The Bracetax syntax

1 Introduction

This document defines the rules of the Bracetax syntax and explain the semantics of its commands. Some examples give the expected behaviour for transformations in HTML and LaTeX using the `ocamlbracetax` library. It's also used as a complete example of bracetax syntax.

2 Rules

In the text there are *only* three special characters:

- The *opening brace*, `{`, which starts the command mode
- The *closing brace*, `}`, which finishes a command
- The *sharp*, `#`, which starts a comment to the end of the line

Those charcters can be escaped with the commands `{{}`, `{}}` and `{#}`.

All whitespace characters are considered as a space (`' '`).

A command has the form:

```
{cmd arg1 arg2 arg3|concerned text}
```

Between the `{` and the `|`, it's the command mode:

- `'\\'` escapes the backslash (used as escaper)
- `'\ '` escapes the space (used as argument separator)
- `'\|'` escapes the pipe (used as command/text separator)
- `'\{'` and `'\}'` escape the braces (except for the `{{}` and `{}}` commands)

A command can have no arguments and no text, *e.g.* `{br}`

And for long blocs: `{cmd arg|Some text}` is equivalent to `{begin cmd arg}Some text{end}`

Non-parsed (*i.e. "code"*) blocs have a special syntax (4.3)

For each command the arguments can be considered optional (they have default values).

3 Comments

Are only allowed in *text mode*:

```
Text # Comments...
{cmd arg|
    Text # Also comments...
}
{cmd # NOT COMMENTS !!! | Text }
```

4 Commands

4.1 Characters

Example:

```
Those are {{}, {}}, {#}, Unbreakable->{~}<-space,
horizontal{...}ellipsis, dashes: en{--}em{---}, and {utf 0x42}, {utf 339}.
```

Those are {, }, #, Unbreakable-> <-space, horizontal... ellipsis, dashes: en– em—, and B, œ.

Here are actually all the commands available for special characters:

Command	**Character**
`{{}`	*Left brace: '{'*
`{}}`	*Right brace: '}'*
`{#}`	*Sharp: '#'*
`{~}`	*Unbreakable space: ' '*
`{...}`	*Horizontal ellipsis: '...'*
`{--}`	*The en-dash: '–'*
`{---}`	*The em-dash: '—'*
`{utf <int>}`	*Any UTF char as integer.*

Table 1: Bracetax's character–commands.

Notes:

- In LaTeX, `{utf <int>}` depends on the `\unichar{...}` command which is far from complete.
- There are many different typographical dashes. Bracetax provides the two most used (together with the standard hyphen-minus which is on every keyboard: '-'). See wikipedia:Dash, wikipedia:Hyphen, wikipedia:Quotation_dash.

4.2 Header

To have a title bloc in you document:

```
{begin header}
    Ignored text !!
    {title|Bracetax}
    {authors|{link http://seb.mondet.org|Sebastien Mondet}}
```

```
    {subtitle|A simple syntax for wikis and more...}
{end}
```

4.3 Non parsed blocs

4.3.1 The code command

The default case:

```
{code}
My free    form {{ } {|} c
  code
{end}
```

But you can set the *end tag*:

```
{code AnotherEndTag}
 Still Free
 form
with:
{end}
inside the bloc
{AnotherEndTag}
```

Those blocs will be rendered as `<pre>` or `\begin{verbatim}`.

4.3.2 The bypass command

The command `{bypass}` has the same syntax than `{code}` but is used to output directly some content, e.g.:

```
{bypass endtag}Some <i>HTML</i>{endtag}
```

It can be used by preprocessors to implement special commands.

For security reasons, this command can be disabled (actually treated as the {code} command) in `ocamlbracetax` and in `brtx` (option `-deny-bypass`). Using {bypass} with a web application written with an interpreted language could lead to code injection issues...

4.3.3 The text command

The `{text}` command works also like `{code}`, but it used to treat all the content as "text", "braces" and "sharps", will have no effect. For example:

```
{text endtag} Some text with {braces, |pipes} and #sharps#.{endtag}
```

will give: Some text with {braces, |pipes} and #sharps#.

4.3.4 The ignore comand

The `{ignore}` command has also the same syntax as the `{code}` one, but is used to *ignore* some text, it can be used to implement so-called "block comments".

```
Some {ignore someendtag}uninteresting,
useless, and unreadable{someendtag} text
```

is equivalent to

```
Some #uninteresting,
#useless, and unreadable{someendtag}
 text
```

4.4 Formating

Some basic formatting:

```
{i|italic}, {b|bold},
{i|italo{b|bold}},
{t|mono-spaced},
super{sup|script}, sub{sub|script}
```

italic, **bold**, *italo**bold***, `mono-spaced`, super$^{\text{script}}$, sub$_{\text{script}}$

4.5 Quoting

4.5.1 Run-In Quotations

```
Quotes:
    {q|default},
    {q '|single},
    {q es|españolas},
    {q fr|françaises},
    {q en|English},
    {q de|Deutsch} and
    {q brout|unknown is default}
```

Quotes: “default”, ‘single’, «españolas», « françaises », “English”, „Deutsch“ and “unknown is default”.

4.5.2 Block Quotations

They are as simple as `{quote| ... }`. For example:

```
{begin quote}
There is a theory which states that if ever anybody discovers exactly
what the Universe is for and why it is here, it will instantly
disappear and be replaced by something even more bizarre and
inexplicable. There is another theory which states that this has
already happened.
{b|Douglas Adams}
{end}
```

will give:

> There is a theory which states that if ever anybody discovers exactly what the Universe is for and why it is here, it will instantly disappear and be replaced by something even more bizarre and inexplicable. There is another theory which states that this has already happened.
> **Douglas Adams**

4.6 Paragraphs and line-breaks

Paragraph: `{p}`, Line-break: `{br}`

4.7 Sections

The syntax is:

```
{section level label|Title of the section}
```

The `level` is an integer: 1, 2, 3 or 4 (default is 1)
The `label` is a string for local links (4.9)/references.

4.8 Lists

A list can be `item` (default) or `enum`, a new item begins with `{*}`:

```
{begin list enum}
  {*} The one
  {*} Two
  {*} Three
    {list item|{*} inside {*} more}
  {*} Four
{end}
```

1. The one
2. Two
3. Three
    - inside
    - more
4. Four

4.9 Links

Local links:

```
The {t|label} is a string for local {link local:sec_links|links}/references.
...
{section 2 sec_links|Links}
```

Other:

```
There's a {link http://www.vim.org|Vim} syntax file.
```

but don't forget the `http://` because "`www.any-site.bouh`" can be interpreted as a local (i.e. `./`) link.

A link may not have contents:

```
{link http://seb.mondet.org}
```

Will give: http://seb.mondet.org

4.10 Figures

The syntax for images is:

```
{image <src> [<w>px|<w>%] <label>|Caption text}
```

For example this image (1):

```
{image
    logo.png
    100px
    img:logo_candidate
    |An image; for now it is the only logo candidate{...}}
```

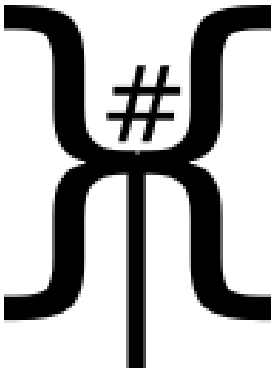

Figure 1: An image; for now it is the only logo candidate...

4.11 Tables

For table we have two commands, the `table`:

```
{table <nb-of-columns> <label> <default-align>|Caption and cells}
```

and the cell:

```
{c <format> <size>|content}
```

The `<format>` is the alignment ('l', 'r' or 'c'), the optional "is header ?" ('h'), in any order (`'_'` means "default").
The `<size>` argument is either the number of columns occupied or the total size *r*x*c* (numbers of rows and columns). Example:

```
{begin table 4 tab:example r}
{c|Simple cell} {c h|Header cell} {c rh|Header right cell} {c r|Right cell}
{c c|Center cell} {c l|Left cell} {c r 2|Right double cell}
{c lh|Left header cell} {c c 2|Center double cell}  {c hl|Left header cell}
{c l 2x3|Two rows, three columns, left} {c|Default, fourth in the row}
                                        {c h|Header, 4th in the row}
{c|Default, 1st} {c|Default, 2nd} {c _ 2x2|2 rows, 2 columns, default}
{c _ 2|Default, 1st and 2nd}
```

Simple cell	**Header cell**	**Header right cell**	Right cell
Center cell	Left cell		Right double cell
Left header cell	Center double cell		**Left header cell**
Two rows, three columns, left			Default, fourth in the row
			Header, 4th in the row
Default, 1st	Default, 2nd	2 rows, 2 columns, default	
Default, 1st and 2nd			
Simple cell	**Header cell**	**Header right cell**	Right cell

Table 2: Caption of the table, default align is right

```
{c|Simple cell} {c h|Header cell} {c rh|Header right cell} {c r|Right cell}
Caption of the table, default align is right
{end}
```

Note: The alignment doesn't seem to be in XHTML, so we use a `style` attribute *and* a `class` attribute (*e.g.* `<td class="centeralign" style="text-align:center;">`).

4.12 Notes

Notes are `\footnote`'s[1] or *sidenotes*[2]

```
Notes are {t|\footnote}'s{note|In LaTeX}
```

5 Post-processing plugin system

With the `code` environment you can also prepare post-processing:

```
{code end_tag name_for_the_plugin}
A   non - parsed    {b|block}
{end_tag}
```

Will add `verbatimbegin/end` comments, *e.g.* for XHTML:

```
<!--verbatimbegin:name_for_the_plugin -->
<pre>
A   non - parsed    {b|block}
</pre>
<!--verbatimend:name_for_the_plugin -->
```

or for LaTeX[3]:

```
%verbatimbegin:name_for_the_plugin
\begin{code}
A   non - parsed    {b|block}
\ end{code}
%verbatimend:name_for_the_plugin
```

Then you can post-process your output searching for those "comment-tags".

[1] In LaTeX

[2] In XHTML... with a CSS

[3] The space between '\' and 'end' is due the impossibility to put '`\end{verbatim}`' in a LaTeX `verbatim` evironment