Orgmode format render test

This page originates in an `org-mode` written document and has been written to create a reference of things that are (im-)possible using the direct org to html conversion. This text would be ignored by jekyll if the option `skip:t` was specified as an option. (**CHECK** : is the content available as excerpt then?)

The frontmatter is parsed by jekyll, obviously

Header level 1

This page contains org-mode constructs as a test to see how they get rendered to html. It's the visual last test to see if I have gotten the css stuff right. The page will be updated each time a css change is made so it can be checked without having to hunt for content which uses a special construct.

Header level 2

This page contains org-mode constructs as a test to see how they get rendered to html. It's the visual last test to see if I have gotten the css stuff right. The page will be updated each time a css change is made so it can be checked without having to hunt for content which uses a special construct.

Header level 3

This page contains org-mode constructs as a test to see how they get rendered to html. It's the visual last test to see if I have gotten the css stuff right. The page will be updated each time a css change is made so it can be checked without having to hunt for content which uses a special construct.

1. Header level 4

   This page contains org-mode constructs as a test to see how they get rendered to html. It's the visual last test to see if I have gotten the css stuff right. The page will be updated each time a css change is made so it can be checked without having to hunt for content which uses a special construct.

   (a) Header level 5

This page contains org-mode constructs as a test to see how they get rendered to html. It's the visual last test to see if I have gotten the css stuff right. The page will be updated each time a css change is made so it can be checked without having to hunt for content which uses a special construct

i. Header level 6

   This page contains org-mode constructs as a test to see how they get rendered to html. It's the visual last test to see if I have gotten the css stuff right. The page will be updated each time a css change is made so it can be checked without having to hunt for content which uses a special construct

   A. Header level 7

      This page contains org-mode constructs as a test to see how they get rendered to html. It's the visual last test to see if I have gotten the css stuff right. The page will be updated each time a css change is made so it can be checked without having to hunt for content which uses a special construct

      Apparently the maximum number of levels for pandoc conversion is 7. An error will be issued: "! LaTeX Error: Too deeply nested."

Blockquotes

> This paragraph was started with a $\text{begin}_{\text{quote}}$ construct

Pull quotes

Pull quotes are provided by a plugins and thus require a liquid template construct. This only works when you surround the complete paragraph in an $\text{begin}_{\text{html}}$ / end_{html} construct.

Footnotes

The documentation of `org-ruby` mentions support of footnotes, but the one[1] I created here does not render very nicely I think. I would have expected there to be a link.

Orgmode can do inline footnotes [2]

[1] This is a footnote
[2] Like this

Lists

- Level 1
  - Level 2
    - Level 3
  - Level 4
    - Level 5
      - Level 6

☐ orgmode has checkbox like list items
  ☐ but these don't do anything special

- term1 :: defined;
- term2 :: defined;
- term3 :: defined with http://example.com link

Code blocks

Code blocks can be produced in the usual way with #+BEGIN$_{\text{SRC}}$ blocks in org-mode. Highlighting is automatic when a pygments lexer is available.

```
def foo
  puts 'foo'
end
```

Same construct with the -n switch, which should turn on line numbers:

```
1 def foo
2   puts 'foo'
3 end
```

and the same construct with the +n switch, which should continue the linenumbering from the previous snippet

```
1 def foo
2   puts 'foo'
3 end
```

Other code related embedding is for githubs gist facility

Rules

In orgmode a line on its own with nothing else but at least 5 dashes is considered a rule.

So, when we use 4 we should just get 4 dashes: —-

Verse

Orgmode has a verse construct, but that does not give output (at the time of writing):

Great clouds overhead
Tiny black birds rise and fall
Snow covers Emacs

– AlexSchroeder

Centered

The center construct gets a text-align:center attribute, so this should work, given correct CSS.

Everything should be made as simple as possible,
but not any simpler

Examples

The example block format from orgmode

```
Some example from a text file.
```

LaTeX

Orgmode has extensive support for LaTeX fragments, but this doesn't transfer well with the exporter in use.

$$x = \sqrt{b} \tag{1}$$

If $a^2 = b$ and $b = 2$, then the solution must be either

$$a = +\sqrt{2}$$

or

$$a = -\sqrt{2}$$

.

$$x = \sqrt{b} \tag{2}$$

If $a^2 = b$ and $b = 2$, then the solution must be either

$$a = +\sqrt{2}$$

or

$$a = -\sqrt{2}$$

.

Test

In orgmode export this would be in a div with class test

Links

description

http link with description

http://example.com

Examples - internal link to header

- the text 'Orgmode' would link to here from everyhwere (radio target) in Orgmode

Orgmode has a way to define linkword, which I have done for this file linkword 'google' used

Images

Image links are just normal links in orgmode, which just happen to link to an image.

link is to image, this is the description

Typically my images will be hosted at either a site like flickr or at my own media-site running mediagoblin, so let's take an image from there.

Missing: image

Just linking to an image source will produce an image tag, unstyled. Typically in blog posts a more or less standard set of image types are needed:

1. Things that liven up the text, typically pull-left or pull-right in my blog. Images will not be that large;
2. Photographs which are the subject or part of the subject of the content. Typically larger and detailed, there may be a need to be able to zoom into these pictures

3. Pictorial elements, part of the theme, perhaps a one-off solution.

Strategies for getting images into a blog, given that we use org-mode our options are rather limited. I'd want to stay away as much as possible from using specific jekyll constructs, that is if we want the blog posting to be useful outside of jekyll.

The price for that is that we then cannot use liquid tags, unless wrapped in an html block and the only way we would make distinctly different images would be using a `#+ATTR_HTML:` construct.

How I want it to look:

Doing the same image in an *orgmode* way is cumbersome, so I'm using a plugin to do this. The goal is to have all media on media.mrblog.nl, which is probably too strict a limitation, but it allows to have very concise image definitions with a short liquid tag.

Here's how the tag looks:

```
{%raw%}
{% gmg shadowed 168 %}
{%endraw%}
```

The plugin does a request (on building) to the media site and retrieves some json data which we could use. (For the caption for example)

Result:

Emphasis

single asterisks: bold

single_underscores:_underline

single forward slashes: italic

`single equal signs: code`

~~single plus signs: strike-through~~

`single tildes: verbatim`

Use of liquid tags

Jekyll uses a template langed called liquid. This section is to see examples of tags and use of other liquid constructs

```
{% raw %}
Sitename taken from the variable site.name :: {{site.name}}
{{'this will be upcased by a filter' | upcase }}
{% endraw %}
```

will render as:

```
Sitename taken from the variable site.name :: {{site.name}}

{{'this will be upcased by a filter' | upcase }}
```

TODO This is a to-do item TEST

If nothing specific is configured, the TODO items is just treated as a header, and rendered without the TODO keyword. If a `todo:t` snippet is present in the #+OPTIONS than the TODO keyword will be shown. The keyword as such will be just part of the header, not rendered specially. There is however extra HTML generated, so the CSS can take care of the extra rendering if needed.