Typst-Easytable Package

version: 0.1.0

2024-02-21

Outline

1. Overview

`typst-easytable` is a simple package for writing tables in Typst.

1.1. Goal of `typst-easytable`

- Concise, highly visible markup
- Some degree of flexibility, versatility

1.2. Non-Goal of `typst-easytable`

- Features that are not needed for most applications

2. Usage

```
#import "@preview/easytable:0.1.0": easytable, elem
```

2.1. A Simple Table

Simple tables can be described simply.

```
#easytable({
  elem.td[How      ][I      ][want      ]
  elem.td[a        ][drink, ][alcoholic ]
  elem.td[of       ][course,][after     ]
  elem.td[the      ][heavy  ][lectures  ]
  elem.td[involving][quantum][mechanics.]
})
```

How	I	want
a	drink,	alcoholic
of	course,	after
the	heavy	lectures
involving	quantum	mechanics.

`elem.td` is a function representing data element in the `elem` module. To add a table header, use the `elem.th` function. It represents header element.

```
#easytable({
  elem.th[Header 1 ][Header 2][Header 3  ]
  elem.td[How      ][I       ][want      ]
  elem.td[a        ][drink,  ][alcoholic ]
  elem.td[of       ][course, ][after     ]
  elem.td[the      ][heavy   ][lectures  ]
  elem.td[involving][quantum ][mechanics.]
})
```

Header 1	Header 2	Header 3
How	I	want
a	drink,	alcoholic
of	course,	after
the	heavy	lectures
involving	quantum	mechanics.

If you feel tedious to write `elem.*`, you can omit it by writing as follows:

```
// If you don't care having functions such as `th` and `td` in global namespace,
// it is easiest to write the import statement here!
#import elem: *

#easytable({
  // If you care, please state the following for each table.
  // import elem: *
  td[How      ][I      ][want      ]
  td[a        ][drink, ][alcoholic ]
  td[of       ][course,][after     ]
  td[the      ][heavy  ][lectures  ]
  td[involving][quantum][mechanics.]
})
```

We will omit `import elem: *` in examples hereafter.

The argument of `elem.td` is variadic, i.e., you can easily create tables with any number of columns.

```
#easytable({
  vline(x: 1, stroke: 0.5pt)
  cstyle(..(center,) * 7)

  th[$*$][$e$][$r$][$r^2$][$s$][$r s$][$r^2s$]
  td[$e$][$e$][$r$][$r^2$][$s$][$r s$][$r^2s$]
  td[$r$][$r$][$r^2$][$e$][$r s$][$r^2s$][$s$]
  td[$r^2$][$r^2$][$e$][$r$][$r^2s$][$s$][$r s$]
  td[$s$][$s$][$r^2s$][$r s$][$e$][$r^2$][$r$]
  td[$r s$][$r s$][$s$][$r^2s$][$r$][$e$][$r^2$]
  td[$r^2s$][$r^2s$][$r s$][$s$][$r^2$][$r$][$e$]
})
```

$*$	e	r	r^2	s	rs	r^2s
e	e	r	r^2	s	rs	r^2s
r	r	r^2	e	rs	r^2s	s
r^2	r^2	e	r	r^2s	s	rs
s	s	r^2s	rs	e	r^2	r
rs	rs	s	r^2s	r	e	r^2
r^2s	r^2s	rs	s	r^2	r	e

Please be careful, the number of arguments to `th` and `td` (and `cstyle` and `cwidth`, described below) elements must be consistent within a table. If not, the Typst processor throws an error. Conversely, there is no need to worry about forgetting to put a cell and not noticing that the layout is broken.

2.2. Changing Alignment and Width of Columns

To change the alignment for each column, use `cstyle`:

```
#easytable({
  cstyle(left, center, right)
  th[Header 1 ][Header 2][Header 3  ]
  td[How      ][I       ][want      ]
  td[a        ][drink,  ][alcoholic ]
  td[of       ][course, ][after     ]
  td[the      ][heavy   ][lectures  ]
  td[involving][quantum ][mechanics.]
})
```

Header 1	Header 2	Header 3
How	I	want
a	drink,	alcoholic
of	course,	after
the	heavy	lectures
involving	quantum	mechanics.

What if I want to change the length of each column? Use `cwidth`:

```
#easytable({
  cwidth(100pt, 1fr, 20%)
  th[Header 1 ][Header 2][Header 3  ]
  td[How      ][I       ][want      ]
  td[a        ][drink,  ][alcoholic ]
  td[of       ][course, ][after     ]
  td[the      ][heavy   ][lectures  ]
  td[involving][quantum ][mechanics.]
})
```

Header 1	Header 2	Header 3
How	I	want
a	drink,	alcoholic
of	course,	after
the	heavy	lectures
involving	quantum	mechanics.

It is of course possible to use `cstyle` and `cwidth` in combination.

```
#easytable({
  cwidth(100pt, 1fr, 20%)
  cstyle(left, center, right)
  th[Header 1 ][Header 2][Header 3  ]
  td[How      ][I       ][want      ]
  td[a        ][drink,  ][alcoholic ]
  td[of       ][course, ][after     ]
  td[the      ][heavy   ][lectures  ]
  td[involving][quantum ][mechanics.]
})
```

Header 1	Header 2	Header 3
How	I	want
a	drink,	alcoholic
of	course,	after
the	heavy	lectures
involving	quantum	mechanics.

It is also possible to write long content that spans multiple lines.

```
#easytable({
  cwidth(auto, 50%)
  cstyle(right, left)
  th[Term][Long Description]
  td[*LaTeX*][A great typesetting system. May be difficult to learn.]
  td([*Typst*], [
    A great typesetting system! Specifically, it offers the following advantages:

    - Very easy to install
    - Very easy to learn

    We encourage everyone to use it.
  ])
})
```

Term	Long Description
LaTeX	A great typesetting system. May be difficult to learn.
Typst	A great typesetting system! Specifically, it offers the following advantages: • Very easy to install • very easy to learn We encourage everyone to use it.

2.3. Customizing Element

Element `td` has an keyword argument `trans`, which can be used to customize the layout of a particular line.

```
#easytable({
  let td2 = td.with(trans: emph)
  let td3 = td.with(
    trans: (c) => box(fill: blue, inset: 3pt,
text(size: 0.8em, fill: white, c)),
  )

  th[Header 1][Header 2][Header 3]
  td[How][I][want]
  td2[a][drink,][alcoholic]
  td[of][course,][after]
  td3[the][heavy][lectures]
  td[involving][quantum][mechanics.]
})
```

Header 1	Header 2	Header 3
How	I	want
a	*drink,*	*alcoholic*
of	course,	after
the	heavy	lectures
involving	quantum	mechanics.

If you want to assign a common layout to all rows, you can override the definition of `td` itself locally.

```
#easytable(
  {
    let td = td.with(trans: pad.with(x: 3pt))

    th[Header 1][Header 2][Header 3]
    td[How][I][want]
    td[a][drink,][alcoholic]
    td[of][course,][after]
    td[the][heavy][lectures]
    td[involving][quantum][mechanics.]
  },
)
```

Header 1	Header 2	Header 3
How	I	want
a	drink,	alcoholic
of	course,	after
the	heavy	lectures
involving	quantum	mechanics.

Use the `cell_style` argument to change the background color.

```
#easytable({
  let th = th.with(trans: emph)
  let td = td.with(
    cell_style: (x: none, y: none)
      => (fill: if calc.even(y) {
        luma(95%)
      } else {
        none
      })
  )

  th[Header 1][Header 2][Header 3]
  td[How][I][want]
  td[a][drink,][alcoholic]
  td[of][course,][after]
  td[the][heavy][lectures]
  td[involving][quantum][mechanics.]
})
```

Header 1	*Header 2*	*Header 3*
How	I	want
a	drink,	alcoholic
of	course,	after
the	heavy	lectures
involving	quantum	mechanics.

With `hline` you can draw a horizontal line at any position. The same goes for `vline`.

```
#easytable({
  th[Header 1][Header 2][Header 3]
  td[How][I][want]
  hline(stroke: red)
  td[a][drink,][alcoholic]
  td[of][course,][after]
  td[the][heavy][lectures]
  td[involving][quantum][mechanics.]

  // Specifying the insertion point directly
  hline(stroke: 2pt + green, y: 4)
  vline(
    stroke: (paint: blue, thickness: 1pt, dash:
"dashed"),
    x: 2,
    start: 1,
    end: 5,
  )
})
```

Header 1	Header 2	Header 3
How	I	want
a	drink,	alcoholic
of	course,	after
the	heavy	lectures
involving	quantum	mechanics.

2.4. Customizing Column Style

Use `cstyle`. The `cstyle` function accepts a function as well as an alignment as its argument.

```
#easytable({
  let show_tag(c) = box(
    fill: red.darken(60%),
    inset: (x: 2pt),
    outset: (y: 2pt),
    text(fill: white, size: 0.8em, c),
  )

  cstyle(left, center, left)
  th[Header 1][Header 2][Header 3]

  // Change style from the middle of the table
  cstyle(left, center, show_tag)
  td[How][I][want]
  td[a][drink,][alcoholic]
  td[of][course,][after]
  td[the][heavy][lectures]
  td[involving][quantum][mechanics.]
})
```

Header 1	Header 2	Header 3
How	I	want
a	drink,	alcoholic
of	course,	after
the	heavy	lectures
involving	quantum	mechanics.