

Modul:Set/Doku

Dies ist die Dokumentationsseite für **Modul:Set**

Vorlage:Lua

This module includes a number of set operations for Lua tables. It currently has **union**, **intersection** and **complement** functions for both key/value pairs and for values only. It is a meta-module, meant to be called from other Lua modules, and should not be called directly from `#invoke`.

Inhaltsverzeichnis

1 Loading the module	1
2 union	1
3 valueUnion	1
4 intersection	2
5 valueIntersection	2
6 complement	2
7 valueComplement	2

Loading the module

To use any of the functions, first you must load the module.

```
local set = require('Module:Set')
```

union

```
set.union(t1, t2, ...)
```

Returns the union of the key/value pairs of n tables. If any of the tables contain different values for the same table key, the table value is converted to an array holding all of the different values. For example, for the tables `{foo = "foo", bar = "bar"}` and `{foo = "foo", bar = "baz", qux = "qux"}`, union will return `{foo = "foo", bar = {"bar", "baz"}, qux = "qux"}`. An error is raised if the function receives less than two tables as arguments.

valueUnion

```
set.valueUnion(t1, t2, ...)
```

Returns the union of the values of n tables, as an array. For example, for the tables $\{1, 3, 4, 5\}$, and $\{2, \text{bar} = 3, 5, 6\}$, `valueUnion` will return $\{1, 2, 3, 4, 5, 6, 7\}$. An error is raised if the function receives less than two tables as arguments.

intersection

```
set.intersection(t1, t2, ...)
```

Returns the intersection of the key/value pairs of n tables. Both the key and the value must match to be included in the resulting table. For example, for the tables $\{\text{foo} = \text{"foo"}, \text{bar} = \text{"ba"}\}$ and $\{\text{foo} = \text{"foo"}, \text{bar} = \text{"baz"}, \text{qux} = \text{"qux"}\}$, `intersection` will return $\{\text{foo} = \text{"foo"}\}$. An error is raised if the function receives less than two tables as arguments.

valueIntersection

```
set.valueIntersection(t1, t2, ...)
```

Returns the intersection of the values of n tables, as an array. For example, for the tables $\{1, 3\}$, and $\{2, \text{bar} = 3, 5, 6\}$, `valueIntersection` will return $\{3, 5\}$. An error is raised if the function receives less than two tables as arguments.

complement

```
set.complement(t1, t2, ..., tn)
```

Returns the **relative complement** of $t1, t2, \dots$, in tn . The complement is of key/value pairs. This is equivalent to all the key/value pairs that are in tn but are not in any of $t1, t2, \dots, tn-1$. For example, for the tables $\{\text{foo} = \text{"foo"}, \text{bar} = \text{"bar"}, \text{baz} = \text{"baz"}\}$ and $\{\text{foo} = \text{"foo"}, \text{bar} = \text{"baz"}, \text{qux} = \text{"qux"}\}$, `complement` would return $\{\text{bar} = \text{"baz"}, \text{qux} = \text{"qux"}\}$. An error is raised if the function receives less than two tables as arguments.

valueComplement

```
set.valueComplement(t1, t2, ..., tn)
```

This returns an array containing the **relative complement** of $t1, t2, \dots$, in tn . The complement is of values only. This is equivalent to all the values that are in tn but are not in $t1, t2, \dots, tn-1$. For example, for the tables $\{1, 2\}$, $\{1, 2, 3\}$ and $\{1, 2, 3, 4, 5\}$, `valueComplement` would return $\{4, 5\}$. An error is raised if the function receives less than two tables as arguments.