, we'll often use multiple apps, then struggle to weave a workflow together involving them all. Most apps are well aware of this, and so have put a good deal of effort into integrations (e.g. the Microsoft Office series).

Reflecting on this however, this state of affairs seems a bit sub-optimal.

Example — Comp. Sci. Researcher

Consider the following hypothetical, situation:

You are a computer science researcher. You have a nice idea which re-imagines the standard method of implementing a common algorithm. You want to try it out, then maybe write a paper/blog post.

Comp Sci Researcher Hypothetical

Search the existing literature you'd want some sort of notes file, probably a word doc (`.docx`), or other plaintext format (`.md`).

Write up you idea so you don't forget likely use a (second) word or `.md` doc, or if some maths is involved — LaTeX (`.tex`)

Test your idea Open up a notebook to prototype some code, using something like Jupyter (`.ipynb`)

Formally code your idea up Use you code editor of choice, and whatever language

Tabulate speed comparison A basic spreadsheet seems appropriate, `.xlsx`.

Write up a paper The industry standard is LaTeX (`.tex`)

Share a blog post on your website Convert aspects of you paper (`.tex`) to `.md` or some other format that web CMS tools use, probably with examples from your notebook (`.ipynb`) and maybe your speed comparison (`.xlsx`)

Wow .. that's about six different applications/formats, and your work is spread across about 5-8 files too!

A note on workflow

You can do productive, maintainable and reproducible work with all kinds of different software set-ups. So this discussion is not geared toward convincing you there is 'One True Way' to organize things. I do think, however, that if you're in the early phase of your career, it's worth giving some thought to how you're going to organize and manage your work. Here, I will just try to detail one particular method, and the features which I think make it a particularly worthy candidate.

My paraphrased adaptation of Kieran's introduction in "Choosing Your Workfow Applications"

Org-mode

What's with the name?

Why is "mode" in the name?

What’s with the name?

Why is “mode” in the name?

It’s because name of the emacs ’mode’ that this format was designed/implemented in.

What is it?

A plain text markup format, like we mentioned earlier.

What is it?

A plain text markup format, like we mentioned earlier.

It's designed to be easily manipulated by both humans (like markdown) and code (like JSON).

What is it?

A plain text markup format, like we mentioned earlier.

It's designed to be easily manipulated by both humans (like markdown) and code (like JSON).

The mode supplies a syntax, and tools, for programmatically handling structured text, and an ecosystem built upon that fundamental ability. And it runs inside Emacs.

What makes it good?

Possibilities allowed for by that last paragraph. Particularly:

- The *rich* functionally of the `Org-mode` mode
- Org-babel
- Integration

What can you do with it

Now I demo a lot.

What can you do with it

Now I demo a lot.

Please, please, *please* talk to me. We'll both get the most out of this if this segment is highly interactive.

Resources

- Doom Emacs
- What Even Is Org Mode? - Atomized
- Choosing Your Workflow Application (pdf)

And some recommendations:

- Org-mode tutorials | Pragmatic Emacs
- OrgMode E01S01: Headlines & outline mode - YouTube