hydra

/ˈhaɪdrə/

v0.6.2 2025-07-11 MIT

Query and display headings in your documents and templates.

TINGER
tinger@tinger.dev

Hydra provides a simple API to query for headings und section like elements and display them in your document's headers. It aids in creating headers and footers with navigational snippets.

Table of Contents

Part I
Guide

I.1 Introduction

HYDRA is a package which currently provides a single function with the same name `#hydra`. This function can be used to query and display section elements, such as headings,legal paragraphs, documentation sections, and whatever else may semantically declare the start of a document's parts. This is most commonly used inside the header of a document, such that a reader always has a good idea where they are when flipping through pages.

Here's an example of how `#hydra` can be used:

```
// Show the current chapter in the header, but not on chapter title pages.
#set page(header: context hydra(1))
```

The following sections explain some core concepts and go over some features, when they are useful and how you can enable/disable them.

I.1.1 Terminology

The remainder of this document assumes that you are familiar with some of HYDRA's various terms. There's no need to read the definitions now, but you can come back here when you encounter one of those terms.

element Refers to any type of content that you may use as a section marker, these are most commonly headings, but depending on your use case may also be other elements.

primary element An *element* which is primarily looked for and meant to be displayed.

ancestor element An *element* which is the immediate or transitive ancestor to the *primary element*. A level 3 heading is an ancestor to both level 2 (directly) and level 1 headings (transitively).

scope The scope refers to the part of a document which is between the closest ancestors of a primary element.

active element The active element refers to whatever element is considered for display. While this is usually the previous primary element, it may sometimes be the next primary element.

I.1.2 Scoping

Scoping is the primary mechanism with which #hydra determines which elements to display. The search for a primary element is always bounded to its scope, if no primary element is found within a scope, then none is displayed. Consider the following contrived document:

```
= Chapter 1
== Section 1.1

= Chapter 2
=== Subsection 2.0.1
#hydra(2)
```

```
Chapter 1
└ Section 1.1
Chapter 2
└ <none>
  └ Subsection 2.0.1
```

Here, #hydra is used with 2 as its selector argument, this means it shoudl look for level 2 headings, let's call them sections, with level 1 headings being chapters. Because of this, #hydra will only search within the current chapter, displaying Section 1.1 would simply be wrong there. This is the *scope*, it's given by the relationship of the primary element (sections) and the ancestor elements (chapters). Therefore, #hydra would not find any suitable candidate to display and will not display anything.

I.2 Features

Let's go over some of HYDRA's features in no particular order, some of these features are enabled by default and can be disabled, others are deliberately turned off.

I.2.1 Contextual

#hydra will take contextual information into account to provide good defaults. These include inferring the reading direction and binding from the page and text styles respectively and using the top margin to correctly identify primary elements on page starts. This is used to offer correct handling of books as seen in Section I.2.2.b or to remove redundant headers as in Section I.2.2.

I.2.2 Redundancy Checks

#hydra is generally used for heading-like elements, i.e. elements which annotate a section inside of a document. Whenever #hydra is used in a place where its output is considered redundant for the reader, it will not show any output by default. The following sections explain those checks more closely and will generally assume that `#hydra` is looking for headings.

Skip Starting Pages

When a new page starts and introduced a chapter or heading it's usually unecessary to show that same chapter or section in the header. Infact, for chapters this is undesirable too. If `#hydra` is used with `⟨skip-starting⟩: true` on such a starting page, it will not show anything. This is turned on by default.

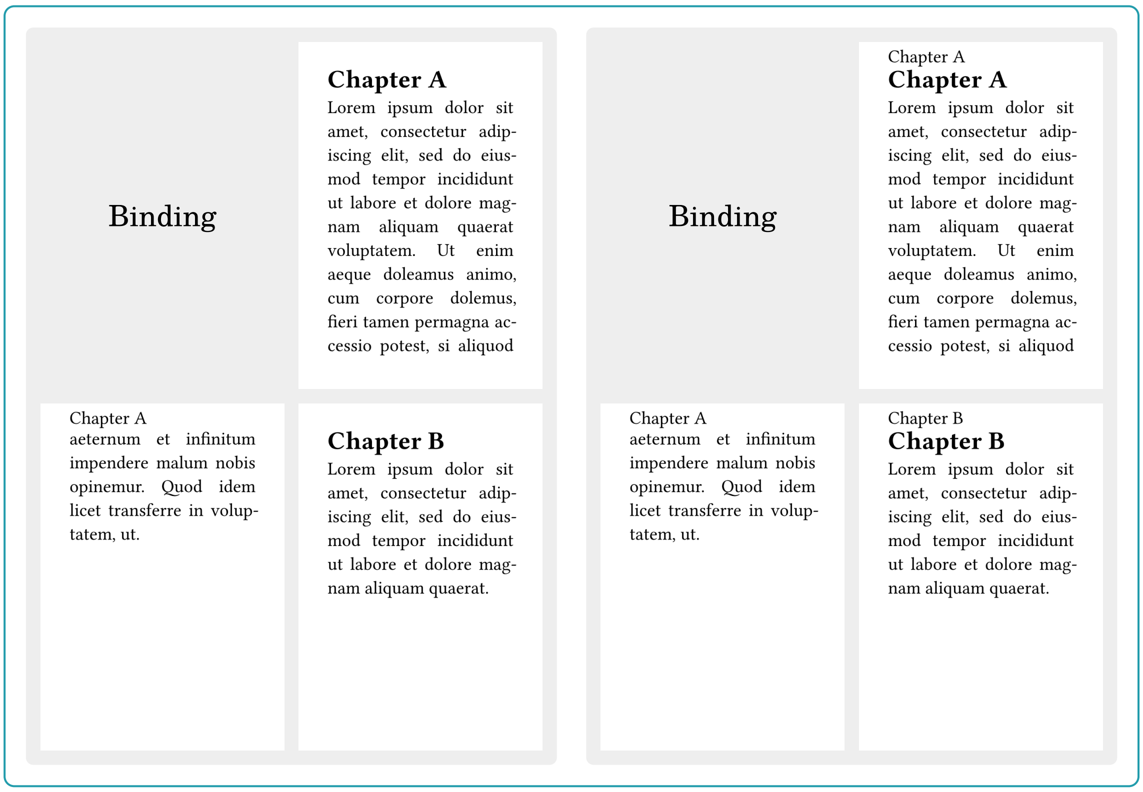

Binding
Chapter A
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod
Chapter A
aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.
Chapter B
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.
Binding
Chapter A
Chapter A
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 animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod
Chapter A
aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.
Chapter B
Chapter B
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.

Figure 1 Two example documents showing the difference between `⟨skip-starting⟩: true` (left) and `⟨skip-starting⟩: false` (right).

For more complex selectors this will not correctly work if the first element on this page is an ancestor, see hydra#8. You should also make sure you don't use show rules which affect the vertical starting position of the heading.

```
// do
#show heading: it => block(v(8cm) + it)

// don't
#show heading: it => v(8cm) + it
```

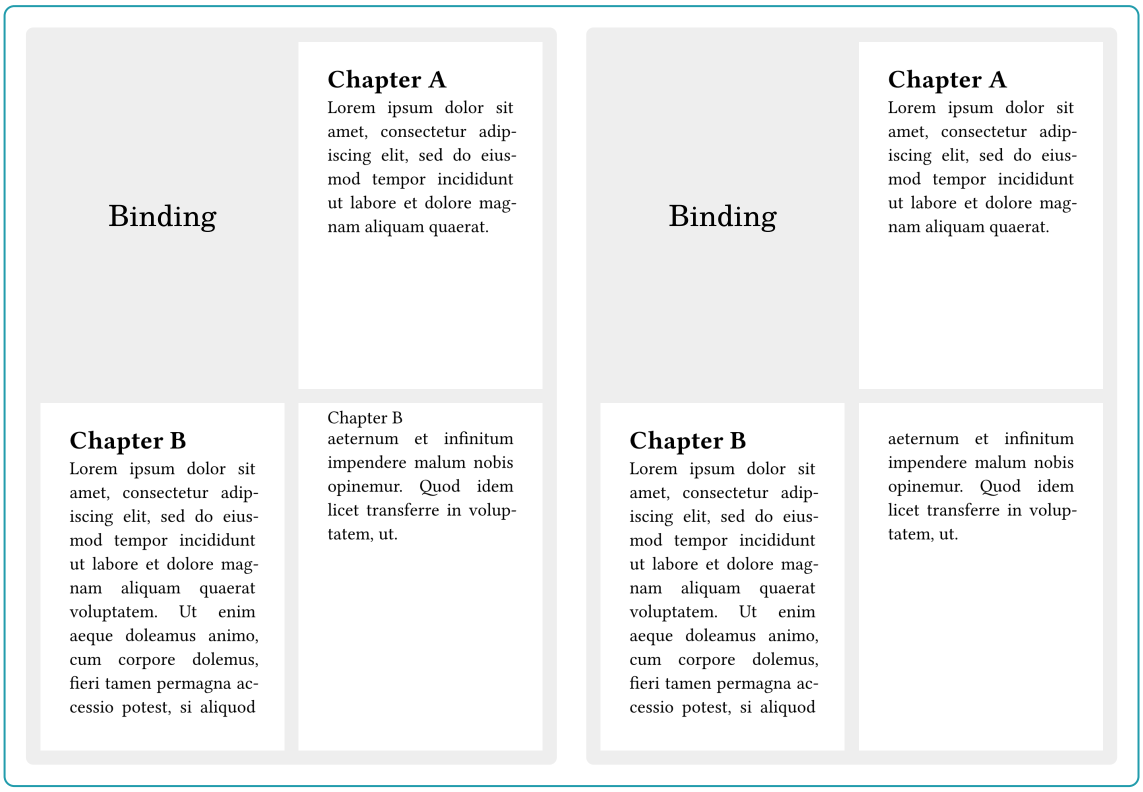
Listing 1 Two different show rules which look the same but have an impact on `#hydra`. The first rule will allow `#hydra` to correctly detect elements, the second one will not, this is related to how Typst calculates an element location.

The example above serves as a quick fix, but if you're using `#v(8cm)` for vertical spacing in headings you may want to take a look at `block.above` instead.

Book Mode

Let's say you're writing a book, this means that for a reader, there are always two page visible when reading, the *trailing* (even) and the *leading* (odd) page. For left-to-right documents these would correspond to the *left* and *right* page respectively, for right-to-left documents this is reversed.

If `#hydra` is used on a *leading* page with `⟨book⟩: true`, then it will not show an active element, if it is still visible on the *trailing* page. This is turned off by default.



Binding

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

Chapter B
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 animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod

Chapter B
aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.

Binding

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

Chapter B
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 animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod

aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.

Figure 2 Two example documents showing the difference between `⟨book⟩: false` (left) and `⟨book⟩: true` (right).

I.2.3 Anchoring

To use `#hydra` outside of the page header, an `#anchor` must be placed, otherwise `#hydra` will have problems identifying which elements are where. `#hydra` will always use the last anchor it finds to search, it doesn't have to be inside the header, but should generally be, otherwise the behavior may be unexpected.

```
#import "@preview/hydra:0.6.2": hydra, anchor
#set page(header: anchor(), footer: context hydra())
```

Listing 2 An example of using `#anchor`.

The above example shows how an `#anchor` can be used to use `#hydra` in the page footer.

I.2.4 Custom Elements

HYDRA is built with custom elements in mind. Some documents may use other elements for chapters or section-like content. HYDRA allows defining its own selectors for tight control over how elements are queried.

Let's say you're using a custom element for chapters by defining a figure with a custom type:

```
#let chapter = figure.with(kind: "chapter", supplement: [Chapter])
// ... show rules and additional setup

#chapter[Introduction]
#chapter[Main]
= Section 1.1
== Subsection 1.1.1
= Section 1.2
#chapter[Annex]
```

Listing 3 An example document using custom elements.

Note that this example is contrived, assuming the user wanted to use a single `=` for sections instead of chapters, they could instead use `#set heading(offset: 1)`. This kind of use case was more common before this feature existed.

If you now want to to be able to use these custom chapters with `#hydra`, you can do so by defining your own `hydra-selector`:

```
#import "@preview/hydra:0.6.2": hydra, selectors

#let chap = figure.where(kind: "chapter")
#let sect = selectors.custom(heading.where(level: 1), ancestor: chap)

// Display the chapter on the left and the section on the right.
#set page(header: context if calc.odd(here().page()) {
  align(left, hydra(chap))
} else {
  align(right, hydra(sect))
})
```

Listing 4 An example of how to use custom `hydra-selector`s.

The usage of `selectors.custom` allows specifying an element's ancestors, to ensure the scope is correctly defined. The selectors module also contains some useful default selectors for headings.

I.3 Frequently Asked Questions

The following questions and answers largely use the simple heading use case, but may apply to any custom elements and selectors.

Q How can I use `#hydra` in the page footer?

A You can do so my placing an `#anchor` in the page header, see Section I.2.3.

Q Why does `#hydra` not show a heading where I want it to?

A `#hydra` will automatically detect where showing a heading would be redundant and omit it. See Section I.2.2 on when that is the case, as well as how it is detected.

Q Why does `#hydra` not show the correct heading?

A
- Similar to the previous question, if your document is largely empty save for a few headings, then you may encounter a bug like hydra#7 and can resolve it the same way.
- As before, if you use custom selectors make sure to read Section I.2.4 on how to correctly define ancestors

If you encounter an issue with `#hydra` reporting the wrong active element, please report it at GitHub:tingerrr/hydra.

Q I updated HYDRA, why doesn't it work anymore?

A This depends on the update you've done:
- You updated to a new *major* version (the `1` in `1.2.3`): Make sure to read the changelog for any breaking changes.
- You updated to a new *minor* version (the `2` in `1.2.3`): Make sure to read the changelog for any breaking changes. Once HYDRA reaches maturity (`1.0.0`) *minor* version bumps will no longer allow breaking changes.
- You updated to a new *patch* version (the `3` in `1.2.3`): If you used unstable APIs, then there was never a guarantee that they stay the way they are across any version bump. Check the changelog or source code to see if the API you used is still available. Read Section II to see which APIs are stable or unstable.

  However, if you have not used any unstable APIs, then you've found a bug, please report it at GitHub:tingerrr/hydra.

Part II
Reference

II.1 Custom Types

The following type definitions are used to simplify the documentation. It's mostly pseudo code at the moment, `a | b` refers to a type of either `a` or `b`, i.e. a type union. Other than that it is very similar to Typst itself, using type hints in place of values. Most of these types are fairly unimportant for the actual end user the most important ones are:

- `hydra-selector`: used as an alternative target to `#hydra.sel`
- `hydra-context`: given to almost any internal function as well as various callback arguments on `#hydra`
- `candidates`: given to `#hydra.prev-filter` and `#hydra.next-filter`

Internal functions do not validate these types, therefore incorrect usage may break on any patch version even if it previously worked out of chance, conformance to these schemas should always result in a working API of HYDRA.

```
let queryable = label | function | selector
```

Any type which can be used in `query`, `function` refers to the subset of element functions which are locatable.

Defines a selector for an ancestor or primary element.

```
let hydra-selector = (
  target: queryable,
  filter: ((hydra-context, candidates) => bool) | none,
)
```

Defines a pair of primary and ancestor element selectors.

```
let full-selector = (
  primary: hydra-selector,
  ancestors: hydra-selector | none,
)
```

Defines the candidates that have been found in a specific context.

```
let candidates = (
  primary: (prev: content | none, next: content | none, last: content |
  none),
  ancestor: (prev: content | none, next: content | none),
)
```

Defines the options passed to `#hydra` and resolved contextual information needed for querying and displaying.

```
let hydra-context = (
  prev-filter: (hydra-context, candidates) => bool,
  next-filter: (hydra-context, candidates) => bool,
  display: (hydra-context, content) => content,
  skip-starting: bool,
  use-last: bool,
  book: bool,
  anchor: label | none,
  anchor-loc: location,
  primary: hydra-selector,
  ancestors: hydra-selector,
)
```

II.2 Modules

stable

II.2.1 hydra

The package entry point. All functions validate their inputs and panic using error messages directed at the end user.

`#anchor` `#hydra`

`#anchor → content`

An anchor used to search from. When using `#hydra` outside of the page header, this should be placed inside the page header to find the correct searching `hydra-context`. `#hydra` always searches from the last anchor it finds, if and only if it detects that it is outside of the top-margin.

context

```
#hydra(
  ⟨prev-filter⟩: auto,
  ⟨next-filter⟩: auto,
  ⟨display⟩: auto,
  ⟨skip-starting⟩: true,
  ⟨use-last⟩: false,
  ⟨book⟩: false,
  ⟨anchor⟩: <hydra-anchor>,
  ..⟨sel⟩
) → content
```

Query for an element within the bounds of its ancestors. The `hydra-context` passed to various callbacks contains the resolved top-margin, the current location, as well as the binding direction, primary and ancestor element selectors and customized functions.

Argument

`⟨prev-filter⟩: auto` `function` | `auto`

This function is called at most once before rendering the candidate and should return true if they are eligible for display. The primary next candidate may be none. If this is `auto` no filter is applied and every candidate is considered eligible.

Signature: `( hydra-context , candidates )→ bool`

Argument

`⟨next-filter⟩: auto` `function` | `auto`

This function is called at most once before rendering the candidate and should return true if they are eligible for display. The primary prev candidate may be none. If this is `auto` no filter is applied and every candidate is considered eligible.

Signature: (`hydra-context` , `candidates`)→ `bool`

Argument

`⟨display⟩: auto` `function` | `auto`

A function which receives the `hydra-context` and a candidate element to display. If this is `auto`, the default implementaion will be used.

Signature: (`hydra-context` , `content`)→ `content`

Argument

`⟨skip-starting⟩: true` `bool`

Whether `#hydra` should show the current candidate even if it's on top of the current page.

Argument

`⟨use-last⟩: false` `bool`

Whether `#hydra` should show the name of the first or last candidate on the page.

Argument

`⟨book⟩: false` `bool`

Whether the binding direction should be considered for redundancy. If the binding direction is set it'll be used to check for redundancy when an element is visible on the previous page.

Argument

`⟨anchor⟩: <hydra-anchor>` `label` | `none`

The label to use for the anchor if `#hydra` is used outside the header. If this is `none`, the anchor is not searched.

Argument

`..⟨sel⟩` `queryable` | `full-selector` | `int`

The element to look for, to use other elements than headings, read the documentation on selectors. This can be an element function or selector, or an integer declaring a heading level.

unstable

II.2.2 core

The core logic module. Some functions may return results with error messages that can be used to panic or recover from instead of panicking themselves.

#display	#get-top-margin	#is-on-starting-page
#execute	#is-active-redundant	#locate-last-anchor
#get-candidates	#is-active-visible	

`#display(⟨ctx⟩)[candidate]` → `content`

Display a heading's numbering and body, this is the default implementation of `#hydra.display`.

This will panic if it doens't receive a `#heading` as its `candidates`.

Argument
`⟨ctx⟩` `hydra-context`

The context in which the element was found.

Argument
`⟨candidate⟩` `content`

The heading to display.

context

`#execute(⟨ctx⟩)` → `content`

Execute the core logic to find and display elements for the given context. The `anchor-loc` of the context will be augmented using the current typst context.

Argument
`⟨ctx⟩` `hydra-context`

The context for which to find and display the element.

context

`#get-candidates(⟨ctx⟩, ⟨scope-prev⟩: true, ⟨scope-next⟩: true)` → `candidates`

Get the element candidates for the given context.

Argument
`⟨ctx⟩` `hydra-context`

The context for which to get the `candidates`.

Argument
`⟨scope-prev⟩: true` `bool`

Whether the search should be scoped by the first ancestor element in this direction.

Argument
`⟨scope-next⟩: true` `bool`

Whether the search should be scoped by the first ancestor element in this direction.

context

`#get-top-margin → length`

Returns the current top margin.

context

`#is-active-redundant(⟨ctx⟩, ⟨candidates⟩) → bool`

Check if showing the active element would be redundant in the given context.

Argument
`⟨ctx⟩` `hydra-context`

The context in which the redundancy of the previous primary candidate should be checked.

Argument
`⟨candidates⟩` `candidates`

The candidates for this context.

context

`#is-active-visible(⟨ctx⟩, ⟨candidates⟩) → bool`

Checks if the previous primary candidate is still visible.

Argument
`⟨ctx⟩` `hydra-context`

The context in which the visibility of the previous primary candidate should be checked.

Argument
`⟨candidates⟩` `candidates`

The candidates for this context.

context

`#is-on-starting-page(⟨ctx⟩, ⟨candidates⟩) → bool`

Checks if the current `hydra-context` is on a starting page, i.e. if the next candidates are on top of this `hydra-context`'s page.

Argument
`⟨ctx⟩` `hydra-context`

The context in which the visibility of the next `candidates` should be checked.

Argument
`⟨candidates⟩` `candidates`

The candidates for this `hydra-context`.

context

`#locate-last-anchor(⟨ctx⟩) → location`

Get the last anchor location.

Panics if the last `#anchor` was not on the page of this context.

Argument

`⟨ctx⟩` `hydra-context`

The context from which to start.

stable

II.2.3 selectors

Contains functions used for creating custom selectors.

`#by-level` `#custom` `#sanitize`

`#by-level(⟨min⟩: none, ⟨max⟩: none, ..⟨exact⟩) → hydra-selector`

Create a heading selector for a given range of levels.

Argument
`⟨min⟩: none` `int` | `none`

The inclusive minimum level to consider as the primary heading.

Argument
`⟨max⟩: none` `int` | `none`

The inclusive maximum level to consider as the primary heading.

Argument
`..⟨exact⟩` `int` | `none`

The exact level to consider as the primary element.

`#custom(⟨element⟩, ⟨filter⟩: none, ⟨ancestors⟩: none, ⟨ancestors-filter⟩: none) → hydra-selector`

Create a custom selector for `#hydra`.

Argument
`⟨element⟩` `queryable`

The primary element to search for.

Argument
`⟨filter⟩: none` `function`

The filter to apply to the element.

Signature: `(hydra-context, candidates) → bool`

Argument
`⟨ancestors⟩: none` `queryable`

The ancestor elements, this should match all of its ancestors.

Argument
`⟨ancestors-filter⟩: none` `function`

The filter applied to the ancestors.

Signature: `(hydra-context, candidates) → bool`

`#sanitize(⟨name⟩, ⟨sel⟩, ⟨message⟩: auto) → hydra-selector`

Turn various values into a `hydra-selector`.

This function is considered unstable, it may change at any time or disappear entirely.

Argument

`⟨name⟩` `str`

The name to use in the assertion message.

Argument

`⟨sel⟩` `queryable` | `full-selector` | `int`

The selector to sanitize.

Argument

`⟨message⟩: auto` `str` | `auto`

The assertion message to use.

`unstable`

II.2.4 util

Utlity functions and values.

unstable

II.2.5 util/core

Utlity functions.

`#auto-or(⟨value⟩, ⟨default⟩, ⟨check⟩: auto) → any`

An alias for `or-default.with(check: auto)`.

Argument
`⟨value⟩`

The value to check.

Argument
`⟨default⟩`

The function to produce the default value with.

Argument
`⟨check⟩: auto`

The sentinel value to check for.

`#none-or(⟨value⟩, ⟨default⟩, ⟨check⟩: none) → any`

An alias for `or-default.with(check: none)`.

Argument
`⟨value⟩`

The value to check.

Argument
`⟨default⟩`

The function to produce the default value with.

Argument
`⟨check⟩: none`

The sentinel value to check for.

`#or-default(⟨value⟩, ⟨default⟩, ⟨check⟩: none) → any`

Substitute `#or-default.value` for the return value of `#or-default.default` if it is equal to `#or-default.check` value.

Argument
`⟨value⟩`

The value to check.

Argument
`⟨default⟩`

The function to produce the default value with.

Argument
`⟨check⟩: none`

The sentinel value to check for.

`#page-binding(⟨dir⟩) → alignment`

Returns the page binding for a text direction.

Source: `page.rs#L368-L373`[1]

Argument
`⟨dir⟩` `direction`

The direction to get the page binding for.

`#text-direction(⟨lang⟩) → direction`

Returns the text direction for a given language, defaults to `#ltr` for unknown languages.

Source: `lang.rs#L50-L57`[2]

Argument
`⟨lang⟩` `str`

The languge to get the text direction for.

`#queryable-functions` `array`

A list of `queryable` element functions.

[1]https://github.com/typst/typst/blob/9646a132a80d11b37649b82c419833003ac7f455/crates/typst/src/layout/page.rs#L368-L373

[2]https://github.com/typst/typst/blob/9646a132a80d11b37649b82c419833003ac7f455/crates/typst/src/text/lang.rs#L50-57

unstable

II.2.6 util/assert

Assertions used for input and state validation.

`#element`	`#queryable`
`#enum`	`#types`

`#element(⟨name⟩, ⟨element⟩, ..⟨expected-funcs⟩, ⟨message⟩: auto) → none`

Assert that `element` is an element creatd by one of the given `expected-funcs`.

Argument
`⟨name⟩` `str`

The name to use for the value in the assertion message.

Argument
`⟨element⟩` `any`

The value to check for.

Argument
`..⟨expected-funcs⟩` `str | auto`

The assertion message to use.

Argument
`⟨message⟩: auto` `type`

The expected element functions of `#element.element`.

`#enum(⟨name⟩, ⟨value⟩, ..⟨expected-values⟩, ⟨message⟩: auto) → none`

Assert that `value` is any of the given `expected-values`.

Argument
`⟨name⟩` `str`

The to name use for the value in the assertion message.

Argument
`⟨value⟩` `any`

The value to check for.

Argument
`..⟨expected-values⟩` `type`

The expected variants of `value`.

Argument
`⟨message⟩: auto` `str | auto`

The assertion message to use.

`#queryable(⟨name⟩, ⟨value⟩, ⟨message⟩: auto)`

Assert that `value` can be used in `query`.

Argument
`⟨name⟩` `str`

The name to use for the value in the assertion message.

Argument
`⟨value⟩` `any`

The value to check for.

Argument
`⟨message⟩: auto` `str` | `auto`

The assertion message to use.

`#types(⟨name⟩, ⟨value⟩, ..⟨expected-types⟩, ⟨message⟩: auto) → none`

Assert that `value` is of any of the given `expected-types`.

Argument
`⟨name⟩` `str`

The name to use for the value in the assertion message.

Argument
`⟨value⟩` `any`

The value to check for.

Argument
`..⟨expected-types⟩` `type`

The expected types of `value`.

Argument
`⟨message⟩: auto` `str` | `auto`

The assertion message to use.

Part III

Index