# 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

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 or target PDF in addition to Moodle.

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

- Components for Learning (C4L) (Atto/TinyMCE plugin): Moodular recreates the HTML structure of C4L blocks, and also renders them for PDF export.

## I.a HTML preview and export

one of Moodular's main features is setting up your document for HTML-equivalent preview. This includes using a single unlimited-height page, using a sans serif font for text, and showing links in blue and underlined. Other aspects of this include the display and blockquotes and raw blocks, which are styled to match Moodle's Boost theme's default style.

To get started apply Moodular's settings like this:

```typ
#import "@preview/moodular:0.0.1" as moodular: c4l

#show: moodular.preview()
// or
#show: moodular.export()
```

`preview()` is the right choice if you only want HTML export, and use "regular" mode only to preview. If you want to instead target both HTML and PDF exports, `export()` is what you want: it won't change the page setup, but apply other styles. Both are just wrappers around `setup()` the exact behavior is documented there.

## I.b Components for Learning (C4L)

A huge part of this package is providing the Components for Learning (C4L) features. Most but not all components are currently supported:

**Contextual components**

- ✅ Do/Don't Cards
- ✅ Example
- ✅ Figure
- ❌ Inline Tag
- ✅ Key concept
- ✅ Quote
- ✅ Reading Context
- ✅ Reminder
- ❌ Tag
- ✅ Tip

**Procedural components**

- ✅ Attention
- ❌ Due Date
- ❌ Estimated time
- ✅ Learning Outcomes
- ✅ Procedural Context

**Evaluative components**

- ✅ Expected Feedback
- ❌ Grading Value

**Helper components**

- ✅ All-purpose card

The gallery has a document previewing some components, and you can klick on the implemented components above to go to their docs, including rendered examples. Please open an issue at https://github.com/SillyFreak/typst-moodular/issues if you need one of the missing components.

# II Module reference

## II.a `moodular`



`setup(mode: string) -> function`

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

```typ
#show: setup(mode: "preview")  // or "export"
```

If you don't need flexibility of the `mode`, you can use `preview()` or `export()` instead.

The function applies the following settings for PDF export:

- if `mode` is `"preview"`, 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 Boost theme'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 `<img src="@@PLUGINFILE@@/..." class="img-fluid">` tags. The `@@PLUGINFILE@@` prefix is used internally by Moodle to indicate references to uploaded files.

**Parameters:**

`mode` (`string` = none) – the mode to use for the document; either `preview` (simulate a non-paged run of HTML content) or `export` (don't change the page setup, as to produce a paged PDF document).

`preview() -> function`

Sets up the document for HTML preview, i.e. simulating a non-paged run of HTML content. Call this as a show rule:

```typ
#show: preview()
```

This is a wrapper around `setup()`, see that function for details.

`export() -> function`

Sets up the document for PDF export, i.e. producing a paged PDF document. Call this as a show rule:

```typ
#show: export()
```

This is a wrapper around `setup()`, see that function for details.

`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

- `remove-spacer()`
- `key-concept()`
- `tip()`
- `reminder()`
- `quote()`
- `do-dont()`
- `reading-context()`
- `example()`
- `figure()`
- `attention()`
- `procedural-context()`
- `learning-outcomes()`
- `expected-feedback()`
- `card()`

`remove-spacer(value: bool) -> content`

Enables or disables the removal of `<p class="c4l-spacer"></p>` elements from the output when exporting to HTML. By default, these elements are rendered to be consistent with the C4L editor moodle plugins (for Atto and TinyMCE), but this looks bad according to my testing on a clean Moodle install; see also https://github.com/reskit/moodle-tiny_c4l/issues/20.

The default of this setting can be overridden by compiling with `--input moodular-remove-spacer=true`.

**Parameters:**

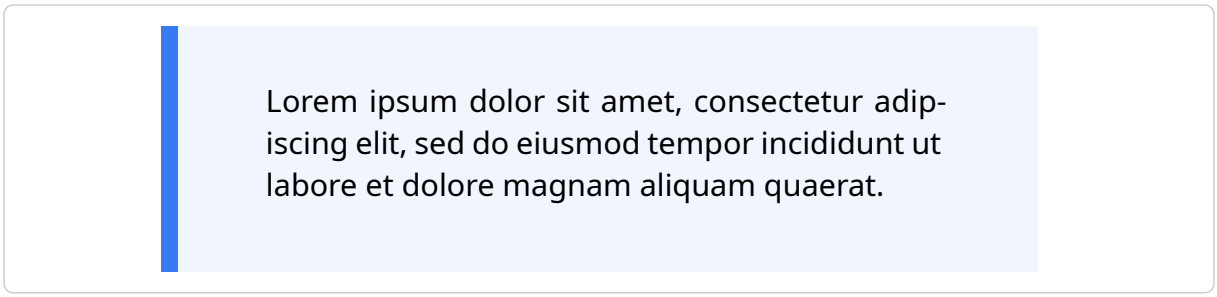
`value` (`bool`) – whether to remove spacer elements or not.

`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:

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




**Parameters:**

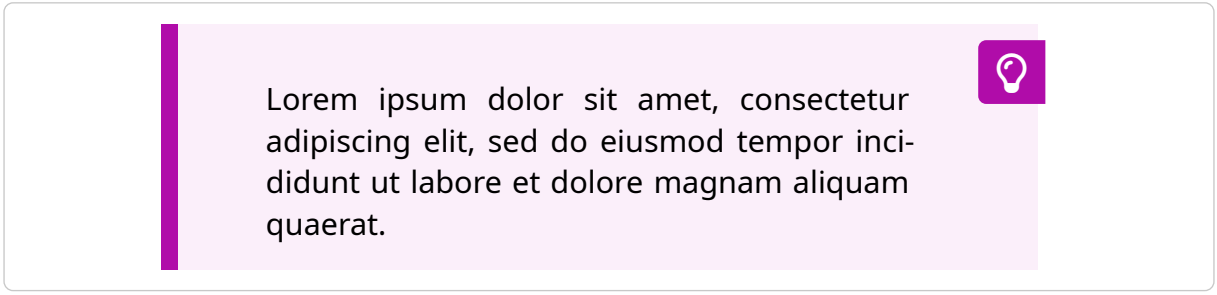
`full-width` ( = `false`) – whether the component should take up the full text width

`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
```




**Parameters:**

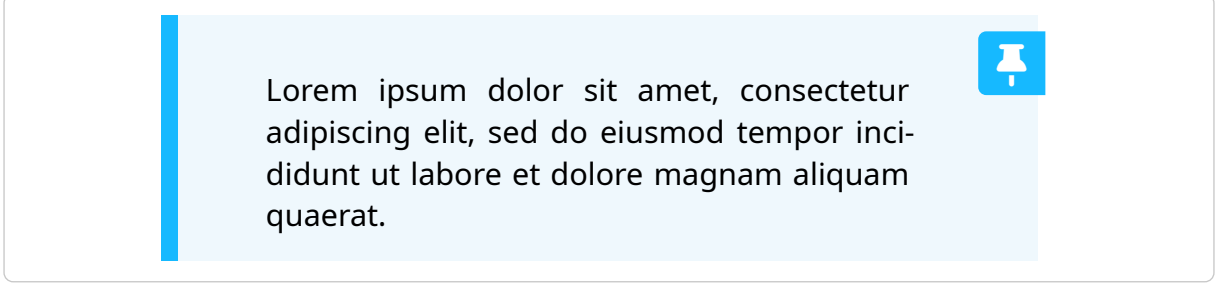
`full-width` ( = `false`) – whether the component should take up the full text width

`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
```



**Parameters:**

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

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

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

```
1 c4l.quote(attribution: lorem(5))[#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.

**Parameters:**

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

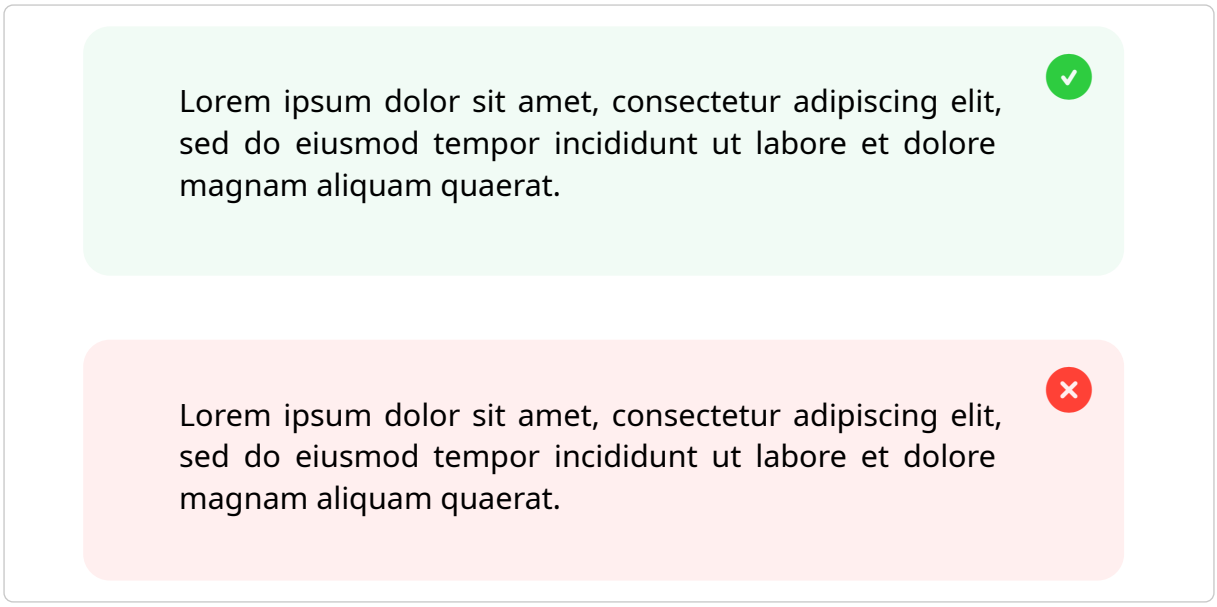
`attribution` ( = `none`) – the attribution associated with the quote

```
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




**Parameters:**

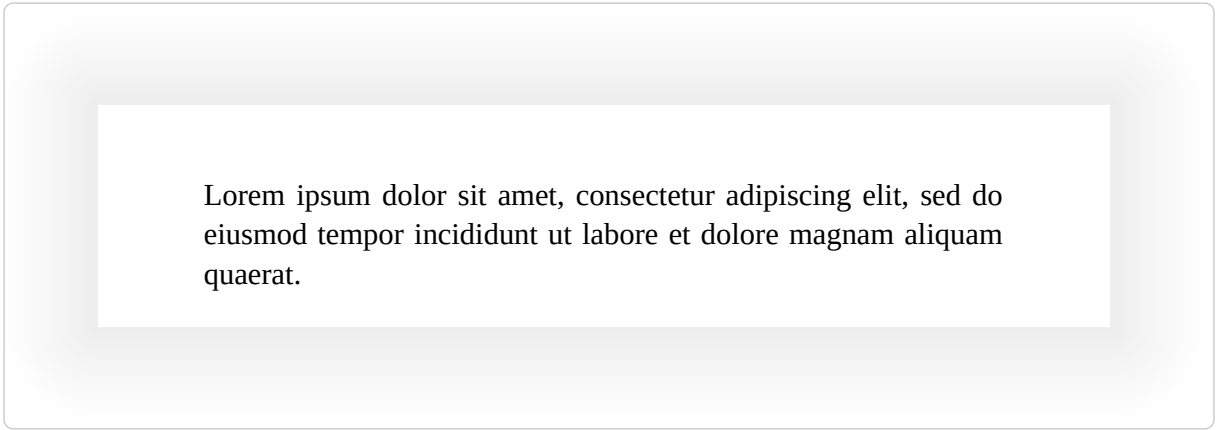
`full-width` ( = `false`) – whether the component should take up the full text width

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

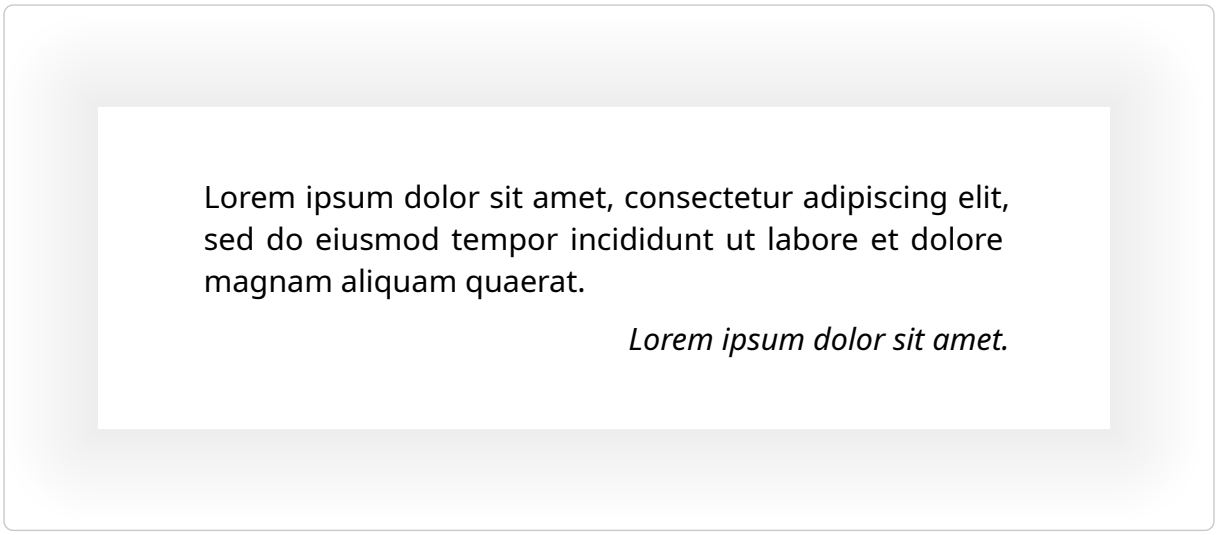
See https://componentsforlearning.org/components/reading-context/ for details and examples rendered in HTML.

This is how the component looks in PDF preview:

```
1  c4l.reading-context(comfort-reading: true)[#lorem(20)]                typc
```




```
1  c4l.reading-context(attribution: lorem(5))[#lorem(20)]                typc
```




**Parameters:**

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

`attribution` ( = `none`) – the attribution associated with the quote

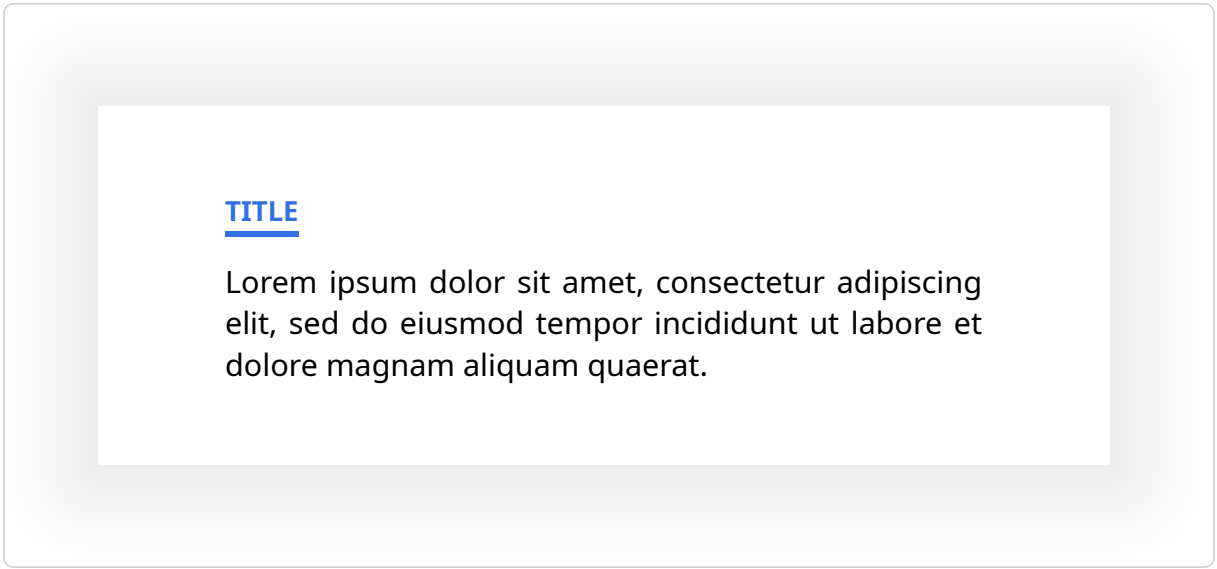
`comfort-reading` ( = `false`) – whether to display the context in a serif font

```
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
```




**Parameters:**

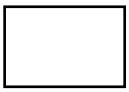
`full-width` ( = `false`) – whether the component should take up the full text width

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

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

This is rendered as a regular figure in PDF preview:

```
1 c4l.figure(rect())
```
typc



```
1 c4l.figure(rect(), caption: lorem(5))
```
typc

Lorem ipsum dolor sit amet.

**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

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

`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:

```
c4l.procedural-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

`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:

```
c4l.learning-outcomes[Title][
  - #lorem(5)
  - #lorem(10)
]
```
typc

TITLE

- Lorem ipsum dolor sit amet.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

```
c4l.learning-outcomes[Title][
  + #lorem(5)
  + #lorem(10)
]
```
typc

TITLE

1. Lorem ipsum dolor sit amet.
2. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

**Parameters:**

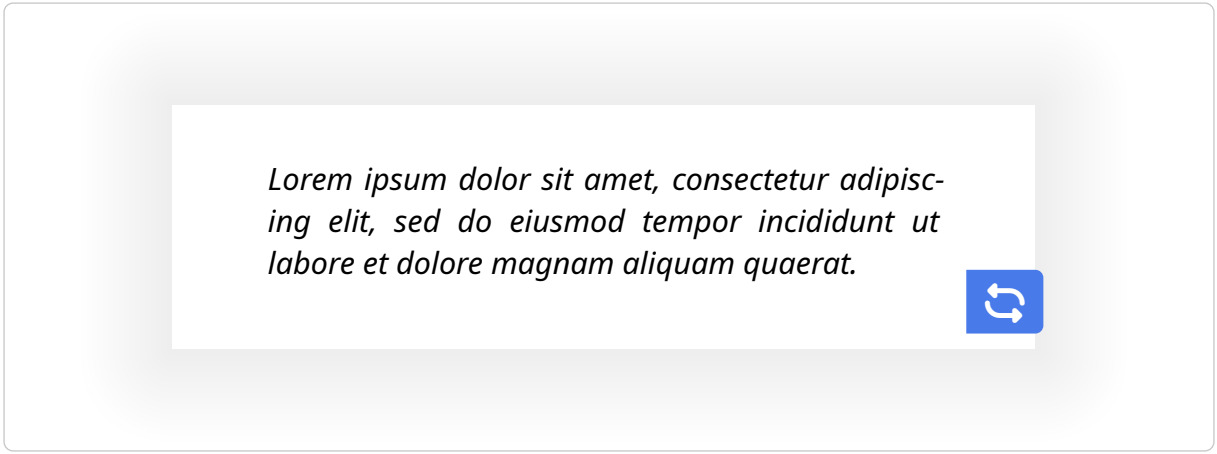
`full-width` ( = `false`) – whether the component should take up the full text width

### `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
```




**Parameters:**

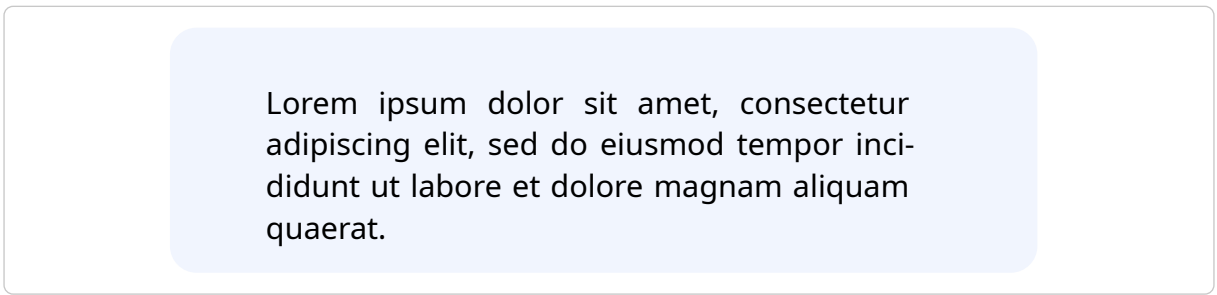
`full-width` ( = `false`) – whether the component should take up the full text width

### `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
```




**Parameters:**

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