

# Crudo

v0.1.0

May 14, 2024

<https://github.com/SillyFreak/typst-packages/tree/main/crudo>

**Clemens Koza**

## **ABSTRACT**

*Crudo* lets you take slices from raw blocks and more: slice, filter, transform and join the lines of raw blocks.

## **CONTENTS**

I Introduction .....	2
II Module reference .....	2

# I INTRODUCTION

raw elements feel similar to arrays and strings in a lot of ways: they feel like lists of lines; it's common to want to extract specific lines, join multiple ones together, etc. As values, though, raw elements don't behave this way.

While a package can't add methods such as `raw.slice()` to an element, we can at least provide functions to help with common tasks. The module reference describes these utility functions:

- `r2l()` and `l2r()` are the building blocks the others build on: *raw-to-lines* and *lines-to-raw* conversions.
- `transform()` is one layer above and allows arbitrarily transforming an array of strings.
- `map()`, `filter()` and `slice()` are analogous to their array counterparts.
- `lines()` is similar to `slice()` but allows more advanced line selections in a single step.
- `join()` combines multiple raw elements and is convenient e.g. to add preambles to code snippets.

## II MODULE REFERENCE

### II.a crudo

- `r2l()`
- `l2r()`
- `transform()`
- `map()`
- `filter()`
- `slice()`
- `lines()`
- `join()`

```
r2l(raw-block: content) -> array
```

*raw-to-lines*: extract lines and properties from a raw element.

<pre>1 crudo.r2l(``txt 2 first line 3 second line 4 ``)</pre>	<pre>(   ("first line", "second line"),   (block: true, lang: "txt"), )</pre>
---	---

Note that even though you will usually want to use this on raw *blocks*, this is not a necessity:

<pre>1 crudo.r2l( 2   raw("first line\nsecond line") 3 )</pre>	<pre>(("first line", "second line"), (:))</pre>
--	---

#### Parameters:

`raw-block (content)` – a single raw element

```
l2r(lines: array, ..properties: arguments) -> content
```

*lines-to-raw*: convert lines into a raw element. Properties for the created element can be passed as parameters.

<pre>1 crudo.l2r( 2   ("first line", "second line") 3 )</pre>	<pre>first line second line</pre>
---	---------------------------------------

Note that even though you will usually want to construct raw *blocks*, this is not assumed. To create blocks, pass the appropriate parameter:

<pre>1 crudo.l2r( 2   ("first line", "second line"), 3   block: true, 4 )</pre>	<pre>1 first line 2 second line</pre>
---	---

### Parameters:

`lines (array)` – an array of strings

`..properties (arguments)` – properties for constructing the new raw element

```
transform(raw-block: content, mapper: function) -> content
```

Transforms all lines of a raw element and creates a new one with the lines. All properties of the element (e.g. block and lang) are preserved.

<pre>1  crudo.transform( 2    ```typc 3    let foo() = { 4      // some comment 5      ... do something ... 6    } 7    ```, 8    lines =&gt; lines.filter(l =&gt; { 9      // only preserve non-comment lines 10     not l.starts-with(regex("\s*//")) 11   }) 12 )</pre>	<pre>1 let foo() = { 2   ... do something ... 3 }</pre>
--	---

### Parameters:

`raw-block (content)` – a single raw element

`mapper (function)` – a function that takes an array of strings and returns a new one

```
map(raw-block: content, mapper: function) -> content
```

Maps individual lines of a raw element and creates a new one with the lines. All properties of the element (e.g. block and lang) are preserved.

```

1 crudo.map(                                     typc
2   ```typc
3   let foo() = {
4     // some comment
5     ... do something ...
6   }
7   ``` ,
8   line => line.trim()
9 )

```

```

1 let foo() = {                                     typc
2   // some comment
3   ... do something ...
4 }

```

### Parameters:

`raw-block ( content )` – a single raw element

`mapper ( function )` – a function that takes a string and returns a new one

`filter(raw-block: content, test: function) -> content`

Filters lines of a raw element and creates a new one with the lines. All properties of the element (e.g. block and lang) are preserved.

```

1 crudo.filter(                                    typc
2   ```typc
3   let foo() = {
4     // some comment
5     ... do something ...
6   }
7   ``` ,
8   l => not l.starts-with(regex("\s*//"))
9 )

```

```

1 let foo() = {                                     typc
2   ... do something ...
3 }

```

### Parameters:

`raw-block ( content )` – a single raw element

`test ( function )` – a function that takes a string and returns a new one

`slice(raw-block: content, ..args: arguments) -> content`

Slices lines of a raw element and creates a new one with the lines. All properties of the element (e.g. block and lang) are preserved.

<pre> 1 crudo.slice( 2   ```typc 3   let foo() = { 4     // some comment 5     ... do something ... 6   } 7   ```, 8   1, 3, 9 ) </pre>	<pre> 1 // some comment 2 ... do something ... </pre>
---	---

### Parameters:

`raw-block ( content )` – a single raw element

`..args ( arguments )` – the same arguments as accepted by `array.slice()`

```
lines(raw-block: content, ..line-numbers: arguments, zero-based: boolean) -> content
```

Extracts lines of a raw element similar to how e.g. printers select page ranges. All properties of the element (e.g. block and lang) are preserved.

This function is comparable to `slice()` but doesn't have the option to specify the *number* of selected lines via count. On the other hand, multiple ranges of pages can be selected, and indices are one-based by default, which may be more natural for line numbers.

Lines are selected by any number of parameters. Each parameter can take either of three forms:

- a single number: that line is included in the output
- an array of numbers: these lines are included in the output (a major usecase being `range()` – but beware that `range()` uses an exclusive end index)
- a string containing numbers (e.g. 1) and inclusive ranges (e.g. 1-2) separated by commas. Whitespace is allowed.

All three kinds of parameters can be mixed, and lines can be selected any number of times and in any order.

<pre> 1 crudo.lines( 2   ```typc 3   let foo() = { 4     // some comment 5     ... do something ... 6   } 7   ```, 8   2, "1,3-4", range(2, 4), 9 ) </pre>	<pre> 1 // some comment 2 let foo() = { 3   ... do something ... 4 } 5 // some comment 6 ... do something ... </pre>
--	--

### Parameters:

`raw-block ( content )` – a single raw element

`..line-numbers ( arguments )` – any number of line number specifiers, as described above

`zero-based (boolean = false)` – whether the supplied numbers are one-based line numbers or zero-based indices

```
join(..raw-blocks: content, main: function) -> content
```

Joins lines of multiple raw elements and creates a new one with the lines. All properties of the main element (e.g. `block` and `lang`) are preserved.

<pre>1  crudo.join(<span style="border: 1px solid black; border-radius: 3px; padding: 0 2px;">typc</span> 2    ``java 3    let foo() = { 4      // some comment 5      ... do something ... 6    } 7    `` , 8    ``typc 9    let bar() = { 10     // some comment 11     ... do something ... 12   } 13   `` , 14   main: -1, 15 )</pre>	<pre>1  let foo() = {<span style="border: 1px solid black; border-radius: 3px; padding: 0 2px;">typc</span> 2    // some comment 3    ... do something ... 4  } 5  let bar() = { 6    // some comment 7    ... do something ... 8  }</pre>
---	--

### Parameters:

`..raw-blocks (content)` – any number of raw elements

`main (function = 0)` – the index of the raw element of which properties should be preserved.

Negative indices count from the back.