Crudo

v0.0.1

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

`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 spcific 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 arbitrarilyy 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(raw-block: `content`) -> `array`

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

```
crudo.r2l(```txt                                typc
first line
second line
```)
```

```
(
  ("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:

```
crudo.r2l(                                      typc
  raw("first line\nsecond line")
)
```

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

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.

```typc
crudo.l2r(
  ("first line", "second line")
)
```

```
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:

```typc
crudo.l2r(
  ("first line", "second line"),
  block: true,
)
```

```
first line
second line
```

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.

```typc
crudo.transform(
  ```typc
  let foo() = {
    // some comment
    ... do something ...
  }
  ```,
  lines => lines.filter(l => {
    // only preserve non-comment lines
    not l.starts-with(regex("\s*//"))
  })
)
```

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

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.

```typc
crudo.map(
  ```typc
  let foo() = {
    // some comment
    ... do something ...
  }
  ```,
  line => line.trim()
)
```

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

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.

```typc
crudo.filter(
  ```typc
  let foo() = {
    // some comment
    ... do something ...
  }
  ```,
  l => not l.starts-with(regex("\s*//"))
)
```

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

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.

```typc
crudo.slice(
  ```typc
  let foo() = {
    // some comment
    ... do something ...
  }
  ```,
  1, 3,
)
```

```typc
  // some comment
  ... do something ...
```

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 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.

```typc
crudo.lines(
  ```typc
  let foo() = {
    // some comment
    ... do something ...
  }
  ```,
  2, "1,3-4", range(2, 4),
)
```

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

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.

```typc
crudo.join(
  ```java
  let foo() = {
    // some comment
    ... do something ...
  }
  ```,
  ```typc
  let bar() = {
    // some comment
    ... do something ...
  }
  ```,
  main: -1,
)
```

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

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.