

# Moodular

Use Typst's HTML export to generate content for your Moodle courses

v0.0.1

<https://github.com/SillyFreak/typst-moodular>

**Clemens Koza**

## CONTENTS

I	Introduction .....	2
II	Module reference .....	2
	II.a moodular .....	2
	II.b c4l .....	3
	II.c htmlx .....	9

## I INTRODUCTION

This package allows you to use Typst's HTML export to generate content for your Moodle courses, but preview them faithfully in Tinymist or the webapp.

This package is somewhat opinionated, as it contains features tailored towards specific Moodle plugins used at our own school:

- Generico: Moodular provides functions to insert Generico tags into your document
- Components for Learning (C4L) (Atto/TinyMCE plugin): Moodular recreates the HTML structure of C4L blocks, and also renders them for PDF export

## II MODULE REFERENCE

### II.a moodular

- `setup()`
- `frame()`

`setup()` -> `function`

Moodular's main Template function, to be called as a show rule:

```
1 #show: setup()
```

The function applies the following settings for PDF export:

- sets the page height to auto so that all content appears on a single continuous page, like on the web
- sets a sans serif font (Noto Sans is currently hardcoded and therefore needs to be present) and displays links blue and underlined
- shows blockquotes with a gray left border
- show raw blocks with a light beige background

which is a best-effort recreation of Moodle's default style.

For HTML export, the following settings are applied:

- raw blocks are wrapped in `<pre class="language-..."><code>` tags to apply Moodle's syntax highlighting
- image elements are transformed into `` tags. The `@@PLUGINFILE@@` prefix is used internally by Moodle to indicate references to uploaded files.

`frame(body: content) -> content`

Conditionally puts content into an `html.frame`, i.e. rendering it as an `<svg>` tag. When exporting to PDF, the content is simply displayed as-is.

#### Parameters:

`body (content)` – the content to put into a frame

## II.b c4l

- `key-concept()`
- `tip()`
- `reminder()`
- `quote()`
- `do-dont()`

- `reading-context()`
- `example()`
- `figure()`
- `attention()`
- `procedural-context()`

- `learning-outcomes()`
- `expected-feedback()`
- `card()`

```
key-concept(body: content, full-width) -> content
```

See <https://componentsforlearning.org/components/key-concept/> for details and examples rendered in HTML.

This is how the component looks in PDF preview:

```
1 c4l.key-concept[#lorem(20)]
```

typc

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.

### Parameters:

`full-width` (= `false`) – whether the component should take up the full text width

```
tip(body: content, full-width) -> content
```

See <https://componentsforlearning.org/components/tip/> for details and examples rendered in HTML.

This is how the component looks in PDF preview:

```
1 c4l.tip[#lorem(20)]
```

typc

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.



### Parameters:

`full-width` (= `false`) – whether the component should take up the full text width

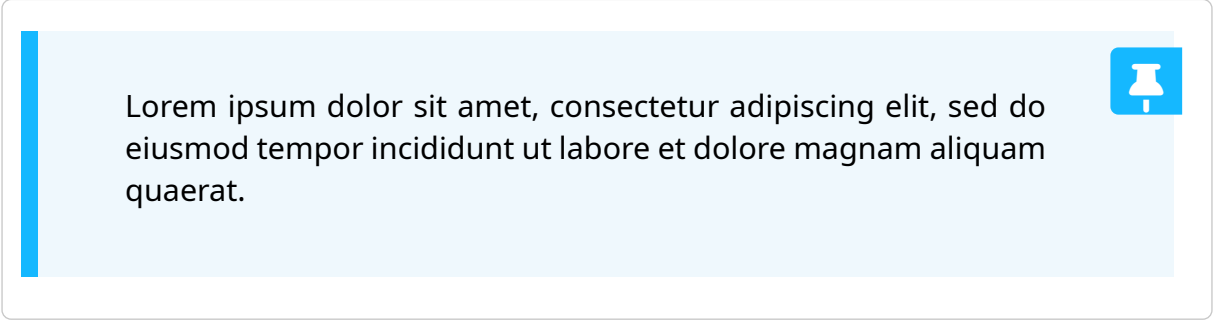
```
reminder(body: content, full-width) -> content
```

See <https://componentsforlearning.org/components/reminder/> for details and examples rendered in HTML.

This is how the component looks in PDF preview:

```
1 c4l.reminder[#lorem(20)]
```

typc

A PDF preview of the reminder component. It features a light blue rectangular box with a vertical blue bar on the left side. Inside the box, the text "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat." is displayed in a serif font. In the top right corner of the box, there is a small blue square icon containing a white pushpin symbol.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.

#### Parameters:

full-width (= false) – whether the component should take up the full text width


```
quote(body: content, full-width) -> content
```

See <https://componentsforlearning.org/components/quote/> for details and examples rendered in HTML.

This is how the component looks in PDF preview:

```
1 c4l.quote[#lorem(20)]
```

typc

A PDF preview of the quote component. It shows a light blue rectangular box with a vertical blue bar on the left side. Inside the box, the text "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat." is displayed in a serif font, enclosed in double quotation marks.

*"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat."*

#### Parameters:

full-width (= false) – whether the component should take up the full text width

```
do-dont(do: content, dont: content, full-width) -> content
```

See <https://componentsforlearning.org/components/do-dont-cards/> for details and examples rendered in HTML.

This is how the component looks in PDF preview:

```
1 c4l.do-dont[#lorem(20)][#lorem(20)]
```

typc

Lorem ipsum dolor sit amet, consectetur adipiscing elit,  
sed do eiusmod tempor incididunt ut labore et dolore  
magnam aliquam quaerat.



Lorem ipsum dolor sit amet, consectetur adipiscing elit,  
sed do eiusmod tempor incididunt ut labore et dolore  
magnam aliquam quaerat.



#### Parameters:

`full-width` (= `false`) – whether the component should take up the full text width

```
reading-context(body: content, full-width) -> content
```

See <https://componentsforlearning.org/components/reading-context/> for details and examples rendered in HTML.

This is how the component looks in PDF preview:

```
1 c4l.reading-context[#lorem(20)]
```

typc

Lorem ipsum dolor sit amet, consectetur adipiscing elit,  
sed do eiusmod tempor incididunt ut labore et dolore  
magnam aliquam quaerat.

#### Parameters:

`full-width` (= `false`) – whether the component should take up the full text width

```
example(title: content, body: content, full-width) -> content
```

See <https://componentsforlearning.org/components/example/> for details and examples rendered in HTML.

This is how the component looks in PDF preview:

```
1 c4l.example[Title][#lorem(20)]
```

typc

### TITLE

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.

#### Parameters:

`full-width` (= `false`) – whether the component should take up the full text width

```
figure(..args, full-width) -> content
```

See <https://componentsforlearning.org/components/figure/> for details and examples rendered in HTML.

This is how the component looks in PDF preview:

```
1 c4l.figure[#lorem(20)]
```

typc

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.

#### Parameters:

`full-width` (= `false`) – whether the component should take up the full text width

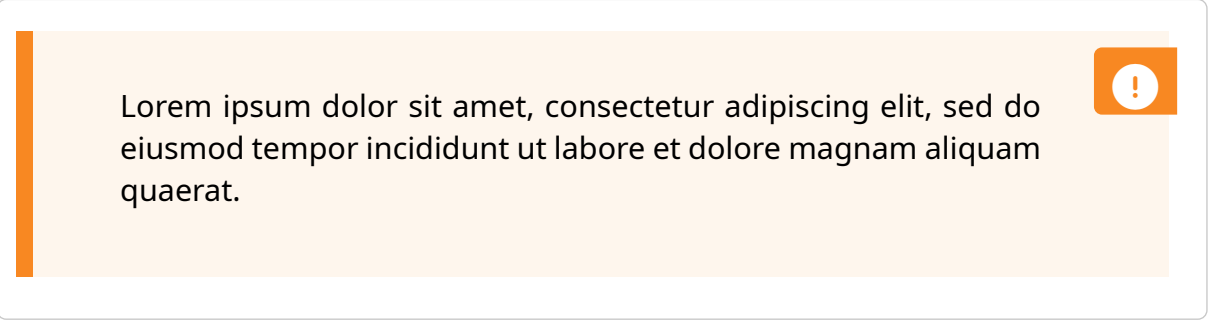
```
attention(body: content , full-width) -> content
```

See <https://componentsforlearning.org/components/attention/> for details and examples rendered in HTML.

This is how the component looks in PDF preview:

```
1 c4l.attention[#lorem(20)]
```

typc

The image shows a PDF preview of the 'attention' component. It features a light orange rectangular box with a vertical orange bar on the left side. Inside the box, the text 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.' is displayed in a black serif font. In the top right corner of the box, there is a small orange square icon containing a white exclamation mark.

#### Parameters:

full-width (= false) – whether the component should take up the full text width

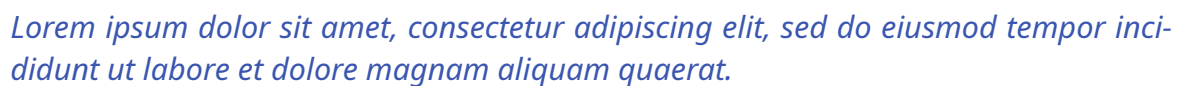
```
procedural-context(body: content , full-width) -> content
```

See <https://componentsforlearning.org/components/procedural-context/> for details and examples rendered in HTML.

This is how the component looks in PDF preview:

```
1 c4l.procedural-context[#lorem(20)]
```

typc

The image shows a PDF preview of the 'procedural-context' component. It displays the text 'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.' in a blue italicized serif font.

#### Parameters:

full-width (= false) – whether the component should take up the full text width

```
learning-outcomes(title: content , body: content , full-width) -> content
```

See <https://componentsforlearning.org/components/learning-outcomes/> for details and examples rendered in HTML.

This is how the component looks in PDF preview:

```
1 c4l.learning-outcomes[Title][#lorem(20)]
```

typc

#### TITLE

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua quaerat.

#### Parameters:

`full-width` (= `false`) – whether the component should take up the full text width

```
expected-feedback(body: content, full-width) -> content
```

See <https://componentsforlearning.org/components/expected-feedback/> for details and examples rendered in HTML.

This is how the component looks in PDF preview:

```
1 c4l.expected-feedback[#lorem(20)]
```

typc

*Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua quaerat.*



#### Parameters:

`full-width` (= `false`) – whether the component should take up the full text width

```
card(body: content, full-width) -> content
```

See <https://componentsforlearning.org/components/all-purpose-card/> for details and examples rendered in HTML.

This is how the component looks in PDF preview:

```
1 c4l.card[#lorem(20)]
```

typc



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua quaerat.

### Parameters:

`full-width` (= `false`) – whether the component should take up the full text width

## II.c htmlx

- `elem()`

- `div()`

- `img()`

```
elem(..args: any) -> content
```

A thin wrapper around Typst's `html.elem` that collects all named arguments into the `attrs` parameter. For example, these two are equivalent:

```
1 html.elem("div", attrs: (class: "x"))[body]
2 htmlx.elem("div", class: "x")[body]
```

typc

```
div() -> content
```

A thin wrapper for creating `div` html elements. In other words, these two are equivalent:

```
1 htmlx.elem("div", class: "x")[body]
2 htmlx.div(class: "x")[body]
```

typc

```
img(src, ..args) -> content
```

A thin wrapper for creating `img` html elements, making the required `src` attribute positional. In other words, these two are equivalent:

```
1 htmlx.elem("img", src: "example.png", alt: "Example")
2 htmlx.img("example.png", alt: "Example")
```

typc