

Marginalia

Write into the margins!

1 Setup

Put something akin to the following at the start of your .typ file:

```
#import "@preview/marginalia:0.2.1" as marginalia: note, notefigure, wideblock

#show: marginalia.setup.with(
  // inner: ( far: 5mm, width: 15mm, sep: 5mm ),
  // outer: ( far: 5mm, width: 15mm, sep: 5mm ),
  // top: 2.5cm,
  // bottom: 2.5cm,
  // book: false,
  // clearance: 12pt,
)
```

Where you can then customize these options to your preferences. Shown here (as comments) are the default values taken if the corresponding keys are unset.●

If `book` is `false`, inner and outer correspond to the left and right margins respectively. If `book` is `true`, the margins swap sides on even and odd pages. Notes are placed in the outside margin by default.

See the appendix for a more detailed explanation of the `setup()` function and its options.

Additionally, I recommend using `typst`'s partial function application feature to customize other aspects of the notes consistently:

```
#let note = note.with(/* options here */)
#let notefigure = notefigure.with(/* same options here */)
```

2 Margin-Notes

By default, the `#note[]` command places a note to the right/outer margin, like so:○. By giving the argument `side: "inner"`, we obtain a note on the inner (left) margin.◆ If `setup.book` is `true`, the side will of course be adjusted automatically. It is also possible to pass `side: "left"` or `side: "right"` if you want a fixed side even in books.

If ◆ we ■ place □ multiple ▲ notes ▲ in ♥ one ♥ line,● they automatically adjust their positions. Additionally, a `dy` argument can be passed to shift their initial position by that amount vertically. They may still get shifted around, unless configured otherwise via the `shift` parameter of `#note[]`.

Contents

1	Setup	1
2	Margin-Notes	1
2.1	Markers	2
2.2	Styling	2
3	Wide Blocks	3
4	Figures	3
4.1	Notefigures	3
4.2	Large Figures	4
5	Other Tidbits	5
5.1	Absolute Placement	5
5.2	Headers and Background	5
6	Troubleshooting / Known Bugs	7
7	Thanks	7
A	Detailed Documentation of all Exported Symbols	8

- You can also skip the configuration step if you're happy with these defaults, but 15mm is not a lot to write in.

- This is a note.

They can contain any content, and will wrap within the note column.

- ◆ Note 1

- Note 2

- Note 3

- ▲ Note 4

- △ Note 5

- ♥ This note was given 15pt dy, but it was shifted more than that to avoid Notes 1–5.

- This note was given 10cm dy and was shifted less than that to stay on the page.

◆ Reversed.

♥ This note was given 15pt dy.

- ◆ Note from second column.
 - Lorem ipsum dolor sit.
 - ◇ Like so. The lorem-ipsum note was also placed with keep-order.
 - This note has flush numbering.
 - Unnumbered notes "avoid" being shifted if possible, preferring to shift other notes up.
- Like this.

17 The `-2pt` in the `#h()` is there because `#note()` inserts a `2pt` space.

Like this one.

18 (the `notecounter` is unaffected by the previous note, as it has `numbering: none`)

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

Notes will shift vertically to avoid other notes, wideblocks, and the top page margin. It will attempt to move one note below a wide-block if there is not enough space above, but if there are multiple notes that would need to be rearranged you must assist by manually setting `dy` such that their initial position is below the wideblock.

Margin notes also fault, they are placed order as they appear work from within aligned with their an- in the text, set `keep-` most containers such chor.◆ To force the `order: true`◆ on *all* as `#block[]s` or notes to appear in the notes whose relative `#column[]s`.○ By de- margin in the same order is important.

2.1 Markers

The margin notes are decorated with little symbols, which by default hang into the gap. If this is not desired, set `flush-numbering: true` on the note.■

Setting the argument `numbering: none`,□ we obtain notes without icon/number:

To change the markers, you can override the `numbering` function which is used to generate the markers.

Advanced Markers

If a different style is desired for the marker in the text and in the margins, you can use the `anchor-numbering` parameter to control the in-text marker:¹⁷

```
#note(
  numbering: (.., i) => text(font: "Inria Sans")[#i#h(0.5em - 2pt)],
  anchor-numbering: (.., i) => super[#i],
)[...]
```

Note that doing this implies `flush-numbering: true`. This is based on the assumption that if you have set two different numbering functions, you want to handle the placement yourself. Non-flush numbers, which are placed, complicate this.

This can also be used to create notes that have an anchor,↵ but no numbering in the note itself.¹⁸

2.2 Styling

Both `note()` and `notefigure()` accept `text-style`, `par-style`, and `block-style` parameters:

- `text-style: (size: 5pt, font: ("Iosevka Extended"))` gives ♥
- `par-style: (spacing: 20pt, leading: -2pt)` gives ♥

The default options here are meant to be as close as possible to the stock footnote style given 11pt text. For other text sizes, set the `text-style` size to 0.85 times your body text size if you want to match the stock footnotes.

block-style

To style the block containing the note body, use the `block-style` parameter.

- `block-style: (stroke: (top: (thickness: 0.5pt, dash: "dotted")), outset: (top: 6pt /* clearance is 12pt */), width: 100%)` gives:●
 - `block-style: (fill: oklch(90%, 0.06, 140deg), outset: (left: 10pt, rest: 4pt), width: 100%, radius: 4pt)` gives:○
 - `block-style: (fill: oklch(90%, 0.06, 140deg), inset: (x: 4pt), outset: (y: 4pt), width: 100%, radius: 4pt)` gives:◆
- For more advanced use-cases, you can also pass a function as the `block-style`. It will be called with one argument, either `"left"` or `"right"`, depending on the side the note will be placed on. Inside the function, `context` is available.◆■

■ Purple 2

```
#let block-style = (side) => {
  if side == "left" {
    (stroke: (left: none, rest: 0.5pt + purple), outset: (left: marginalia.get-left().far, rest: 4pt))
  } else {
    (stroke: (right: none, rest: 0.5pt + purple), outset: (right: marginalia.get-right().far, left: 9pt, rest: 4pt))
  }
}
#note(block-style: block-style)[Purple]
#note(side: "inner", block-style: block-style)[Purple 2]
```

- This is a note with a dotted stroke above.

So is this.

○ This is a note with a green background and flush-numbering: true.

So is this.

- ◆ This is a note with an outset green background.

So is this.

◇ Purple

3 Wide Blocks

The command `#wideblock[]` can be used to wrap content in a wide block which spans into the margin-note-column.

Note: when using an asymmetric page layout with `setup.book: true`, wideblocks which span across pagebreaks are messy, because there is no way for the wideblock to detect the pagebreak and adjust its position after it.

▲ Wow!

It is possible to use notes in a wide block:□▲. They will automatically shift downwards to avoid colliding with the wideblock.▲

`#wideblock(side: "inner") [...]`: The `side` option allows extending the block into the inside margin instead. This is analogous to the `side` option on notes and notefigures and allows placing notes in their usual column.

□ Voila.

In this manual, an inner wideblock is used to set the appendix to make it take up fewer pages. This is also why the appendix is no longer using `setup.book: true`.♥

♥ Notes above a `#wideblock[]` will shift upwards if necessary.

`#wideblock(side: "both") [...]`: Additionally, wideblocks can extend on both sides, for extra wide content. This is especially useful for figures, more on that below.

4 Figures

4.1 Notefigures

For small figures, you can place them in the margin with `#notefigure()`. It accepts all arguments `#figure()` takes (except placement and scope), plus all arguments `#note[]` takes (except `align-baseline`). However, by default it has no marker, and to get a marker like other notes, you must pass



Figure 1: A notefigure.



Figure 2: A marked notefigure.



Figure 3: Aligned to top of figure with `dy: 0pt`.



Figure 5: Figure with custom numbering

NB: `show-caption` expects a function with two arguments, check the [docs](#).

`numbering: marginalia.note-numbering`, and it will get a marker like other notes:

Additionally, the `dy` argument now takes a relative length, where `100%` is the distance between the top of the figure content and the first baseline of the caption. By default, figures have a `dy` of `0pt - 100%`, which results in the caption being aligned horizontally to the text.

A label can be attached to the figure using the `label` argument, as was done here for Figure 2. (Sadly, it is not possible to attach labels normally.)

Notefigures can also be given `side`, `text-style`, `par-style`, and `block-style` parameters, – like `#note[]` – as is demonstrated in Figure 4. Furthermore, the `numbering`, `anchor-numbering`, and `flush-numbering` parameters work as expected.³²

Note that `#show figure.caption: /**/` rules are ignored for `#notefigure[]`s, use the `show-caption` parameter instead.

4.2 Large Figures

For larger figures, use the following set and show rules if you want top-aligned captions:

```
#set figure(gap: 0pt) // necessary in both cases
#set figure.caption(position: top)
#show figure.caption.where(position: top): note.with(numbering: none, align-baseline: false)
```



If you want bottom-aligned captions, it gets a bit more complicated, as there isn't a way (yet) to align a `#note[]` with its bottom edge, so we need to measure the caption:

```
#set figure(gap: 0pt) // necessary in both cases
#set figure.caption(position: bottom) // (this is the default)
#show figure.caption.where(position: bottom): it => context {
  let text-style= (* your text-style here *)
  let par-style= (* your par-style here *)
  let height = measure(
    width: marginalia._config.get().outer.width,
    { // we need to set the style here to simulate measuring the note
      set text(size: 9.35pt, style: "normal", weight: "regular", ..text-style)
      set par(spacing: 1.2em, leading: 0.5em, hanging-indent: 0pt, ..par-style)
    }
  ).height
  note(numbering: none, anchor-numbering: none, dy: -height, align-baseline: false, shift: "avoid", keep-order: true, text-style: text-style, par-style: par-style, it)
}
```



Figure 4: Styled figure.

Figure 6: A figure.

Figure 7: A figure.

Top	Wide Figures For wide figures, simply place a figure in a wideblock. The caption gets placed beneath the figure automatically, courtesy of regular wide-block-avoidance.	Top (no shift) Top (no shift, no baseline align) (this is assuming you have one of the above <code>show</code> rules)
-----	--	---

```
#wideblock(figure(image(..), caption: [A figure in a wide block.]))
```



Figure 8: A figure in a wide block.



Figure 9: A figure in a reversed wide block.



Figure 10: A figure in an extra-wide wideblock.

5 Other Tidbits

5.1 Absolute Placement

You can place notes in absolute positions relative to the page using `place`:

```
#place(top, note(numbering: none, side: "inner")[Top])
#place(bottom, note(numbering: none, side: "inner")[Bottom])
```

To avoid these notes moving about, use `shift: false` (or `shift: "ignore"` if you don't mind overlaps.)

```
#place(top, note(numbering: none, shift: false)[Top (no shift)])
#place(bottom, note(numbering: none, shift: false)[Bottom (no shift)])
```

By default, notes are aligned to their first baseline. To align the top of the note instead, set `align-baseline` to `false`.

5.2 Headers and Background

This is not (yet) a polished feature and requires to access `marginalia._config.get().book` to read the respective config option. In your documents, consider removing this check and simplifying the `if` a bit.◇

Here's how the headers in this document were made:

```
#set page(header: context {
  marginalia.notecounter.update(0)
  let book = marginalia._config.get().book
  let leftm = marginalia.get-left()
  let rightm = marginalia.get-right()
  if here().page() > 1 {
    wideblock(side: "both", {
      box(width: leftm.width, {
```

◇ Also, please don't `.update()` the `marginalia._config` directly, this can easily break the notes.

Bottom		Bottom (no shift) Bottom (no shift, no baseline al.)
--------	--	---

```

    if not (book) or calc.odd(here().page()) [
      Page
      #counter(page).display("1 of 1", both: true)
    ] else [
      #datetime.today().display(**/)
    ]
  })
  h(leftm.sep)
  box(width: 1fr, smallcaps[Marginalia])
  h(rightm.sep)
  box(width: rightm.width, {
    if not (book) or calc.odd(here().page()) [
      #datetime.today().display(**/)
    ] else [
      Page
      #counter(page).display("1 of 1", both: true)
    ]
  })
})
}
})

```

- Not that you should copy them, they're mostly here to showcase the columns and help me verify that everything gets placed in the right spot.

And here's the code for the lines in the background:■

```

#set page(background: context {
  let leftm = marginalia.get-left()
  let rightm = marginalia.get-right()
  place(top, dy: marginalia._config.get().top,
    line(length: 100%, stroke: luma(90%)))
  place(top, dy: marginalia._config.get().top - page.header-ascent,
    line(length: 100%, stroke: luma(90%)))
  place(bottom, dy: -marginalia._config.get().bottom,
    line(length: 100%, stroke: luma(90%)))
  place(dx: leftm.far,
    rect(width: leftm.width, height: 100%, stroke: (x: luma(90%))))
  place(dx: leftm.far + leftm.width + leftm.sep,
    rect(width: 10pt, height: 100%, stroke: (left: luma(90%))))
  place(right, dx: -rightm.far,
    rect(width: rightm.width, height: 100%, stroke: (x: luma(90%))))
  place(right, dx: -rightm.far - rightm.width - rightm.sep,
    rect(width: 10pt, height: 100%, stroke: (right: luma(90%))))
})

```

♥ Notes on the other side are usually fine though.

6 Troubleshooting / Known Bugs

- If the document needs multiple passes to figure out page-breaks, it can break the note positioning.
 - This can usually be resolved by placing a `#pagebreak()` or `#pagebreak(weak: true)` in an appropriate location.
- Relatedly, everything breaks if you try to use pages with width or height set to `auto`, as this package needs to know the actual measurements of the page to figure out where to place stuff.
- Nested notes may or may not work.▲ In nearly all cases, they seem to lead to a “layout did not converge within 5 attempts” warning, so it is probably best to avoid them if possible.
 - Just use multiple paragraphs in one note, or place multiple notes in the main text instead.
 - If really necessary, use `shift: "ignore"` on the nested notes and manually set `dy`.
- If `book` is `true`, wideblocks that break across pages are broken. Sadly there doesn’t seem to be a way to detect and react to page-breaks from within a block, so I don’t know how to fix this.
- If you encounter anything else which looks like a bug to you, please create an “issue” on GitHub if no-one else has done so already.

□ This can happen for example with outlines which barely fit/don’t fit onto the page.

▲ In this manual, for example, it works fine (with warnings) here, ▲ but not on the first page.♥

▲ Probably because there aren’t many other notes around.

7 Thanks

Many thanks go to Nathan Jessurun for their drafting package, which has served as a starting point and was very helpful in figuring out how to position margin-notes.

The wideblock functionality was inspired by the one provided in the tufte-memo template.

Also shout-out to tidy, which was used to produce the appendix.

(This project is not affiliated with https://marginalia-search.com/, but that is *also* a cool project.)

A Detailed Documentation of all Exported Symbols

Breaking Changes

0.2.1

- The functions `configure()` and `page-setup()` have been combined into one `setup()` function.

A.1 `get-left`

Returns a dictionary with the keys `far`, `width`, `sep` containing the respective widths of the left margin on the current page. (On both even and odd pages.)

Requires context.

Parameters

`get-left()` → dictionary

A.2 `get-right`

Returns a dictionary with the keys `far`, `width`, `sep` containing the respective widths of the right margin on the current page. (On both even and odd pages.)

Requires context.

Parameters

`get-right()` → dictionary

A.3 `note`

Create a marginnote. Will adjust it's position downwards to avoid previously placed notes, and upwards to avoid extending past the bottom margin.

Breaking Changes

0.1.5

- `reverse` has been replaced with `note.side`.
→ use ``side: "inner"`` instead of ``reverse: true``
- `numbered` has been replaced with `note.numbering`.
→ use ``numbering: "none"`` instead of ``numbered: false``

Parameters

```
note(  
  numbering: none function string ,  
  anchor-numbering: none auto function string ,  
  flush-numbering: auto boolean ,  
  side: auto "outer" "inner" "left" "right" ,  
  dy: length ,  
  align-baseline: boolean ,  
  keep-order: boolean ,  
  shift: boolean auto "avoid" "ignore" ,  
  text-style: dictionary ,  
  par-style: dictionary ,  
  block-style: dictionary function ,  
  <body>: content  
)
```

Functions:

- `get-left()`
- `get-right()`
- `note()`
- `note-numbering()`
- `notefigure()`
- `setup()`
- `wideblock()`

Variables:

- `notecounter`
- `note-markers`
- `note-markers-alternating`

Mostly internal.

Mostly internal.

numbering: note-numbering `none` or `function` or `string`

Function or numbering-string to generate the note markers from the notecounter. If none, will not step the notecounter.

Examples:

- `(..i) ⇒ super(numbering("1", ..i))` for superscript numbers
- `(..i) ⇒ super(numbering("a", ..i))` for superscript letters
- `marginalia.note-numbering.with(repeat: false, markers: ())` for small blue numbers

anchor-numbering: `auto` `none` or `auto` or `function` or `string`

Used to generate the marker for the anchor (i.e. the one in the surrounding text)

- If `auto`, will use the given `note.numbering`.

flush-numbering: `auto` `auto` or `boolean`

Disallow note markers hanging into the whitespace.

- If `auto`, acts like `false` if `note.anchor-numbering` is `auto`.

side: `auto` `auto` or `"outer"` or `"inner"` or `"left"` or `"right"`

Which side to place the note. `auto` defaults to `"outer"`. In non-book documents, `"outer"/"inner"` are equivalent to `"right"/"left"` respectively.

dy: `0pt` `length`

Initial vertical offset of the note. Note may get shifted still to avoid other notes.

align-baseline: `true` `boolean`

Whether to align the baselines or not.

- If `false`, the top of the note is aligned with the main-text baseline.

keep-order: `false` `boolean`

Notes with `keep-order: true` are not re-ordered relative to one another.

shift: `auto` `boolean` or `auto` or `"avoid"` or `"ignore"`

Whether the note may get shifted around to avoid other notes.

- `true`: The note may shift to avoid other notes, wide-blocks and the top/bottom margins.
- `false`: The note is placed exactly where it appears, and other notes may shift to avoid it.
- `"avoid"`: The note is only shifted if shifting other notes is not sufficient to avoid a collision. E.g. if it would collide with a wideblock or a note with `shift: false`.
- `"ignore"`: Like `false`, but other notes do not try to avoid it.
- `auto: true` if numbered, `"avoid"` otherwise.

text-style: (size: `9.35pt`, style: `"normal"`, weight: `"regular"`) `dictionary`

Will be used to `set` the text style.

par-style: (spacing: `1.2em`, leading: `0.5em`, hanging-indent: `0pt`) `dictionary`

Will be used to `set` the par style.

block-style: (width: 100%) dictionary or function

Will be passed to the block containing the note body. If this is a function, it will be called with "left" or "right" as its argument, and the result is passed to the block.

A.4 note-numbering

Format note marker

Parameters

`note-numbering`(markers: array(string), repeat: boolean, <.>, <number>: int) → content

markers: note-markers-alternating array(string)

```
#for i in array.range(1,15) [
  #note-numbering(markers: note-markers, i)
]\
#for i in array.range(1,15) [
  #note-numbering(markers: note-markers-
alternating, i)
]
```



repeat: true boolean

Whether to (true) loop over the icons, or (false) continue with numbers after icons run out.

```
#for i in array.range(1,15) [
  #note-numbering(repeat: true, i)
]\
#for i in array.range(1,15) [
  #note-numbering(repeat: false, i)
]
```



A.5 notefigure

Creates a figure in the margin.

Parameters numbering, anchor-numbering, flush-numbering, side, keep-order, shift, text-style, par-style, and block-style work the same as for `note()`.

Breaking Changes

0.1.5

- reverse has been replaced with `notefigure.side`.
→ use `side: "inner"` instead of `reverse: true`
- numbered has been replaced with `notefigure.numbering`.
→ use `numbering: marginalia.note-numbering` instead of `numbered: true`

Parameters

```
notefigure(
  numbering: none function string,
  anchor-numbering: none auto function string,
  flush-numbering: auto boolean,
  side: auto "outer" "inner" "left" "right",
  dy: relative length,
  keep-order: boolean,
  shift: boolean auto "avoid" "ignore",
  text-style: dictionary,
  par-style: dictionary,
  block-style: dictionary function,
  show-caption: function,
  gap: length,
  label: none label,
  <content>: content,
  <..figureargs>: arguments
) → content
```

numbering: `none` `none` or `function` or `string`

Same as `note.numbering`, but with different default value.

anchor-numbering: `auto` `none` or `auto` or `function` or `string`

Used to generate the marker for the anchor (i.e. the one in the surrounding text)

- If `auto`, will use the given `notefigure.numbering`.

flush-numbering: `auto` `auto` or `boolean`

Disallow note markers hanging into the whitespace.

- If `auto`, acts like `false` if `note.anchor-numbering` is `auto`.

side: `auto` `auto` or `"outer"` or `"inner"` or `"left"` or `"right"`

Which side to place the note. `auto` defaults to `"outer"`. In non-book documents, `"outer"/"inner"` are equivalent to `"right"/"left"` respectively.

dy: `0pt - 100%` `relative length`

How much to shift the note. `100%` corresponds to the height of content + gap + the first baseline.

Thus `dy: 0pt - 100%` aligns the text and caption baselines.

text-style: (size: `9.35pt`, style: `"normal"`, weight: `"regular"`) `dictionary`

Will be used to `set` the text style.

par-style: (spacing: `1.2em`, leading: `0.5em`, hanging-indent: `0pt`) `dictionary`

Will be used to `set` the par style.

block-style: (width: `100%`) `dictionary` or `function`

Will be passed to the block containing the note body (this contains the entire figure). If this is a function, it will be called with `"left"` or `"right"` as its argument, and the result is passed to the block.

```
show-caption: (number, caption) => {
  number
  caption.supplement
  [ ]
  caption.counter.display(caption.numbering)
  caption.separator
  caption.body
} function
```

A function with two arguments, the number and the caption. Will be called as the caption show rule.

If `notefigure.numbering` is `none`, number will be `none`.

`gap: 0.55em` `length`

Pass-through to `#figure()`, but used to adjust the vertical position.

`label: none` `none` or `label` A label to attach to the figure.

`<content>` `content` Positional

The figure content, e.g. an image. Pass-through to `#figure()`, but used to adjust the vertical position.

`<..figureargs>` `arguments` Positional Pass-through to `#figure()`.
(E.g. caption)

A.6 setup

This will update the marginalia config and setup the page with the provided config options. (This means this will insert a pagebreak.)

Use as

```
#show: marginalia.setup.with(/* options here */)
```

The default values for the margins have been chosen such that they match the default typst margins for a4. It is strongly recommended to change at least one of either inner or outer to be wide enough to actually contain text.

Breaking Changes

0.1.5

- `numbering` has been replaced with `note.numbering/notefigure.numbering`.
→ set ``numbering: /**/`` directly on your notes instead of via `setup()`.
Use `#let note = note.with(numbering: /**/)` for consistency.
- `flush-numbers` has been replaced by `note.flush-numbering`.
→ set ``flush-numbering: true`` directly on your notes instead of via `setup()`.
Use `#let note = note.with(flush-numbering: /**/)` for consistency.

0.2.1

- This function does no longer apply the configuration partially, but will reset all unspecified options to the default. Additionally, it replaces the `page-setup()` function that was needed previously and is no longer called `configure()`

Parameters

```
setup(
  inner: dictionary,
  outer: dictionary,
  top: length,
  bottom: length,
  book: boolean,
  clearance: length,
  <body>: content
)
```

inner: (far: 5mm, width: 15mm, sep: 5mm) dictionary

Inside/left margins.

- far: Distance between edge of page and margin (note) column.
- width: Width of the margin column.
- sep: Distance between margin column and main text column.

The page inside/left margin should equal far + width + sep.

If partial dictionary is given, it will be filled up with defaults.

outer: (far: 5mm, width: 15mm, sep: 5mm) dictionary

Outside/right margins. Analogous to inner.

top: 2.5cm length Top margin.

bottom: 2.5cm length Bottom margin.

book: false boolean

- If **true**, will use inside/outside margins, alternating on each page.
- If **false**, will use left/right margins with all pages the same.

clearance: 12pt length Minimal vertical distance between notes and to wide blocks.

A.7 wideblock

Creates a block that extends into the outside/right margin.

Note: This does not handle page-breaks sensibly. If `config.book = false`, this is not a problem, as then the margins on all pages are the same. However, when using alternating page margins, a multi-page wideblock will not work properly. To be able to set this appendix in a many-page wideblock, this code was used:

```
#show: marginalia.setup.with(..config, book: false)
#wideblock(side: "inner") [...]
```

Breaking Changes

0.1.5

- reverse and double have been replaced with `wideblock.side`.
 - use ``side: "inner"`` instead of ``reverse: true``
 - use ``side: "both"`` instead of ``double: true``

Parameters

`wideblock(side: auto "outer" "inner" "left" "right" "both", <body>: content) → content`

`side: auto` `auto` or `"outer"` or `"inner"` or `"left"` or `"right"` or `"both"`

Which side to extend into. `auto` defaults to `"outer"`. In non-book documents, `"outer"/"inner"` are equivalent to `"right"/"left"` respectively.

A.8 notecounter counter

The counter used for the note icons.

If you use `note-numbering()` without `note-numbering.repeat`, it is recommended you reset this occasionally, e.g. per heading or per page.

`notecounter.update(1)`

Mostly internal.

A.9 note-markers

Icons to use for note markers.

`("♦", "●", "■", "▲", "♥", "◇", "○", "□", "△", "♡")`

A.10 note-markers-alternating

Icons to use for note markers, alternating filled/outlined.

`("●", "○", "♦", "◇", "■", "□", "▲", "△", "♥", "♡")`