mannot v0.1.0

A package for highlighting and annotating in math blocks in Typst.

Contents

Usage 1
Limitations 2
API 2

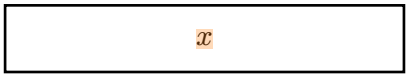
Usage

Import and initialize the package `mannot` on the top of your document.

```
#import "@preview/mannot:0.1.0": *
#show: mannot-init
```

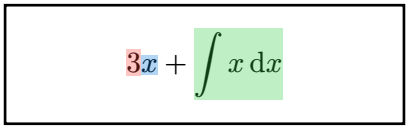
To highlight a part of a math block, use the `mark` function:

```
$
mark(x)
$
```



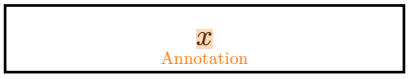
You can also specify a color for the highlighted part:

```
$ // Need # before color names.
mark(3, color: #red) mark(x, color: #blue)
+ mark(integral x dif x, color: #green)
$
```



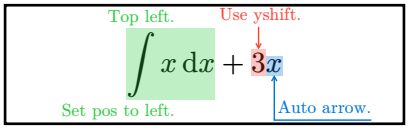
To add an annotation to a highlighted part, use the `annot` function. You need to specify the tag of the marked content:

```
$
mark(x, tag: #<x>)  // Need # before tags.
#annot(<x>)[Annotation]
$
```



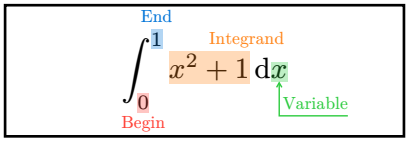
You can customize the position of the annotation and the vertical distance from the marked content:

```
$
mark(integral x dif x, tag: #<i>, color: #green)
+ mark(3, tag: #<3>, color: #red) mark(x, tag: #<x>, color: #blue)

#annot(<i>, pos: left)[Set pos to left.]
#annot(<i>, pos: top + left)[Top left.]
#annot(<3>, pos: top, yshift: 1.2em)[Use yshift.]
#annot(<x>, pos: right, yshift: 1.2em)[Auto arrow.]
$
```



For convenience, you can define custom mark functions:

```
#let rmark = mark.with(color: red)
#let gmark = mark.with(color: green)
#let bmark = mark.with(color: blue)

$
integral_rmark(0, tag: #<i0>)^bmark(1, tag: #<i1>)
```



```
  mark(x^2 + 1, tag: #<i2>) dif gmark(x, tag: #<i3>)

  #annot(<i0>)[Begin]
  #annot(<i1>, pos: top)[End]
  #annot(<i2>, pos: top + right)[Integrand]
  #annot(<i3>, pos: right, yshift: .6em)[Variable]
  $
```

Limitations

If you mark a inline math element containing linebreaks, its layout will be broken:

```

  $mark(x + x + x + x + x + x + x + x)$

```

$$x + x + x + x + x + x + x + x$$

API

- core-mark()
- mark()
- mannot-init()
- core-annot()
- annot()

core-mark

Marks a part of a math block with a custom overlay.

This function measures the position and size of the marked content, places an overlay on it, and exposes metadata labeled with a tag. This metadata includes the original content, its position (x, y), size (width, height), and the color used for the overlay. Its purpose is to create custom marking functions.

Example

```
  let mymark(body, tag: none) = {
    let overlay(width, height, color) = {
      rect(width: width, height: height, stroke: color)
    }
    return core-mark(body, tag: tag, color: red, overlay: overlay, padding: (y: .1em))
  }

  $ mymark(x, tag: #<e>) $

  context {
    let info = query(<e>).last()
    repr(info.value)
  }
```

x

```
(
  body: [x],
  x: 444.68pt,
  y: 447.73pt,
  width: 6.29pt,
  height: 7.18pt,
  color: rgb("#ff851b"),
)
```

Parameters

```
core-mark(
  body: content,
  tag: none label,
  color: color,
  overlay: none function,
  padding: none length dictionary,
  ctx: auto string
) -> content
```

body `content`

The content to be marked within a math block.

tag `none` or `label`

An optional tag associated with the metadata.

Default: `none`

color `color`

The color for the overlay and annotations put to this.

Default: `black`

overlay `none` or `function`

An optional function to create a custom overlay. The function takes `width`, `height`, and `color` as arguments, where `width` and `height` represent the size of the marked content including padding, and `color` is the same color passed to core-mark.

Default: `none`

padding `none` or `length` or `dictionary`

The space between the marked content and the border of the overlay. You can specify `left`, `right`, `top`, `bottom`, `x`, `y`, or a `rest` value.

Default: `(:)`

ctx `auto` or `string`

The context of the marked content.

Default: `auto`

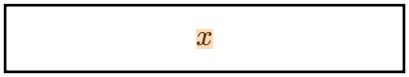
mark

Marks a part of a math block with highlighting.

Marked content can be annotated with `annot` function.

Example

```
1 $ mark(x) $
```



Parameters

```
mark(
  body: content,
  tag: none label,
  color: auto color,
  fill: auto none color gradient pattern,
  stroke: none auto length color gradient stroke pattern dictionary,
  radius: relative dictionary,
  padding: none length dictionary,
  ctx
) -> content
```

body `content`

The content to be highlighted within a math block.

tag `none` or `label`

An optional tag used to identify the highlighted content for later annotation.

Default: `none`

color `auto` or `color`

The color used for the highlight. If set to `auto` and `fill` is `auto`, `color` will set be set to `orange`. Otherwise, it defaults to `black`.

Default: `auto`

fill `auto` or `none` or `color` or `gradient` or `pattern`

The fill style for the highlight rectangle. If set to `auto`, `fill` will be set to `color.transparentize(70%)`.

Default: `auto`

stroke `none` or `auto` or `length` or `color` or `gradient` or `stroke` or `pattern` or `dictionary`

The stroke style for the highlight rectangle.

Default: `none`

radius `relative` or `dictionary`

The corner radius of the highlight rectangle.

Default: `(:)`

padding `none` or `length` or `dictionary`

he space between the highlighted content and the border of the highlight. You can specify `left`, `right`, `top`, `bottom`, `x`, `y`, or a `rest` value.

Default: `(y: .1em)`

mannot-init

Setup function.

To start using mannot, you need to initialize mannot using a show rule:

```
#show: mannot-init
```

This removes unintentional spaces around marked elements within math blocks.

Parameters

`mannot-init(body)`

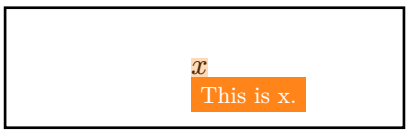
core-annot

Places a custom annotation to the marked content.

This function creates a custom annotation by placing an overlay on the content that was previously marked with a specific tag. It must be used within the same math block that contains the marked content.

Example

```
let myannot(tag, annotation) = {
  let overlay(width, height, color) = {
    set text(white, .8em)
    let annot-block = box(fill: color, inset: .4em, annotation)
    place(annot-block, dy: height)
  }
  return core-annot(tag, overlay)
}

$
mark(x, tag: #<e>)
#myannot(<e>)[This is x.]
$
```



Parameters

```
core-annot(
  tag: label,
  overlay: function
)
```

tag `label`

The tag associated with the content you want to annotate. This tag must match a previously used tag in a `core-mark` call.

overlay `function`

The function to create a custom annotation. It takes `width`, `height`, and `color` as arguments, where `width` and `height` represent the size of the marked content, and `color` is the base color passed to `core-mark`.

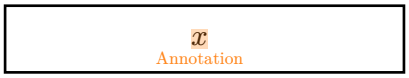
annot

Places an annotation to the marked content.

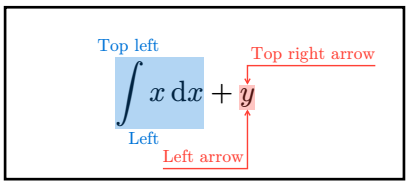
This function must be used within the same math block that contains the marked content.

Example

```
$
mark(x, tag: #<e>)
#annot(<e>)[Annotation]
$
```




```
$
mark(integral x dif x, tag: #<x>, color: #blue)
+ mark(y, tag: #<y>, color: #red)

#annot(<x>, pos: left)[Left]
#annot(<x>, pos: top + left)[Top left]
#annot(<y>, pos: left, yshift: 2em)[Left arrow]
#annot(<y>, pos: top + right, yshift: 1em)[Top right arrow]
$
```



Parameters

```
annot(
  tag: label,
  annotation: content,
  pos: alignment,
  yshift: length,
  text-props: dictionary,
  par-props: dictionary,
  alignment: auto alignment,
  show-arrow: auto bool,
  arrow-stroke: auto none color length dictionary stroke,
  arrow-padding: length
)
```

tag `label`

The tag associated with the content you want to annotate. This tag must match a previously used tag in a `core-mark` call.

annotation `content`

The content of the annotation.

pos `alignment`

The position of the annotation relative to the marked content. Possible values are (`top` or `bottom`) + (`left`, `center` or `right`).

Default: `center + bottom`

yshift `length`

The vertical distance between the annotation and the marked content.

Default: `.2em`

text-props `dictionary`

Properties for the annotation text. If the `fill` field is not specified, it defaults to the color used in the marking.

Default: `(size: .6em, bottom-edge: "descender")`

par-props `dictionary`

Properties for the annotation paragraph.

Default: `(leading: .3em)`

alignment `auto` or `alignment`

The alignment of the annotation text within its box.

Default: `auto`

show-arrow `auto` or `bool`

Whether to display an arrow connecting the annotation to the marked content. If set to `auto`, an arrow will be shown when `yshift > .4em`.

Default: `auto`

arrow-stroke `auto` or `none` or `color` or `length` or `dictionary` or `stroke`

The stroke style for the arrow. If the `paint` field is not specified, it defaults to the color used in the marking.

Default: `.08em`

arrow-padding `length`

The space between the arrow and the annotation box.

Default: `.2em`