t4t

Tools For Typst

v0.2.0 2023-08-12 MIT

A utility package for typst package authors

Jonas Neugebauer

https://github.com/jneug/typst-tools4typst

Tools for Typst (t4t in short) is a utility package for Typst package and template authors. It provides solutions to some recurring tasks in package development.

The package can be imported or any useful parts of it copied into a project. It is perfectly fine to treat t4t as a snippet collection and to pick and choose only some useful functions. For this reason, most functions are implemented without further dependencies.

Hopefully, this collection will grow over time with **Typst** to provide solutions for common problems.

Table of contents

N.1. Test functions
N.1.1. Command reference 2
N.2. Default values9
N.2.1. Command reference 10
N.3. Assertions
N.3.1. Command reference 14
N.4. Element helpers 17
N.4.1. Command reference 17
N.5. Math functions
N.5.1. Command reference
N.6. Alias functions
I. Index

```
#import "@preview/t4t:0.2.0": is
```

These functions provide shortcuts to common tests like #is.eq(). Some of these are not shorter as writing pure typst code (e.g. a == b), but can easily be used in .any() or .find() calls:

```
1 // check all values for none
2 if some-array.any(is-none) {
3    ...
4 }
5
6 // find first not none value
7 let x = (none, none, 5, none).find(is.not-none)
8
9 // find position of a value
10 let pos-bar = args.pos().position(is.eq.with("|"))
```

There are two exceptions: #is-none() and #is-auto(). Since keywords can't be used as function names, the is module can't define a function to do is.none(). Therefore the functions #is-none() and #is-auto() are provided in the base module of t4t:

```
#import "@preview/t4t:0.2.0": is-none, is-auto
```

The is submodule still has these tests, but under different names (#is.n() and #is.non() for none and #is.a() and #is.aut() for auto).

N.1.1. Command reference

```
#a()
                            #elem()
                                                        #none-of-type()
#all-of-type()
                            #empty()
                                                        #not-a()
                                                        #not-any()
#any()
                            #eq()
                                                        #not-auto()
#any-type()
                            #has()
#arr()
                            #label()
                                                        #not-empty()
#aut()
                            #loc()
                                                         #not-n()
#bool()
                            #n()
                                                        #not-none()
                            #neg()
#color()
                                                        #one-not-none()
#content()
                                                        #same-type()
                            #neq()
#dict()
                            #non()
                                                         #sequence()
```

#is.neg(test)

Creates a new test function, that is true, when test is false.

Can be used to create negations of tests like:

```
#let not-raw = is.neg(is.raw)
```

```
test function boolean
```

test to negate. #is.eq(compare, value) Tests if values compare and value are equal. any compare first value value any second value #is.neq(compare, value) Tests if values compare and value are not equal. compare any first value value any second value #is.n(..values) Tests if any one of values is equal to none. ..values any values to test #is.non() Alias for n(). #is.not-none(..values)

Tests if none of values is equal to none.

```
..values
                                                                                      any
   values to test
#is.not-n()
  Alias for not-none()
#is.one-not-none(..values)
  Tests, if at least one value in values is not equal to none.
  Useful for checking mutliple optoinal arguments for a valid value:
  #if is.one-not-none(..args.pos()) [
    #args.pos().find(is.not-none)
  ]
  ..values
                                                                                      any
   values to test
#is.a(..values)
  Tests if any one of values is equal to auto.
  ..values
                                                                                      any
   values to test
#is.aut()
  Alias for a()
#is.not-auto(..values)
  Tests if none of values is equal to auto.
  ..values
                                                                                      any
   values to test
```

#is.not-a()

Alias for not-auto()

#is.empty(value)

Tests, if value is *empty*.

A value is considered *empty* if it is an empty array, dictionary or string, or the value none.



#is.not-empty(value)

Tests, if value is not *empty*.

See empty() for an explanation what *empty* means.



#is.any(..compare, value)

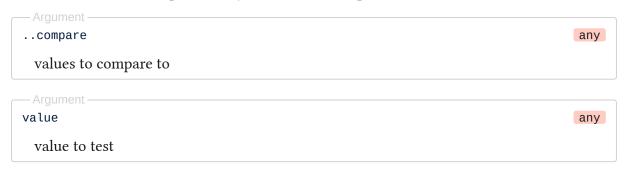
Tests, if value is not *empty*.

See empty() for an explanation what *empty* means.



#is.not-any(..compare, value)

Tests if value is not equals to any one of the other passed in values.



#is.has(..keys, value)

Tests if value contains all the passed keys.

Either as keys in a dictionary or elements in an array. If value is neither of those types, false is returned.



#is.type(t, value)

Tests if value is of type t.



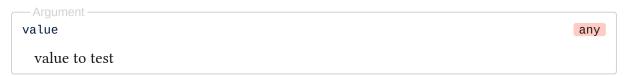
#is.dict(value)

Tests if value is of type dictionary.



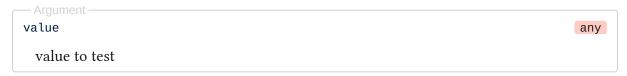
#is.arr(value)

Tests if value is of type array.



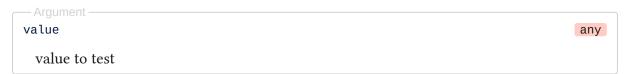
#is.content(value)

Tests if value is of type content.



#is.label(value)

Tests if value is of type label.



#is.color(value)

Tests if value is of type color.



#is.stroke(value)

Tests if value is of type stroke.

```
value value to test
```

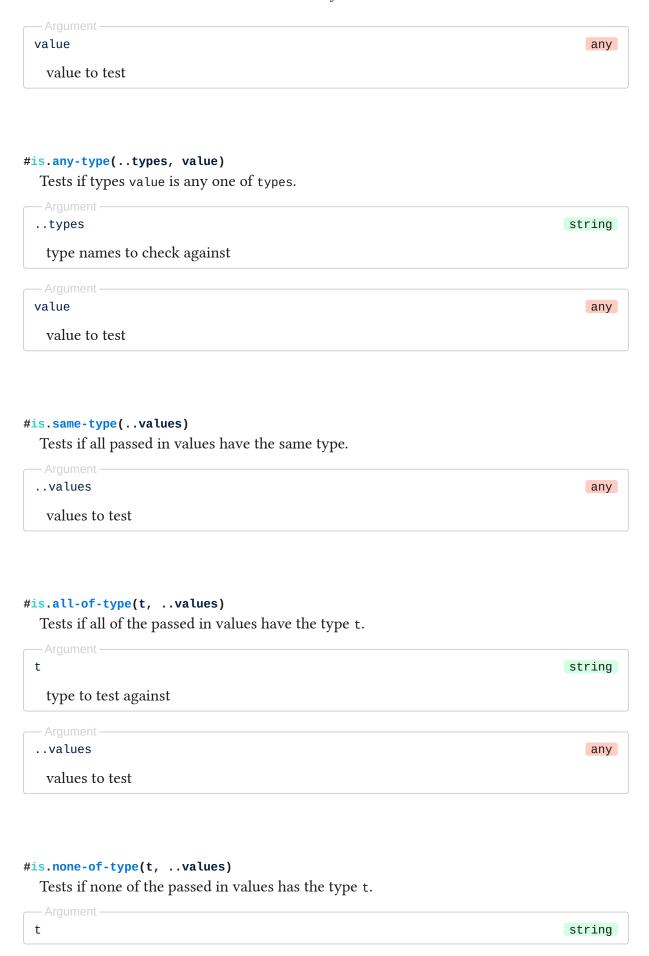
#is.loc(value)

Tests if value is of type location.

```
value value to test
```

#is.bool(value)

Tests if value is of type boolean.





#is.elem(func, value)

Tests if value is a content element with value.func() == func.

If func is a string, value will be compared to repr(value.func()), instead. Both of these effectively do the same:

```
#is.elem(raw, some_content)
#is.elem("raw", some_content)
```





#is.sequence(value)

Tests if value is a sequence of content.

N.2. Default values

```
#import "@preview/t4t:0.2.0": def
```

These functions perform a test to decide, if a given value is *invalid*. If the test *passes*, the default is returned, the value otherwise.

Almost all functions support an optional do argument, to be set to a function of one argument, that will be applied to the value, if the test fails. For example:

```
1 // Sets date to a datetime from an optional
2 // string argument in the format "YYYY-MM-DD"
3 let date = def.if-none(
4  datetime.today(), // default
5  passed_date, // passed in argument
6  do: (d) >= {// post-processor
7  d = d.split("-")
8  datetime(year=d[0], month=d[1], day=d[2])
9  }
10 )
```

N.2.1. Command reference

```
#as-arr() #if-auto() #if-none()
#if-any() #if-empty() #if-not-any()
#if-arg() #if-false() #if-true()
```

#def.if-true(test, default, do: none, value)

Returns default if test is true, value otherwise.

If test is false and do is set to a function, value is passed to do, before being returned.

```
Argument
default
default value to return

Argument
do: none
postprocessor for value

Argument
value
the value eto test
```

#def.if-false(test, default, do: none, value)

Returns default if test is false, value otherwise.

If test is true and do is set to a function, value is passed to do, before being returned.

```
test boolean a test result
```

0.2.1 Default values



#def.if-none(default, do: none, value)

Returns default if value is none, value otherwise.

If value is not none and do is set to a function, value is passed to do, before being returned.



#def.if-auto(default, do: none, value)

Returns default if value is auto, value otherwise.

If value is not auto and do is set to a function, value is passed to do, before being returned.



```
value
                                                                                    any
   the valu eto test
#def.if-any(..compare, default, do: none, value)
  Returns default if value is equal to any value in compare, value otherwise.
  #def.if-any(
    none, auto, // ..compare
                     // default
    1pt,
                     // value
    thickness
  )
  If value is in compare and do is set to a function, value is passed to do, before being returned.
  ..compare
                                                                                    any
   list of values to compare value to
 default
                                                                                    any
   default value to return
                                                                               function
 do: none
   postprocessor for value
 value
                                                                                    any
   value to test
#def.if-not-any(..compare, default, do: none, value)
  Returns default if value is not equal to any value in compare, value otherwise.
  #def.if-not-any(
    left, right, top, bottom, // ..compare
                                  // default
    left,
    position
                                  // value
  If value is in compare and do is set to a function, value is passed to do, before being returned.
  ..compare
                                                                                    any
   list of values to compare value to
```

0.2.1 Default values

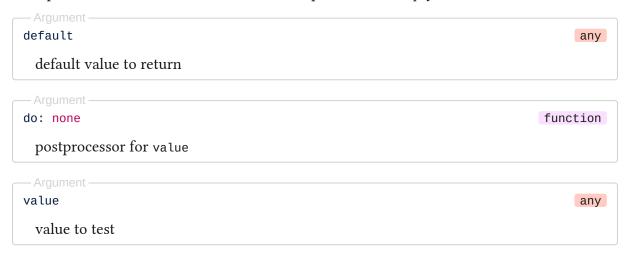


#def.if-empty(default, do: none, value)

Returns default if value is empty, value otherwise.

If value is not empty and do is set to a function, value is passed to do, before being returned.

Depends on is.empty(). See there for an explanation of empty.



#def.if-arg(default, do: none, args, key)

Returns default if key is not an existing key in args.named(), args.named().at(key) otherwise.

If value is not in args and do is set to a function, the value is passed to do, before being returned.



0.2.1 Default values



#def.as-arr(..values)

Always returns an array containing all values.

Any arrays in values will be flattened into the result. This is useful for arguments, that can have one element or an array of elements:

```
#def.as-arr(author).join(", ")
```

N.3. Assertions

```
#import "@preview/t4t:0.2.0": assert
```

This submodule overloads the default assert function and provides more asserts to quickly check if given values are valid. All functions use assert in the background.

Since a module in Typst is not callable, the assert function is now available as #assert.that(). #assert.eq() and #assert.ne() work as expected.

All assert functions take an optional argument message to set the error message for a failed assertion.

N.3.1. Command reference

```
#all-of-type() #ne() #not-any-type()
#any() #neq() #not-none()
#any-type() #new() #that()
#eq() #not-any() #that-not()
```

#assert.that()

Asserts that test is true. See assert

#assert.that-not(test, message: "")

Asserts that test is false. boolean test Assertion to test. message: "" str A message to show if the assert fails. #assert.eq() Asserts that two values are equal. #assert.ne() Asserts that two values are not equal. #assert.neq() Alias for ne() #assert.not-none() Asserts that a value is not none #assert.any(..values, value, message: "") Assert that value is any one of values. ..values any A set of values to compare value to. value any Value to compare. message: "" str A message to show if the assert fails.

0.3.1 Assertions

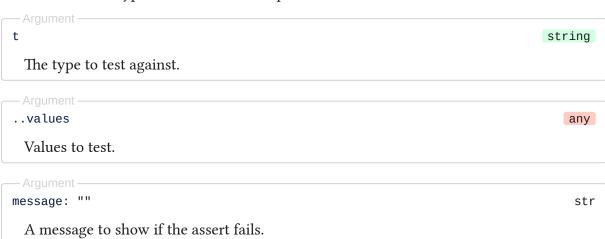
```
#assert.not-any(..values, value, message: "")
  Assert that value is not any one of values.
  assert.not-any(none, auto, 3)
  ..values
                                                                                    any
   A set of values to compare value to.
 value
                                                                                    any
   Value to compare.
 message: ""
                                                                                      str
   A message to show if the assert fails.
#assert.any-type(..types, value, message: "")
  Assert that values type is any one of types.
                                                                                 string
  ..types
   A set of types to compare the type of value to.
 value
                                                                                    any
   Value to compare.
 message: ""
                                                                                      str
   A message to show if the assert fails.
#assert.not-any-type(..types, value, message: "")
  Assert that values type is not any one of types.
  ..types
                                                                                 string
   A set of types to compare the type of value to.
 value
                                                                                    any
   Value to compare.
```

Message: "" str

A message to show if the assert fails.

#assert.all-of-type(t, ..values, message: "")

Assert that the types of all values are equal to t.



#assert.new(test)

Creates a new assertion from test.

The new assertion will take a value and pass it to test. test should return a boolean.

```
Argument
test

A test function: (string) => boolean
```

N.4. Element helpers

```
#import "@preview/t4t:0.2.0": get
```

This submodule is a collection of functions, that mostly deal with content elements and *get* some information from them. Though some handle other types like dictionaries.

N.4.1. Command reference

```
#args() #dict-merge() #stroke-thickness()
#dict() #stroke-paint() #text()
```

#get.dict(..dicts)

Create a new dictionary from the passed values.

All named arguments are stored in the new dictionary as is. All positional arguments are grouped in key/value-pairs and inserted into the dictionary:

```
#get.dict("a", 1, "b", 2, "c", d:4, e:5)
// gives (a:1, b:2, c:none, d:4, e:5)
```

#get.dict-merge(..dicts)

Recursivley merges the passed in dictionaries.

```
#get.dict-merge(
    (a: 1),
    (a: (one: 1, two:2)),
    (a: (two: 4, three:3))
)
// gives (a:(one:1, two:4, three:3))
```

Based on work by @johannes-wolf for johannes-wolf/typst-canvas.

```
#get.args(args, prefix: "")
```

Creats a function to extract values from an argument sink args.

The resulting function takes any number of positional and named arguments and creates a dictionary with values from args.named(). Positional arguments to the function are present in the result, if they are present in args.named(). Named arguments are always present, either with their value from args.named() or with the provided value.

A prefix can be specified, to extract only specific arguments. The resulting dictionary will have all keys with the prefix removed, though.

```
#get.text(element, sep: "")
```

0.4.1 Element helpers

Recursively extracts the text content of a content element.

If present, all child elements are converted to text and joined with sep.

```
#get.stroke-paint(stroke, default: rgb("#000000"))
```

Returns the color of stroke. If no thickness information is available, default is used. **Deprecated since Typst 0.7.0**: use stroke.thickness instead.

Based on work by @PgBiel for PgBiel/typst-tablex.

```
#get.stroke-thickness(stroke, default: 1pt)
```

Returns the thickness of stroke. If no thickness information is available, default is used. **Deprecated since Typst 0.7.0**: use stroke.thickness instead.

N.5. Math functions

```
#import "@preview/t4t:0.2.0": math
```

Some functions to complement the native calc module.

N.5.1. Command reference

```
#clamp() #lerp() #map()

#math.minmax(a, b)
Returns an array with the minimum of a and b as the first element and the maximum as the second:

#let (min, max) = math.minmax(a, b)

Argument
a
First value.

Argument
b
any
```

#math.clamp(min, max, value)

Second value.

0.5.1 Math functions

Clamps a value between min and max.

In contrast to clamp() this function works for other values than numbers, as long as they are comparable.

```
text-size = math.clamp(0.8em, 1.2em, text-size)

Argument
min
Maximum for value.

Argument
value
The value to clamp.
any
```

#math.lerp(min, max, t)

Calculates the linear interpolation of t between min and max.

t should be a value between 0 and 1, but the interpolation works with other values, too. To constrain the result into the given interval, use clamp():

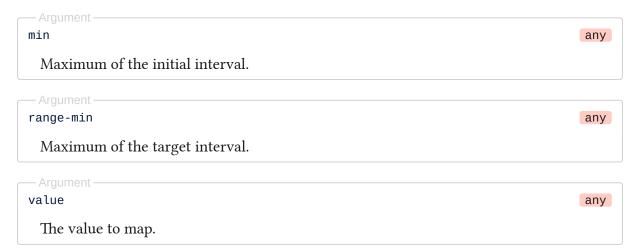
```
#let width = math.lerp(0%, 100%, x)
#let width = math.lerp(0%, 100%, math.clamp(0, 1, x))
```

```
#math.map(
    min,
    max,
    range-min,
    range-max,
    value
)
    Maps a value from the interval [min, max] into the interval [range-min, range-max]:

    #let text-weight = int(math.map(8pt, 16pt, 400, 800, text-size))
```

0.5.1 Math functions

The types of min, max and value have to be the same. The types of range-min and range-max also.



N.6. Alias functions

```
#import "@preview/t4t:0.2.0": alias
```

Some of the native Typst function as aliases, to prevent collisions with some common argument namens.

For example using numbering as an argument is not possible, if the value is supposed to be passed to the #numbering() function. To still allow argument names, that are in line with the common Typst names (like type, align ...), these alias functions can be used:

```
#let excercise( no, numbering: "1)" ) = [
    Exercise #alias.numbering(numbering, no)
2
3
```

The following functions have aliases right now:

- numbering
- align type
- label
- text

- raw
- table
- list
- enum

- terms
- grid
- stack
- columns

Part I.

Index

Α	M
#a 2, 4	#map 20
#all-of-type 8, 17	#minmax 19
#any 5, 15	
#any-type 8, 16	N
#args 18	#n 2, 3
#arr 6	#ne 14, 15
#as-arr 14	#neg 2
#aut 2, 4	#neq 3, 15
	#new 17
В	#non 2, 3
#bool 7	#none-of-type 8
	#not-a 5
C	#not-any 5, 16
#clamp 19	#not-any-type 16
#color 7	#not-auto4
#content 7	#not-empty 5
D	#not-n 4
	#not-none 3, 15
#dict 6, 18	#numbering 21
#dict-merge 18	
E	0
#elem 9	#one-not-none 4
#empty 5	S
#eq	
7, 3, 13	#same-type 8
H	#sequence 9
#has 6	#stroke 7
	#stroke-paint 19
I and the second	#stroke-thickness 19
#if-any 12	T
#if-arg 13	#text 18
#if-auto 11	#that 14
#if-empty 13	#that-not 15
#if-false 10	#type 6
#if-none 11	,
#if-not-any 12	
#if-true 10	
#is-auto 2	
#is-none 2	
L	
_	
#label 7	
#lerp 20	
#loc7	