Margin Notes

Setup

Unfortunately `typst` doesn't expose margins to calling functions, so you'll need to set them explicitly. This is done using `set-page-properties` **before you place any content**:

```
// At the top of your source file
// Of course, you can substitute any margin numbers you prefer
// provided the page margins match what you pass to `set-page-properties`
#import "../drafting.typ": *
#let (l-margin, r-margin) = (1in, 2in)
#set page(
  margin: (left: l-margin, right: r-margin, rest: 0.1in),
  paper: "us-letter",
)
#set-page-properties(margin-left: l-margin, margin-right: r-margin)
```

The basics

```
#lorem(20)
#margin-note(side: left)[Hello, world!]
#lorem(10)
#margin-note[Hello from the other side]

#lorem(25)
#margin-note[When notes are about to overlap, they're automatically shifted]
#margin-note(stroke: aqua + 3pt)[To avoid collision]
#lorem(25)

#let caution-rect = rect.with(inset: 1em, radius: 0.5em, fill: orange.lighten(80%))
#inline-note(rect: caution-rect)[
  Be aware that notes will stop automatically avoiding collisions if 4 or more notes
  overlap. This is because `typst` warns when the layout doesn't resolve after 5 attempts
  (initial layout + adjustment for each note)
]
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

Hello, world!

Hello from the other side

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus.

When notes are about to overlap, they're automatically shifted

To avoid collision

Be aware that notes will stop automatically avoiding collisions if 4 or more notes overlap. This is because `typst` warns when the layout doesn't resolve after 5 attempts (initial layout + adjustment for each note)

Adjusting the default style

All function defaults are customizable through updating the module state:

```
#lorem(4) #margin-note(dy: -2em)[Default style]
#set-margin-note-defaults(stroke: orange, side: left)
#lorem(4) #margin-note[Updated style]
```

Lorem ipsum dolor sit. Lorem ipsum dolor sit.

Default style

Updated style

Even deeper customization is possible by overriding the default `rect`:

```
#import "@preview/colorful-boxes:1.1.0": stickybox

#let default-rect(stroke: none, fill: none, width: 0pt, content) = {
  stickybox(rotation: 30deg, width: width/1.5, content)
}
#set-margin-note-defaults(rect: default-rect, stroke: none, side: right)

#lorem(20)
#margin-note(dy: -25pt)[Why not use sticky notes in the margin?]

// Undo changes from this example
#set-margin-note-defaults(rect: rect, stroke: red)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.

Why not use sticky notes in the margin?

Multiple document reviewers

```
#let reviewer-a = margin-note.with(stroke: blue)
#let reviewer-b = margin-note.with(stroke: purple)
#lorem(20)
#reviewer-a[Comment from reviewer A]
#lorem(15)
#reviewer-b(side: left)[Comment from reviewer B]
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.

Comment from reviewer A

Comment from reviewer B

Inline Notes

```
#lorem(10)
#inline-note[The default inline note will split the paragraph at its location]
#lorem(10)
#inline-note(par-break: false, stroke: (paint: orange, dash: "dashed"))[
  But you can specify `par-break: false` to prevent this
]
#lorem(10)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

The default inline note will split the paragraph at its location

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. But you can specify `par-break: false` to prevent this Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

Hiding notes for print preview

```
#set-margin-note-defaults(hidden: true)

#lorem(20)
#margin-note[This will respect the global "hidden" state]
#margin-note(hidden: false, dy: -2.5em)[This note will never be hidden]
// Undo these changes
#set-margin-note-defaults(hidden: false)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.

This note will never be hidden

Positioning

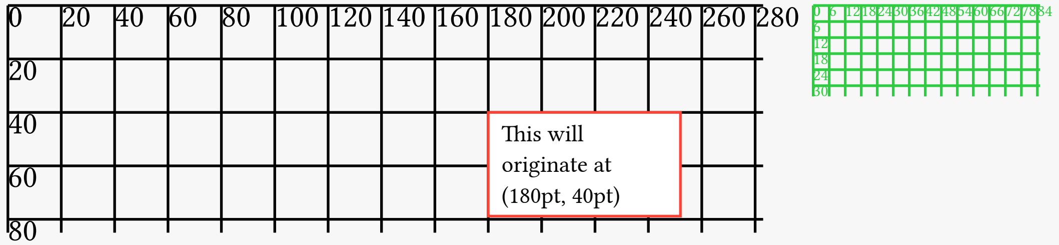
Precise placement: rule grid

Need to measure space for fine-tuned positioning? You can use `rule-grid` to cross-hatch the page with rule lines:

```
#rule-grid(width: 10cm, height: 3cm, spacing: 20pt)
#place(
  dx: 180pt,
  dy: 40pt,
  rect(fill: white, stroke: red, width: 1in, "This will originate at (180pt, 40pt)")
)

// Optionally specify divisions of the smallest dimension to automatically calculate
// spacing
#rule-grid(dx: 10cm + 3em, width: 3cm, height: 1.2cm, divisions: 5, square: true,  stroke: green)

// The rule grid doesn't take up space, so add it explicitly
#v(3cm + 1em)
```



Absolute positioning

What about absolutely positioning something regardless of margin and relative location? `absolute-place` is your friend. You can put content anywhere:

```
#locate(loc => {
  let (dx, dy) = (25%, loc.position().y)
  let content-str = (
    "This absolutely-placed box will originate at (" + repr(dx) + ", " + repr(dy) + ")"
    + " in page coordinates"
  )
  absolute-place(
    dx: dx, dy: dy,
    rect(
      fill: green.lighten(60%),
      radius: 0.5em,
      width: 2.5in,
      height: 0.5in,
      [#align(center + horizon, content-str)]
    )
  )
})
#v(0.5in)
```

This absolutely-placed box will originate at (25%, 643.16pt) in page coordinates

The “rule-grid” also supports absolute placement at the top-left of the page by passing `relative: false`. This is helpful for “rule”-ing the whole page.