# t4t

v0.2.0 MIT

A utility package for typst package authors

# Jonas Neugebauer

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

# Table of contents

N.1. Module is	2
N.2. Module def	)
N.3. Module alias	3
N.4. Module assert 13	3
N.5. Module get	1
N.6. Module math 15	5
I Index	

# N.1. Module is

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

#### #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 to negate.
```

#### #eq(compare, value)

Tests if values compare and value are equal.

```
Argument compare none first value

Argument value none second value
```

#### #neq(compare, value)

Tests if values compare and value are not equal.



second value

#### #n(..values)

Tests if any one of values is equal to none.

```
Argument ...values none values to test
```

#### #non()

Alias for n().

# #not-none(..values)

Tests if none of values is equal to none.

```
Argument
..values

values to test
```

#### #not-n()

Alias for not-none()

# #one-not-none(..values)

Tests, if at least one value in values is not equal to none.

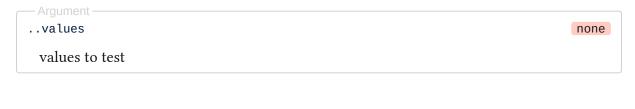
Useful for checking mutliple optoinal arguments for a valid value:

```
#if is.one-not-none(..args.pos()) [
    #args.pos().find(is.not-none)
]
```

```
Argument ...values none values to test
```

# #a(..values)

Tests if any one of values is equal to auto.



#### #aut()

Alias for a()

# #not-auto(..values)

Tests if none of values is equal to auto.



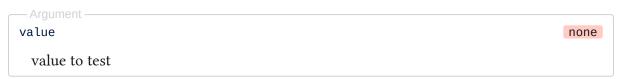
#### #not-a()

Alias for not-auto()

#### #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.



#### #not-empty(value)

Tests, if value is not empty.

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



# #any(..compare, value)

Tests, if value is not empty.

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



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

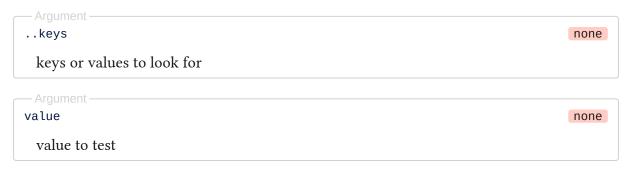
Tests if value is not equals to any one of the other passed in values.



# #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.



# #type(t, value)

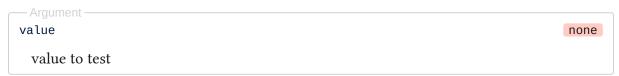
Tests if value is of type t.

t Argument	none
name of the type	
Argument —	

value value to test

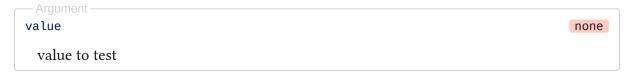
# #dict(value)

Tests if value is of type dictionary.



# #arr(value)

Tests if value is of type array.



# #content(value)

Tests if value is of type content.



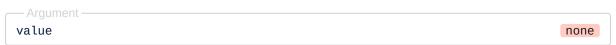
# #label(value)

Tests if value is of type label.



# #color(value)

Tests if value is of type color.



value to test

# #stroke(value)

Tests if value is of type stroke.

value value to test

#### #loc(value)

Tests if value is of type location.

value value to test

# #bool(value)

Tests if value is of type boolean.

value value to test

# #any-type(..types, value)

Tests if types value is any one of types.



#same-type(..values)

Tests if all passed in values have the same type.



# #all-of-type(t, ..values)

Tests if all of the passed in values have the type t.



# #none-of-type(t, ..values)

Tests if none of the passed in values has the type t.

```
Argument
t
type to test against

Argument
..values
values to test
```

# #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)
```

Argument — func	none
element function	
Argument —	



#### #sequence(value)

Tests if value is a sequence of content.

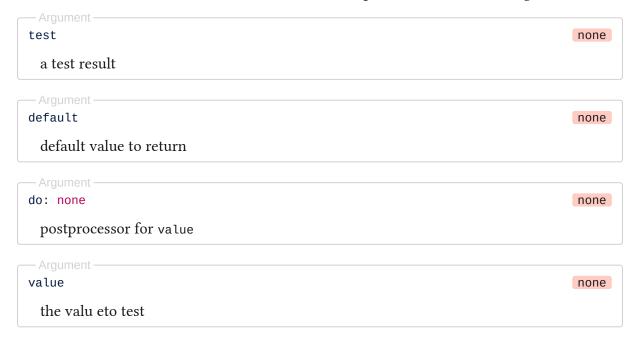
# N.2. Module def

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

#### #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.



# #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.



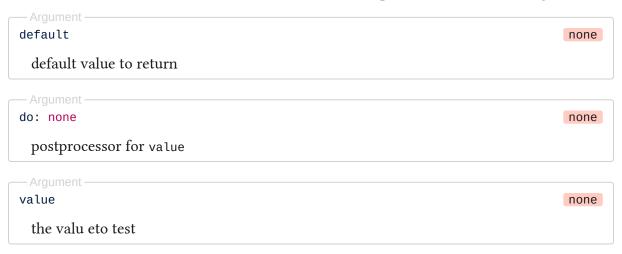
#### 0.2 Module def



#### #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.



#### #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
                                                                                  none
   the valu eto test
#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
                                                                                  none
   list of values to compare value to
 default
                                                                                  none
   default value to return
 do: none
                                                                                  none
   postprocessor for value
 value
                                                                                  none
   value to test
```

```
#if-not-any(..compare, default, do: none, value)
```

Returns default if value is not equal to any value in compare, value otherwise.

If value is in compare and do is set to a function, value is passed to do, before being returned.

```
Argument ...compare list of values to compare value to

Argument _____
```

#### 0.2 Module def

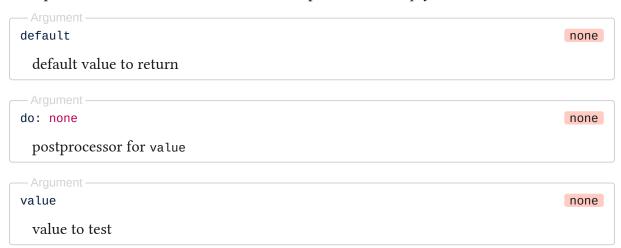


#### #if-empty(default, do: none, value)

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*.



#### #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.





#### #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. Module alias

# N.4. Module assert

• that() • that-not() • eq()

# #that()

Asserts that the passed test is true.

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

Asserts that the passed test is false.

#### #eq()

Asserts that the passed values are equal.

#### #ne()

Asserts that the passed values are not equal.

```
#neq()
```

Alias for ne()

# N.5. Module get

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

#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)
```

#### #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.

#### #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.

```
#my-func(
    width: 50%,
    text-fill: red, text-size: 1.2em
)[#lorem(5)]
```

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

Recursively extracts the text content of a content element.

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

```
#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.

```
#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.6. Module math

# Part I.

# Index

A	#non 3
#a 4	$\hbox{\#none-of-type} 8$
#all-of-type 8	#not-a 4
#any5	#not-any 5
#any-type7	#not-auto 4
#args 14	#not-empty 4
#arr6	#not-n 3
#as-arr 13	#not-none 3
#aut 4	
7,442	0
В	$\hbox{\tt\#one-not-none} \dots \dots$
#bool 7	
	S
C	#same-type 7
#color6	#sequence 9
#content 6	#stroke 7
_	# stroke-paint 15
D	$\hbox{\#stroke-thickness} 15$
#dict 6, 14	_
#dict-merge 14	Т
_	#text 15
E	#that 13
#elem 8	#that-not 13
#empty 4	#type 5
	21
#eq 2, 13	
#eq	21
#eq 2, 13	
#eq	
#eq 2, 13	
#eq	
#eq       2, 13         H       #has       5         I       #if-any       11         #if-arg       12         #if-auto       10         #if-empty       12         #if-false       9         #if-none       10         #if-true       9         L       #label       6         #loc       7	
#eq	
#eq       2, 13         H       #has       5         I       #if-any       11         #if-arg       12         #if-auto       10         #if-empty       12         #if-false       9         #if-none       10         #if-true       9         L       #label       6         #loc       7	
#eq	
#eq       2, 13         H       #has       5         I       #if-any       11         #if-arg       12         #if-auto       10         #if-empty       12         #if-false       9         #if-none       10         #if-true       9         L       #label       6         #loc       7         N       #n       3	