

Crudo

v0.1.1 September 28, 2024

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

All functions that accept raw elements as parameters alternatively accept simple strings. In these cases, a string code behaves like `raw(code)`, i.e. it's not a block element and has no `lang` set on it. This is mostly useful with `join()`, which takes multiple raw elements, but the other functions don't disallow this usage.

II MODULE REFERENCE

II.a crudo

- `r2l()`
- `l2r()`
- `transform()`

- `map()`
- `filter()`
- `slice()`

- `lines()`
- `join()`

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

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

```
1 crudo.r2l(``txt
2 first line
3 second line
4 ``)
```

typc

```
(
  ("first line", "second line"),
  (block: true, lang: "txt"),
)
```

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

```
1 crudo.r2l(
2   raw("first line\nsecond line")
3 )
```

typc

```
((("first line", "second line"), (:))
```

For flexibility, regular strings are also supported. Strings don't have a language and aren't blocks:

```
1 crudo.r2l("first line\nsecond line")
```

typc

```
((("first line", "second line"), (:))
```

Parameters:

`raw-block` (`content` or `str`) – a single raw element or (multi line) string

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

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

```
1 crudo.l2r(  
2   ("first line", "second line")  
3 )
```

typc

```
first line  
second line
```

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

```
1 crudo.l2r(  
2   ("first line", "second line"),  
3   block: true,  
4 )
```

typc

```
first line  
second line
```

Parameters:

`lines (array)` – an array of strings

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

```
transform(raw-block: content str, 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.

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

typc

```
let foo() = {  
  ... do something ...  
}
```

Parameters:

`raw-block (content or str)` – a single raw element or (multi line) string

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

```
map(raw-block: content str, 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(
2   ``typc
3   let foo() = {
4     // some comment
5     ... do something ...
6   }
7   ```,
8   l => l.trim()
9 )

```

```

let foo() = {
  // some comment
  ... do something ...
}

```

Parameters:

raw-block (content or str) – a single raw element or (multi line) string

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

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

```

```

let foo() = {
  ... do something ...
}

```

Parameters:

raw-block (content or str) – a single raw element or (multi line) string

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

```
slice(raw-block: content str, ..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.

```

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

```

```

// some comment
... do something ...

```

Parameters:

`raw-block (content or str)` – a single raw element or (multi line) string

`..args (arguments)` – the same arguments as accepted by [array.slice\(\)](#)

```
lines(raw-block: content str, ..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. "2-3") separated by commas. Range limits may be omitted (e.g. "-2", "2-"), meaning the range starts/ends at the first/last line. Whitespace is allowed.

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

```
1  crudo.lines(                                     typec
2    ``typec
3    let foo() = {
4      // some comment
5      ... do something ...
6      // another comment
7    }
8    `` ,
9    "-2,4-,1", "2-3", range(3, 5), 5,
10 )
```

```
let foo() = {
  // some comment
  // another comment
}
let foo() = {
  // some comment
  ... do something ...
  ... do something ...
  // another comment
}
```

Parameters:

`raw-block (content or str)` – a single raw element or (multi line) string

`..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: arguments, main: int auto) -> 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.

```

1  crudo.join(
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 )

```

```

let foo() = {
  // some comment
  ... do something ...
}
let bar() = {
  // some comment
  ... do something ...
}

```

String parameters are allowed; the main parameter defaults to the first raw block:

```

1  crudo.join(
2    "// these strings don't",
3    "// determine the properties",
4    ``typ
5    // this raw block does:
6    // still Typst!
7    `` ,
8  )

```

```

// these strings don't
// determine the properties
// this raw block does:
// still Typst!

```

Parameters:

`..raw-blocks (arguments)` – any number of single raw elements or (multi line) strings

`main (int or auto = auto)` – the index of the raw element of which properties should be preserved.

Negative indices count from the back. `auto` chooses the first positional argument that is a raw element and not a string, if any.