Meta-data syntax implemented by Maruku

This document describes a syntax for attaching meta-data to block-level elements (headers, paragraphs, code blocks,...), and to span-level elements (links, images,...).

Table of contents:

1. Overview

This proposal describes two additions to the Markdown syntax:

1. inline attribute lists (IAL)

```
## Header ##       {: key=val .class #id ref_id}
```

2. attribute lists definitions (ALD)

```
{:ref_id: key=val .class #id}
```

Every span-level or block-level element can be followed by an IAL:

```
### Header ###     {: #header1 class=c1}

Paragraph *with emphasis*{: class=c1}
second line of paragraph
{: class=c1}
```

In this example, the three IALs refer to the header, the emphasis span, and the entire paragraph, respectively.

IALs can reference ALDs. The result of the following example is the same as the previous one:

```
### Header ###  {: #header1 c1}

Paragraph *with emphasis*{:c1}
```

```
second line of paragraph
{:c1}

{:c1: class=c1}
```

2. Attribute lists

This is an example attribute list, which shows everything you can put inside:

```
{: key1=val key2="long val" #myid .class1 .class2 ref1 ref2}
```

More in particular, an attribute list is a whitespace-separated list of elements of 4 different kinds:

1. key/value pairs (quoted if necessary)
2. references to ALD (`ref1` , `ref2`)
3. id specifiers (`#myid`)
4. class specifiers (`.myclass`)

2.1. `id` and `class` are special

For ID and classes there are special shortcuts:

- `#myid` is a shortcut for `id=myid`
- `.myclass` means "add `myclass` to the current `class` attribute".
  So these are equivalent:

  ```
  {: .class1 .class2}
  {: class="class1 class2"}
  ```

The following attribute lists are equivalent:

```
{: #myid .class1 .class2}
{: id=myid class=class1 .class2}
{: id=myid class="class1 class2"}
{: id=myid class="will be overridden" class=class1 .class2}
```

3. Where to put inline attribute lists

3.1. For block-level elements

For paragraphs and other block-level elements, IAL go **after** the element:

```
This is a paragraph.
Line 2 of the paragraph.
{: #myid .myclass}

A quote with a citation url:
> Who said that?
{: cite=google.com}
```

Note: empty lines between the block and the IAL are not tolerated. So this is not legal:

```
This is a paragraph.
Line 2 of the paragraph.

{: #myid .myclass}
```

Attribute lists may be indented up to 3 spaces:

```
Paragraph1
␣{:ok}

Paragraph2
␣␣{:ok}

Paragraph2
␣␣␣{:ok}
```

3.2. For headers

For headers, you can put attribute lists on the same line:

```
### Header ###     {: #myid}

Header     {: #myid .myclass}
------
```

or, as like other block-level elements, on the line below:

```
### Header ###
{: #myid}

Header
------
{: #myid .myclass}
```

3.3. For span-level elements

For span-level elements, meta-data goes immediately **after** in the flow.

For example, in this:

```
This is a *chunky paragraph*{: #id1}
{: #id2}
```

the ID of the `em` element is set to `id1` and the ID of the paragraph is set to `id2`.

This works also for links, like this:

```
This is [a link][ref]{:#myid rel=abc rev=abc}
```

For images, this:

```
This is ![Alt text](url "fresh carrots")
```

is equivalent to:

```
This is ![Alt text](url){:title="fresh carrots"}
```

4. Using attributes lists definition

In an attribute list, you can have:

1. `key=value` pairs,
2. id attributes (`#myid`)
3. class attributes (`.myclass`)

Everything else is interpreted as a reference to an ALD.

```
# Header #      {:ref}

Blah blah blah.

{:ref: #myhead .myclass lang=fr}
```

Of course, more than one IAL can reference the same ALD:

```
# Header 1 #      {:1}
...
# Header 2 #      {:1}

{:1: .myclass lang=fr}
```

5. The rules

5.1. The issue of escaping

1. No escaping in code spans/blocks.
2. Everywhere else, **all** PUNCTUATION characters **can** be escaped, and **must** be escaped when they could trigger links, tables, etc.

   A punctuation character is anything not a letter, a number, or whitespace (`[^a-zA-Z0-9\s\n]`).

3. As a rule, quotes **must** be escaped inside quoted values:

   - Inside `"quoted values"`, you **must** escape `"`.
   - Inside `'quoted values'`, you **must** escape `'`.
   - Other examples:

`"bah 'bah' bah"` = `"bah \'bah\' bah"` = `'bah \'bah\' bah'`

`'bah "bah" bah'` = `'bah \"bah\" bah'` = `"bah \"bah\" bah"`

4. There is an exception for backward compatibility, in links/images titles:

```
[text](url "title"with"quotes")
```

   The exception is not valid for attribute lists and in other contexts, where you have to use the canonical syntax.

5.2. Syntax for attribute lists

Consider the following attribute list:

```
{: key=value ref key2="quoted value" }
```

In this string, `key`, `value`, and `ref` can be substituted by any string that does not contain whitespace, or the unescaped characters `}`, `=`, `'`, `"`.

Inside a quoted value you **must** escape the other kind of quote.

Also, you **must** escape a closing curly brace `}` inside quoted values. This rule is for making life easier for interpreter that just want to skip the meta-data.

5.3. Just skip, if you want

If you don't implement this syntax, you can get rid of the IAL by using this regular expression (this is written in Ruby):

```
r = /\{:(\\\}|[^\}])*\}/

s.gsub(r, '') # ignore metadata
```

Basically: match everything contained in a couple of `{:` and `}`, taking care of escaping of `}`. This `\\\}|[^\}]` means: eat either any character which is not a `}` or an escape sequence `\}`.

For this example,

```
this is
{: skipped="\}" val=\} bar}

for me
{: also this}
```

the result is:

```
this is


for me
```