1 Function Documentation

concat-counter

Concatenate two counter-like objects. The state can be obtained using `counter-at()`.

Parameters

```
concat-counter(
  counter-1: counter-like ,
  counter-2: counter-like
) -> [redacted]
```

slice-counter

Slice a counter-like object. The state can be obtained using `counter-at()`.

Parameters

```
slice-counter(
  counter: counter-like ,
  end: integer  none ,
  start: integer
) -> [redacted]
```

end `integer` or `none`

The end-level of the slice (inclusive). If none, the slice goes to the last level of the counter.

start `integer`

The start-level of the slice.

Default: `1`

counter-at

Retrieves the value of the `counter-like` at the given location. Behaves similar to `counter.at()`. For concatenated counters an array of both states is returned.

This function is contextual.

Parameters

```
counter-at(
  counter: counter-like ,
  selector: label  selector  location  function
) -> array
```

selector `label` or `selector` or `location` or `function`

The place at which the counter’s value should be retrieved.

counter-get

Retrieves the value of the counter at the current location.

This is equivalent to `counter-at(counter, here())`.

This function is contextual.

Parameters

```
counter-get(counter: counter-like) -> array
```

numbify-numbering

Applies a numbering to the state of a numbify-counter. Similar to `numbering()`.

Parameters

```
numbify-numbering(
  numbering: str function array,
  seperator: content,
  ..numbers: array
) -> function
```

numbering `str` or `function` or `array`

If a string or function is provided, the numbering is applied to the combined state of all counters. If an array is provided, the numbering is applied to each counter individually and then concatenated.

seperator `content`

In case of an array numbering, this seperator is used to join the individual numberings.

Default: `"."`

..numbers `array`

The state of the numbify-counter.

override-counter

Creates a numbering function that replaces the usual counter with the provided counter. The numbering is applied to the counter using `numbify-numbering()`.

The returned function is contextual.

Parameters

```
override-counter(
  counter: counter-like,
  numbering: str function array,
  seperator: content
) -> function
```

counter `counter-like`

The new counter.

numbering `str` or `function` or `array`

The numbering that will be applied to the counter.

seperator `content`

The seperator that will be used to join the individual numberings.

Default: `"."`

heading-level-range

Returns a selector that selects a range of heading levels.

Parameters

```
heading-level-range(
  end: integer,
  start: integer
) -> selector
```

end `integer`

The end level (inclusive).

start `integer`

The start level (inclusive).

Default: `1`

reset-counter

Reset the counter. The update only occurs if the resulting content is put into the document.

counter (counter, str, label, selector, location, function): TODO

Parameters

```
reset-counter(counter) -> content
```

reset-counter-rule

Reset the counter. This is a helper function that can be used in a show-rule.

Parameters

`reset-counter-rule(counter: `[redacted]` str label selector location function) -> function`

counter [redacted] or `str` or `label` or `selector` or `location` or `function`

TODO

is-counter-operation

Checks if the argument was constructed by one of the counter-operations.

Parameters

`is-counter-operation(arg: `[redacted]`) -> boolean`

is-concat-counter

Checks if the argument was constructed by the `concat-counter` operation.

Parameters

`is-concat-counter(arg: `[redacted]`) -> boolean`

is-slice-counter

Checks if the argument was constructed by the `slice-counter` operation.

Parameters

`is-slice-counter(arg: `[redacted]`) -> boolean`

is-numbify-counter

Checks if the argument is either a `counter` or the result of a counter-operation.

Parameters

`is-numbify-counter(arg: `[redacted]`) -> boolean`

from-counter-like

Turns a `counter-like` into a `numbify-counter` by constructing the counter from the argument if necessary. A `counter-like` is a `counter`, something that a `counter` can be constructed from or the result of a counter-operation.

Parameters

`from-counter-like(counter: counter-like) -> `[redacted]