# Developer's Toolbox

Manual

---

typst toolbox 0.1.0 MIT

Maycon F. Melo*

# Contents



*https://github.com/mayconfmelo

# Description

Easily implement handy features from a curated collection of frequently used package functionality, such as data storage or custom defaults. This library was created as part of the development of*my other Typst projects;*[1] it contains functionality shared across multiple projects that would otherwise need to be maintained and updated individually, but is now centralized in a single place. This is not intended to be a full-fledged development toolset, but rather a compartmentalization of shared resources.

You can use this library without restrictions as a development aid, especially for packages.

# Main

```
#import "@preview/toolbox:0.1.0"
```

## Custom Defaults

```
#default(
  when: false,
  value: (:),
```

[1] https://typst.app/universe/search/?q=author%3A%22Maycon%20F.%20Melo%22

```
    otherwise: (:),
    disabled,
  )
```

Substitutes Typst defaults for custom ones, allowing to set new defaults that can be easily changed using `#set` rules; it works by mapping custom defaults to the originals, replacing them when in use.

**`when:`** boolean

Condition to apply custom default.

**`value:`** dictionary any

Custom default value (condition is `true`).

**`otherwise:`** dictionary any

Alternative value when default is not being used (condition is `false`).

**`disabled`** boolean

Disable custom default when `true`.

This command is commonly used inside `#set` rules.

## Content to String

```
#content2str(data)
```

Converts content to string.

**`data`** content

Content data

# Storage

```
#import "preview/toolbox:0.1.0": storage
```

## Set Namespace

```
#storage.namespace(name)
```

Set the namespace used as storage. Namespaces allows multiple packages/templates to use `#storage` at the same time, each accessing its own proper space. This changes where data is stored **globally**, use it with caution.

**`name`** string *(required)*

Namespace name.

# Add Data

```
#storage.add(
  key,
  value,
  append: false,
  namespace: auto,
)
```

Insert a new entry in the storage.

**`key`** string

Storage entry name.

**`value`** ant

Value to be stored.

**`append:`** boolean

If entry already exists, append value when `true`; replaces it when `false`.

**`namespace:`** string

Add to the given namespace.

# Remove Data

```
#storage.remove(
  key,
  namespace: auto,
)
```

Removes an existing entry from storage.

**`key`** string

Storage entry name.

**`namespace:`** string

Remove from the given namespace.

## Get Data

```
#storage.get(
  ..args,
  namespace: auto,
)
```

Retrieves a value from storage, or the entire namespace itself.

**`args.pos().at(0)`** string

Storage entry name; if not set, returns the whole namespace.

**`args.pos().at(1)`** any

Default value returned when the storage entry is not found; otherwise returns `none`.

**`namespace:`** string

Get from the given namespace.

## Get Final Data

```
#storage.final(
    ..args,
  namespace: auto,
)
```

Retrieves the final value (or namespace) state from storage.

**`args.pos().at(0)`** string

Storage entry name; if not set, returns the final state for the whole namespace.

**`args.pos().at(1)`** any

Default value returned when the storage entry is not found; otherwise returns `none`.

**`namespace:`** string

Final state from the given namespace.

## Reset Database

```
#storage.reset(
  data,
  namespace: auto,
)
```

Set a new value for the entire storage namespace. While the new value can be of any type, the other storage commands can only work with `dictionary` values.

**`data`** dictionary any

New namespace value.

**`namespace:`** string

Reset the given namespace.

# Components

```
#import "@preview/toolbox: 0.1.0": comp
```

## Paper-friendly URLs

```
#comp.url(target, id, text)
```

Creates a paper-friendly link, attached to a footnote containing the URL itself to ensure readability when printed.

**`target`** string label *(required)*

URL used as link and footnote, or label referencing a previous `#url` command.

**`id`** label

Optional label for further reference.

**`text`** string content

Text shown as link (fallback to the URL).

## Package URLs

```
#comp.pkg(target, id)
```

Generates paper-friendly links for general package/library repositories.

**`target`** string label

`"https://example.com/name"`
`"https://example.com/{name}/slug/"`

Package URL, or label referencing a previous `#pkg` command. The package name is inferred from the last URL slug or retrieved from curly brackets.

**`id`** label

Optional label for further reference.

# Callout

```
#comp.callout(
  icon: "information-circle",
  title: none,
  text: (:),
  background: (:),
  body,
)
```

Creates a highly customizable callout box.

**`icon:`** string

Icon name, as set by*Heroicons*[2] .

**`title:`** string content none

Title, if any.

**`text:`** color dictionary

`#text` options (the special `#text.title` is used for title).

**`background:`** color dictionary

`#block` options.

# Get

```
#import "@preview/toolbox: 0.1.0": get
```

[2]https://heroicons.com/

# Null Value

```
#get.null
```

A null value that matches only itself. Useful as substitute for `none` default values, as it is truly unique and not naturally returned by anything in Typst.

# Automatic Values

```
#get.auto-val(
  origin,
  replace,
)
```

Sets a meaningful value to `auto` values: if a value is automatic, returns the replacement; otherwise, returns the value unchanged.

**`origin`** auto any

Value to be checked.

**`replace`** any

Replacement for the value if `auto`.

# Date

```
#get.date(..date)
```

Create a `#datetime` date from named and/or positional arguments, combined or not. Panics if two values are set for the same data component — like a positional and also a named value for the year.

**`..date`** arguments *(required)*

`(year, month, day)`

Components of the data, as positional and/or named arguments; fallback to current year, month 1, and day 1 when these components are not set.

# Has

```
#import "@preview/toolbox: 0.1.0": has
```

# Field

```
#has.field(data, values)
```

Check if content has a given field.

**`data`** content
  The content itself.

**`values`** string array of strings
  One or more field names.

# Key

```
#has.key(data, values)
```

Check if dictionary or module has a given key.

**`data`** dictionary module
  The dictionary or module itself.

**`values`** string array of strings
  One or more key names.

# Value

```
#has.value(data, values)
```

Check if dictionary or module has a given value.

**`data`** dictionary module
  The dictionary or module itself.

**`values`** string array
  One or more values.

# Item

```
#has.item(data, values)
```

Check if an array has one or more given items.

**data** string
The array itself.

**values** string array of strings
One or more items.

# It's

```
#import "@preview/toolbox: 0.1.0": its
```

## None

```
#its.none-val(data)
```

Check whether a value is `none`.

**data** none any
Value to be checked.

## Null

```
#its.null()
```

Check whether a value is `#get.null`.

**data** any
Value to be checked.

## Empty

```
#its.empty(data)
```

Check whether a value is empty: `""` or `[]` or `()` or `(:)`.

**data** any
Value to be checked.

## Context

```
#its.context-val(data)
```

Check whether a value is a `#context()`.

**`data`** any
Value to be checked.

## Sequence of Contents

```
#its.sequence(data)
```

Check whether a value is a sequence of contents.

**`data`** any
Value to be checked.

## Space Content

```
#its.space(data)
```

Check whether a value is a content with just a space.

**`data`** any
Value to be checked.

## Function

```
#its.func(data, values)
```

Check whether a value function is one of the given ones.

**`data`** any
Value to be checked.

**`values`** function array of functions
One or more functions.

## Type

```
#its.type(data, values)
```

Check whether a value type is one of the given ones.

**`data`** any

Value to be checked.

**`values`** type array of types

One or more types.

# Copyright



The logo was obtained from Flaticon website.

The Fluent support is a fork of a*linguify*[3] feature, and all the overall project concept is heavily inspired in this great package.

[3] https://typst.app/universe/package/linguify