pydash Documentation

Release 2.4.2

Derrick Gilland

February 27, 2015

Contents

A utility library for doing "stuff" in a functional way. Based on the Lo-Dash Javascript library.

CHAPTER 1

Links

- Project: https://github.com/dgilland/pydash
- Documentation: http://pydash.readthedocs.org
- PyPi: https://pypi.python.org/pypi/pydash/
- TravisCI: https://travis-ci.org/dgilland/pydash

CHAPTER 2

Quickstart

The functions available from pydash can be used in two styles.

The first is by using the module directly or importing from it:

```
>>> import pydash as pyd
>>> from pydash import flatten

# Arrays
>>> flatten([1, 2, [3, [4, 5, [6, 7]]]])
[1, 2, 3, [4, 5, [6, 7]]]

>>> pyd.flatten_deep([1, 2, [3, [4, 5, [6, 7]]]])
[1, 2, 3, 4, 5, 6, 7]

# Collections
>>> pyd.pluck([{'name': 'moe', 'age': 40}, {'name': 'larry', 'age': 50}], 'name')
['moe', 'larry']

# Functions
>>> curried = pyd.curry(lambda a, b, c: a + b + c)
>>> curried(1, 2)(3)
6

# Objects
>>> pyd.omit({'name': 'moe', 'age': 40}, 'age')
{'name': 'moe'}

# Utilities
>>> pyd.times(3, lambda index: index)
[0, 1, 2]

# Chaining
>>> pyd.chain([1, 2, 3, 4]).without(2, 3).reject(lambda x: x > 1).value()
[1]
```

The second style is to use the _ instance:

```
>>> from pydash import _

>>> _.flatten([1, 2, [3, [4, 5, [6, 7]]]])
[1, 2, 3, [4, 5, [6, 7]]]

>>> _([1, 2, 3, 4]).without(2, 3).reject(lambda x: x > 1).value()
[1]
```

See also:

For further details consult *API Reference*.

CHAPTER 3

Guide

3.1 Installation

pydash requires Python >= 2.6 or >= 3.3. It has no external dependencies.

To install from PyPi:

```
pip install pydash
```

3.2 Quickstart

The functions available from pydash can be used in two styles.

The first is by using the module directly or importing from it:

```
>>> import pydash as pyd
>>> from pydash import flatten

# Arrays
>>> flatten([1, 2, [3, [4, 5, [6, 7]]]])
[1, 2, 3, [4, 5, [6, 7]]]

>>> pyd.flatten_deep([1, 2, [3, [4, 5, [6, 7]]]])
[1, 2, 3, 4, 5, 6, 7]

# Collections
>>> pyd.pluck([{'name': 'moe', 'age': 40}, {'name': 'larry', 'age': 50}], 'name')
['moe', 'larry']

# Functions
>>> curried = pyd.curry(lambda a, b, c: a + b + c)
>>> curried(1, 2)(3)
6

# Objects
>>> pyd.omit({'name': 'moe', 'age': 40}, 'age')
{'name': 'moe'}

# Utilities
>>> pyd.times(3, lambda index: index)
[0, 1, 2]
```

```
# Chaining
>>> pyd.chain([1, 2, 3, 4]).without(2, 3).reject(lambda x: x > 1).value()
[1]
```

The second style is to use the _ instance:

```
>>> from pydash import _

>>> _.flatten([1, 2, [3, [4, 5, [6, 7]]]])
[1, 2, 3, [4, 5, [6, 7]]]

>>> _([1, 2, 3, 4]).without(2, 3).reject(lambda x: x > 1).value()
[1]
```

See also:

For further details consult *API Reference*.

3.3 Lo-Dash Differences

3.3.1 Naming Conventions

pydash adheres to the following conventions:

- Function names use `snake_case` instead of `camelCase`.
- Any Lo-Dash function that shares its name with a reserved Python keyword will have an _ appended after it (e.g. `filter` in Lo-Dash would be `filter_` in pydash.
- Lo-Dash's `toArray()` is pydash's `to_list()`.
- Lo-Dash's `functions()` is pydash's `callables()`. This particular name difference was chosen in order to allow for the `functions.py` module file to exist at root of the project. Previously, `functions.py` existed in `pydash/api/` but in `v2.0.0`, it was decided to move everything in `api/` to `pydash/`. Therefore, In to avoid import ambiguities, the `functions()` function was renamed.

3.3.2 Callbacks

As of `v2.0.0`, callback functions no longer need to handle all possible arguments. Prior to `v2.0.0`, callbacks had to define all arguments or have star-args:

```
# Valid in v1
def mycallback(item, value, obj):
    pass

# Valid in v1
def mycallback(item, *args):
    pass

# Invalid in v1
def mycallback(item):
    pass
```

But in `v2.0.0` partial callback signatures are handled properly:

```
# Valid in v2
def mycallback(item, value, obj):
    pass

# Valid in v2
def mycallback(item, *args):
    pass

# Valid in v2
def mycallback(item):
    pass
```

3.3.3 Extra Aliases

The following extra function aliases exist in pydash but not in Lo-Dash:

- `pydash.utilities.prop()` >> `pydash.utilities.property_()`
- `pydash.predicates.is_re()` >> `pydash.predicates.is_reg_exp()`
- `pydash.strings.escape_re()` >> `pydash.strings.escape_reg_exp()`

3.3.4 Extra Functions

In addition to porting Lo-Dash, pydash contains functions found in lodashcontrib, lodashdeep, and lodashmath.

The following functions exist in pydash but not in Lo-Dash:

- `pydash.numerical.add()`, `pydash.numerical.sum_()`
- `pydash.arrays.append()`, `pydash.arrays.push()`
- `pydash.numerical.average()`, `pydash.numerical.avg()`, `pydash.numerical.mean()`
- `pydash.arrays.cat()`, `pydash.arrays.concat()`
- `pydash.functions.conjoin()`
- `pydash.objects.deep_get()`
- `pydash.objects.deep_has()`
- `pydash.objects.deep_map_values()`
- `pydash.collections.deep_pluck()`
- `pydash.objects.deep_property()`, `pydash.objects.deep_prop()`
- `pydash.objects.deep_set()`
- `pydash.functions.disjoin()`
- `pydash.strings.explode()`
- `pydash.objects.get_path()`
- `pydash.objects.has_path()`
- `pydash.strings.implode()`
- `pydash.arrays.intercalate()`
- `pydash.arrays.interleave()`

- pydash.arrays.intersperse()
- pydash.predicates.is_associative()
- pydash.predicates.is_even()
- pydash.predicates.is_float()
- pydash.predicates.is_decreasing()
- pydash.predicates.is_increasing()
- pydash.predicates.is_indexed()
- pydash.predicates.is_instance_of()
- pydash.predicates.is_integer()
- pydash.predicates.is_json()
- pydash.predicates.is_monotone()
- pydash.predicates.is_negative()
- pydash.predicates.is_odd()
- pydash.predicates.is_positive()
- pydash.predicates.is_strictly_decreasing()
- pydash.predicates.is_strictly_increasing()
- pydash.predicates.is_zero()
- pydash.functions.iterated()
- pydash.arrays.join()
- pydash.functions.juxtapose()
- pydash.arrays.mapcat()
- pydash.collections.mapiter()
- pydash.numerical.median()
- pydash.numerical.moving_average(),pydash.numerical.moving_avg()
- pydash.arrays.pop()
- pydash.numerical.power(),pydash.numerical.pow_()
- pydash.collections.reductions()
- pydash.collections.reductions_right()
- pydash.objects.rename_keys()
- pydash.arrays.reverse()
- pydash.numerical.round_(),pydash.numerical.curve()
- pydash.numerical.scale()
- pydash.objects.set_path()
- pydash.arrays.shift()
- pydash.numerical.slope()
- pydash.arrays.sort()

- pydash.arrays.splice()
- pydash.arrays.split_at()
- pydash.numerical.std_deviation(), pydash.numerical.sigma()
- pydash.objects.to_string()
- pydash.numerical.transpose()
- pydash.arrays.unshift()
- pydash.objects.update_path()
- pydash.strings.url()
- pydash.numerical.variance()
- pydash.numerical.zscore()

3.3.5 Function Behavior

Some of pydash's functions behave differently:

- pydash.utilities.memoize() uses all passed in arguments as the cache key by default instead of only using the first argument.

3.3.6 Templating

- pydash doesn't have template(). See *Templating* for more details.

3.4 Templating

Templating has been purposely left out of pydash. Having a custom templating engine was never a goal of pydash even though Lo-Dash includes one. There already exist many mature and battle-tested templating engines like **'Jinja2'_** and **'Mako'_** which would be much more suited to handling templating needs. However, if there was ever a strong request/justification for having templating in pydash (or a pull-request implementing it), then this decision could be re-evaluated.

3.5 Versioning

This project follows Semantic Versioning with the following caveats:

- Only the public API (i.e. the objects imported into the pydash module) will maintain backwards compatibility between MINOR version bumps.
- Objects within any other parts of the library are not guaranteed to not break between MINOR version bumps.

With that in mind, it is recommended to only use or import objects from the main module, pydash.

3.6 Upgrading

3.6.1 From v1.0.0 to v2.0.0

There were several breaking and potentially breaking changes in `v2.0.0`:

- `pydash.arrays.flatten()` is now shallow by default. Previously, it was deep by default. For deep flattening, use either `flatten(..., is_deep=True)` or `flatten_deep(...)`.
- `pydash.predicates.is_number()` now returns `False` for boolean `True` and `False`. Previously, it returned `True`.
- Internally, the files located in `pydash.api` were moved to `pydash`. If you imported from `pydash.api.<module>`, then it's recommended to change your imports to pull from `pydash`.
- The function `functions()` was renamed to `callables()` to avoid ambiguities with the module `functions.py`.

Some notable new features:

- Callback functions no longer require the full call signature definition. See *Callbacks* for more details.
- A new "_" instance was added which supports both method chaining and module method calling. See *"_" Instance* for more details.

CHAPTER 4

API Reference

Includes links to source code.

4.1 API Reference

All public functions are available from the main module.

```
import pydash

pydash.<function>
```

This is the recommended way to use pydash.

```
# OK
from pydash import where
where({})

# OK
import pydash
pydash.where({})

# NOT RECOMMENDED
from pydash.collections import where
```

Only the main pydash module API is guaranteed to adhere to semver. It's possible that backwards incompatibility outside the main module API could be broken between minor releases.

4.1.1 "_" Instance

There is a special _ instance available from `pydash` that supports method calling and method chaining from a single object:

```
from pydash import _

# Method calling
_.initial([1, 2, 3, 4, 5]) == [1, 2, 3, 4]

# Method chaining
_([1, 2, 3, 4, 5]).initial().value() == [1, 2, 3, 4]

# Method aliasing to underscore suffixed methods that shadow builtin names
```

```
_.map is _.map_
_([1, 2, 3]).map(_.to_string).value() == _([1, 2, 3]).map_(_.to_string).value()
```

The _ instance is basically a combination of using `pydash.<function>` and `pydash.chain`.

A full listing of aliased _ methods:

- `_.object` is `pydash.arrays.object_()`
- `_.slice` is `pydash.arraysslice_()`
- `_.zip` is `pydash.arrayszip_()`
- `_.all` is `pydash.collections.all_()`
- `_.any` is `pydash.collections.any_()`
- `_.filter` is `pydash.collections.filter_()`
- `_.map` is `pydash.collection.map_()`
- `_.max` is `pydash.collection.max_()`
- `_.min` is `pydash.collection.min_()`
- `_.reduce` is `pydash.collection.reduce_()`
- `_.pow` is `pydash.numerical.pow_()`
- `_.round` is `pydash.numerical.round_()`
- `_.sum` is `pydash.numerical.sum_()`
- `_.property` is `pydash.utilities.property_()`
- `_.range` is `pydash.utilities.range_()`

4.1.2 Arrays

Functions that operate on lists.

New in version 1.0.0.

`pydash.arrays.`**`append`**(*array*, **items*)

Push items onto the end of *array* and return modified *array*.

Parameters

- **array** (*list*) – List to push to.
- **items** (*mixed*) – Items to append.

Returns Modified *array*.

Return type list

Warning: *array* is modified in place.

New in version 2.2.0.

`pydash.arrays.`**`cat`**(**arrays*)

Concatenates zero or more lists into one.

Parameters **arrays** (*list*) – Lists to concatenate.

Returns Concatenated list.

Return type list

New in version 2.0.0.

pydash.arrays.**chunk**(*array*, *size=1*)

Creates a list of elements split into groups the length of *size*. If *array* can't be split evenly, the final chunk will be the remaining elements.

Parameters

- **array** (*list*) – List to chunk.
- **size** (*int, optional*) – Chunk size. Defaults to `1`.

Returns New list containing chunks of *array*.

Return type list

New in version 1.1.0.

pydash.arrays.**compact**(*array*)

Creates a list with all falsey values of array removed.

Parameters **array** (*list*) – List to compact.

Returns Compacted list.

Return type list

New in version 1.0.0.

pydash.arrays.**concat**(**arrays*)

Concatenates zero or more lists into one.

Parameters **arrays** (*list*) – Lists to concatenate.

Returns Concatenated list.

Return type list

New in version 2.0.0.

pydash.arrays.**difference**(*array*, **lists*)

Creates a list of list elements not present in the other lists.

Parameters

- **array** (*list*) – List to process.
- **lists** (*list*) – Lists to check.

Returns Difference of the lists.

Return type list

New in version 1.0.0.

pydash.arrays.**drop**(*array*, *n*)

Creates a slice of *array* with *n* elements dropped from the beginning.

Parameters

- **array** (*list*) – List to process.
- **n** (*int*) – Number of elements to drop.

Returns Dropped list.

Return type list

New in version 1.0.0.

Changed in version 1.1.0: Added `n` argument and removed as alias of `rest()`.

`pydash.arrays.`**`drop_right`**(*array*, *n*)

Creates a slice of *array* with *n* elements dropped from the end.

Parameters

- **array** (*list*) – List to process.
- **n** (*int*) – Number of elements to drop.

Returns Dropped list.

Return type list

New in version 1.1.0.

`pydash.arrays.`**`drop_right_while`**(*array*, *callback=None*)

Creates a slice of *array* excluding elements dropped from the end. Elements are dropped until the *callback* returns falsey. The *callback* is invoked with three arguments: `(value, index, array)`.

Parameters

- **array** (*list*) – List to process.
- **callback** (*mixed*) – Callback called per iteration

Returns Dropped list.

Return type list

New in version 1.1.0.

`pydash.arrays.`**`drop_while`**(*array*, *callback=None*)

Creates a slice of *array* excluding elements dropped from the beginning. Elements are dropped until the *callback* returns falsey. The *callback* is invoked with three arguments: `(value, index, array)`.

Parameters

- **array** (*list*) – List to process.
- **callback** (*mixed*) – Callback called per iteration

Returns Dropped list.

Return type list

New in version 1.1.0.

`pydash.arrays.`**`find_index`**(*array*, *callback=None*)

This method is similar to `pydash.collections.find()`, except that it returns the index of the element that passes the callback check, instead of the element itself.

Parameters

- **array** (*list*) – List to process.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Index of found item or `-1` if not found.

Return type int

New in version 1.0.0.

pydash.arrays.**find_last_index**(*array*, *callback=None*)

This method is similar to `find_index()`, except that it iterates over elements from right to left.

Parameters

- **array** (*list*) – List to process.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Index of found item or `-1` if not found.

Return type int

New in version 1.0.0.

pydash.arrays.**first**(*array*)

Return the first element of *array*.

Parameters **array** (*list*) – List to process.

Returns First element of list.

Return type mixed

See also:

- `first()` (main definition)
- `head()` (alias)
- `take()` (alias)

New in version 1.0.0.

pydash.arrays.**flatten**(*array*, *is_deep=False*)

Flattens a nested array. If *is_deep* is `True` the array is recursively flattened, otherwise it is only flattened a single level.

Parameters

- **array** (*list*) – List to process.
- **is_deep** (*bool, optional*) – Whether to recursively flatten *array*.

Returns Flattened list.

Return type list

New in version 1.0.0.

Changed in version 2.0.0: Removed `callback` option. Added `is_deep` option. Made it shallow by default.

pydash.arrays.**flatten_deep**(*array*)

Flattens a nested array recursively. This is the same as calling `flatten(array, is_deep=True)`.

Parameters **array** (*list*) – List to process.

Returns Flattened list.

Return type list

New in version 2.0.0.

pydash.arrays.**head**(*array*)

Return the first element of *array*.

Parameters **array** (*list*) – List to process.

Returns First element of list.

Return type mixed

See also:

- `first()` (main definition)
- `head()` (alias)
- `take()` (alias)

New in version 1.0.0.

pydash.arrays.**index_of**(*array*, *value*, *from_index=0*)
Gets the index at which the first occurrence of value is found.

Parameters

- **array** (*list*) – List to search.
- **value** (*mixed*) – Value to search for.
- **from_index** (*int, optional*) – Index to search from.

Returns Index of found item or `-1` if not found.

Return type int

New in version 1.0.0.

pydash.arrays.**initial**(*array*)
Return all but the last element of *array*.

New in version 1.0.0.

pydash.arrays.**intercalate**(*array*, *separator*)
Like `intersperse()` for lists of lists but shallowly flattening the result.

Parameters

- **array** (*list*) – List to intercalate.
- **separator** (*mixed*) – Element to insert.

Returns Intercalated list.

Return type list

New in version 2.0.0.

pydash.arrays.**interleave**(**arrays*)
Merge multiple lists into a single list by inserting the next element of each list by sequential round-robin into the new list.

Parameters **arrays** (*list*) – Lists to interleave.

Retruns: list: Interleaved list.

New in version 2.0.0.

pydash.arrays.**intersection**(**arrays*)
Computes the intersection of all the passed-in arrays.

Parameters **arrays** (*list*) – Lists to process.

Returns Intersection of provided lists.

Return type list

New in version 1.0.0.

pydash.arrays.**intersperse**(*array*, *separator*)
Insert a separating element between the elements of *array*.

Parameters

- **array** (*list*) – List to intersperse.
- **separator** (*mixed*) – Element to insert.

Returns Interspersed list.

Return type list

New in version 2.0.0.

pydash.arrays.**join**(*array*, *separator*)
Return the elements of *array* as a string joined with *separator*.

Parameters

- **array** (*list*) – List to join.
- **separator** (*str*) – String to join with.

Returns Joined string.

Return type str

..versionadded:: 2.2.0

pydash.arrays.**last**(*array*)
Return the last element of *array*.

New in version 1.0.0.

pydash.arrays.**last_index_of**(*array*, *value*, *from_index=None*)
Gets the index at which the last occurrence of value is found.

Parameters

- **array** (*list*) – List to search.
- **value** (*mixed*) – Value to search for.
- **from_index** (*int, optional*) – Index to search from.

Returns Index of found item or `False` if not found.

Return type int

New in version 1.0.0.

pydash.arrays.**mapcat**(*array*, *callback=None*)
Map a callback to each element of a list and concatenate the results into a single list using `cat()`.

Parameters

- **array** (*list*) – List to map and concatenate.
- **callback** (*mixed*) – Callback to apply to each element.

Returns Mapped and concatenated list.

Return type list

New in version 2.0.0.

pydash.arrays.**object_**(*keys*, *values=None*)

Creates a dict composed from lists of keys and values. Pass either a single two dimensional list, i.e. `[[key1, value1], [key2, value2]]`, or two lists, one of keys and one of corresponding values.

Parameters

- **keys** (*list*) – either a list of keys or a list of `[key, value]` pairs
- **values** (*list, optional*) – list of values to zip

Returns Zipped dict.

Return type dict

See also:

- `zip_object()` (main definition)
- `object_()` (alias)

New in version 1.0.0.

pydash.arrays.**pull**(*array*, **values*)

Removes all provided values from the given array.

Parameters

- **array** (*list*) – List to pull from.
- **values** (*mixed*) – Values to remove.

Returns Modified *array*.

Return type list

Warning: *array* is modified in place.

New in version 1.0.0.

pydash.arrays.**pull_at**(*array*, **indexes*)

Removes elements from *array* corresponding to the specified indexes and returns a list of the removed elements. Indexes may be specified as a list of indexes or as individual arguments.

Parameters

- **array** (*list*) – List to pull from.
- **indexes** (*int*) – Indexes to pull.

Returns Modified *array*.

Return type list

Warning: *array* is modified in place.

New in version 1.1.0.

pydash.arrays.**push**(*array*, **items*)

Push items onto the end of *array* and return modified *array*.

Parameters

- **array** (*list*) – List to push to.
- **items** (*mixed*) – Items to append.

Returns Modified *array*.

Return type list

Warning: *array* is modified in place.

New in version 2.2.0.

pydash.arrays.**remove**(*array*, *callback=None*)
Removes all elements from a list that the callback returns truthy for and returns an array of removed elements.

Parameters

- **array** (*list*) – List to remove elements from.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Removed elements of *array*.

Return type list

Warning: *array* is modified in place.

New in version 1.0.0.

pydash.arrays.**rest**(*array*)
Return all but the first element of *array*.

Parameters **array** (*list*) – List to process.

Returns Rest of the list.

Return type list

See also:

- rest() (main definition)
- tail() (alias)

New in version 1.0.0.

pydash.arrays.**reverse**(*array*)
Return *array* in reverse order.

Parameters **array** (*list|string*) – Object to process.

Returns list|string: Reverse of object.

New in version 2.2.0.

pydash.arrays.**shift**(*array*)
Remove the first element of *array* and return it.

Parameters **array** (*list*) – List to shift.

Returns First element of *array*.

Return type mixed

Warning: *array* is modified in place.

New in version 2.2.0.

pydash.arrays.**slice_**(*array*, *start=0*, *end=None*)
Slices *array* from the *start* index up to, but not including, the *end* index.

Parameters

- **array** (*list*) – Array to slice.
- **start** (*int, optional*) – Start index. Defaults to `0`.
- **end** (*int, optional*) – End index. Defaults to selecting the value at `start` index.

Returns Sliced list.

Return type list

New in version 1.1.0.

pydash.arrays.**sort**(*array*, *comparison=None*, *key=None*, *reverse=False*)
Sort *array* using optional *comparison*, *key*, and *reverse* options and return sorted *array*.

Note: Python 3 removed the option to pass a custom comparison function and instead only allows a key function. Therefore, if a comparison function is passed in, it will be converted to a key function automatically using `functools.cmp_to_key`.

Parameters

- **array** (*list*) – List to sort.
- **comparison** (*callable, optional*) – A custom comparison function used to sort the list. Function should accept two arguments and return a negative, zero, or position number depending on whether the first argument is considered smaller than, equal to, or larger than the second argument. Default is `None`. This argument is mutually exclusive with *key*.
- **key** (*callback, optional*) – A function of one argument used to extract a a comparison key from each list element. Default is `None`. This argument is mutually exclusive with *comparison*.
- **reverse** – Whether to reverse the sort. Default is `False`.

Returns Sorted list.

Return type list

Warning: *array* is modified in place.

New in version 2.2.0.

pydash.arrays.**sorted_index**(*array*, *value*, *callback=None*)
Determine the smallest index at which the value should be inserted into array in order to maintain the sort order of the sorted array. If callback is passed, it will be executed for value and each element in array to compute their sort ranking. The callback is invoked with one argument: `(value)`. If a property name is passed for callback, the created `pydash.collections.pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `pydash.collections.where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **array** (*list*) – List to inspect.
- **value** (*mixed*) – Value to evaluate.
- **callback** (*mixed, optional*) – Callback to determine sort key.

Returns Smallest index.

Return type int

New in version 1.0.0.

pydash.arrays.**sorted_last_index**(*array*, *value*, *callback=None*)

This method is like `sorted_index()` except that it returns the highest index at which a value should be inserted into a given sorted array in order to maintain the sort order of the array.

Parameters

- **array** (*list*) – List to inspect.
- **value** (*mixed*) – Value to evaluate.
- **callback** (*mixed, optional*) – Callback to determine sort key.

Returns Highest index.

Return type int

New in version 1.1.0.

pydash.arrays.**splice**(*array*, *index*, *how_many=None*, **items*)

Modify the contents of *array* by inserting elements starting at *index* and removing *how_many* number of elements after *index*.

Parameters

- **array** (*list*) – List to splice.
- **index** (*int*) – Index to splice at.
- **how_many** (*int, optional*) – Number of items to remove starting at *index*. If `None` then all items after *index* are removed. Default is `None`.
- **items** (*mixed*) – Elements to insert starting at *index*. Each item is inserted in the order given.

Returns The removed elements of *array*.

Return type list

Warning: *array* is modified in place.

New in version 2.2.0.

pydash.arrays.**split_at**(*array*, *index*)

Returns a list of two lists composed of the split of *array* at *index*.

Parameters

- **array** (*list*) – List to split.
- **index** (*int*) – Index to split at.

Returns Split list.

Return type list

New in version 2.0.0.

pydash.arrays.**tail**(*array*)

Return all but the first element of *array*.

Parameters **array** (*list*) – List to process.

Returns Rest of the list.

Return type list

See also:

- `rest()` (main definition)
- `tail()` (alias)

New in version 1.0.0.

`pydash.arrays.`**`take`**(*array*, *n*)

Creates a slice of *array* with *n* elements taken from the beginning.

Parameters

- **array** (*list*) – List to process.
- **n** (*int*) – Number of elements to take.

Returns Taken list.

Return type list

New in version 1.0.0.

Changed in version 1.1.0: Added `n` argument and removed as alias of `first()`.

`pydash.arrays.`**`take_right`**(*array*, *n*)

Creates a slice of *array* with *n* elements taken from the end.

Parameters

- **array** (*list*) – List to process.
- **n** (*int*) – Number of elements to take.

Returns Taken list.

Return type list

New in version 1.1.0.

`pydash.arrays.`**`take_right_while`**(*array*, *callback=None*)

Creates a slice of *array* with elements taken from the end. Elements are taken until the *callback* returns falsey. The *callback* is invoked with three arguments: `(value, index, array)`.

Parameters

- **array** (*list*) – List to process.
- **callback** (*mixed*) – Callback called per iteration

Returns Dropped list.

Return type list

New in version 1.1.0.

`pydash.arrays.`**`take_while`**(*array*, *callback=None*)

Creates a slice of *array* with elements taken from the beginning. Elements are taken until the *callback* returns falsey. The *callback* is invoked with three arguments: `(value, index, array)`.

Parameters

- **array** (*list*) – List to process.

- **callback** (*mixed*) – Callback called per iteration

Returns Taken list.

Return type list

New in version 1.1.0.

pydash.arrays.**union**(**arrays*)

Computes the union of the passed-in arrays.

Parameters **arrays** (*list*) – Lists to unionize.

Returns Unionized list.

Return type list

New in version 1.0.0.

pydash.arrays.**uniq**(*array*, *callback=None*)

Creates a duplicate-value-free version of the array. If callback is passed, each element of array is passed through a callback before uniqueness is computed. The callback is invoked with three arguments: `(value, index, array)`. If a property name is passed for callback, the created `pydash.collections.pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `pydash.collections.where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **array** (*list*) – List to process.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Unique list.

Return type list

See also:

- `uniq()` (main definition)
- `unique()` (alias)

New in version 1.0.0.

pydash.arrays.**unique**(*array*, *callback=None*)

Creates a duplicate-value-free version of the array. If callback is passed, each element of array is passed through a callback before uniqueness is computed. The callback is invoked with three arguments: `(value, index, array)`. If a property name is passed for callback, the created `pydash.collections.pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `pydash.collections.where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **array** (*list*) – List to process.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Unique list.

Return type list

See also:

- uniq() (main definition)
- unique() (alias)

New in version 1.0.0.

pydash.arrays.**without**(*array*, **values*)

Creates an array with all occurrences of the passed values removed.

Parameters

- **array** (*list*) – List to filter.
- **values** (*mixed*) – Values to remove.

Returns Filtered list.

Return type list

New in version 1.0.0.

pydash.arrays.**xor**(*array*, **lists*)

Creates a list that is the symmetric difference of the provided lists.

New in version 1.0.0.

pydash.arrays.**zip_**(**arrays*)

Groups the elements of each array at their corresponding indexes. Useful for separate data sources that are coordinated through matching array indexes.

Parameters **arrays** (*list*) – Lists to process.

Returns Zipped list.

Return type list

New in version 1.0.0.

pydash.arrays.**unshift**(*array*, **items*)

Insert the given elements at the beginning of *array* and return the modified list.

Parameters

- **array** (*list*) – List to modify.
- **items** (*mixed*) – Items to insert.

Returns Modified list.

Return type list

> **Warning:** *array* is modified in place.

New in version 2.2.0.

pydash.arrays.**unzip**(*array*)

The inverse of zip_(), this method splits groups of elements into lists composed of elements from each group at their corresponding indexes.

Parameters **array** (*list*) – List to process.

Returns Unzipped list.

Return type list

New in version 1.0.0.

pydash.arrays.**zip_object**(*keys*, *values=None*)

Creates a dict composed from lists of keys and values. Pass either a single two dimensional list, i.e. `[[key1, value1], [key2, value2]]`, or two lists, one of keys and one of corresponding values.

Parameters

- **keys** (*list*) – either a list of keys or a list of `[key, value]` pairs
- **values** (*list, optional*) – list of values to zip

Returns Zipped dict.

Return type dict

See also:

- `zip_object()` (main definition)
- `object_()` (alias)

New in version 1.0.0.

4.1.3 Chaining

Method chaining interface.

New in version 1.0.0.

pydash.chaining.**chain**(*value*)

Creates a `Chain` object which wraps the given value to enable intuitive method chaining. Chaining is lazy and won't compute a final value until `Chain.value()` is called.

Parameters **value** (*mixed*) – Value to initialize chain operations with.

Returns `Chain`: Instance of `Chain` initialized with *value*.

New in version 1.0.0.

Changed in version 2.0.0: Made chaining lazy.

pydash.chaining.**tap**(*value*, *interceptor*)

Invokes *interceptor* with the *value* as the first argument and then returns *value*. The purpose of this method is to "tap into" a method chain in order to perform operations on intermediate results within the chain.

Parameters

- **value** (*mixed*) – Current value of chain operation.
- **interceptor** (*function*) – Function called on *value*.

Returns *value* after *interceptor* call.

Return type mixed

New in version 1.0.0.

pydash.chaining.**thru**(*value*, *interceptor*)

Returns the result of calling *interceptor* on *value*. The purpose of this method is to pass *value* through a function during a method chain.

Parameters

- **value** (*mixed*) – Current value of chain operation.
- **interceptor** (*function*) – Function called with *value*.

Returns Results of `interceptor(value)`.

Return type mixed

New in version 2.0.0.

4.1.4 Collections

Functions that operate on lists and dicts.

New in version 1.0.0.

`pydash.collections.`**`all_`**`(`*collection*, *callback=None*`)`

Checks if the callback returns a truthy value for all elements of a collection. The callback is invoked with three arguments: `(value, index|key, collection)`. If a property name is passed for callback, the created `pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Whether all elements are truthy.

Return type bool

See also:

- `every()` (main definition)
- `all_()` (alias)

New in version 1.0.0.

`pydash.collections.`**`any_`**`(`*collection*, *callback=None*`)`

Checks if the callback returns a truthy value for any element of a collection. The callback is invoked with three arguments: `(value, index|key, collection)`. If a property name is passed for callback, the created `pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callbacked** (*mixed, optional*) – Callback applied per iteration.

Returns Whether any of the elements are truthy.

Return type bool

See also:

- `some()` (main definition)
- `any_()` (alias)

New in version 1.0.0.

pydash.collections.**at**(*collection*, **indexes*)

Creates a list of elements from the specified indexes, or keys, of the collection. Indexes may be specified as individual arguments or as arrays of indexes.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **indexes** (*mixed*) – The indexes of *collection* to retrieve, specified as individual indexes or arrays of indexes.

Returns filtered list

Return type list

New in version 1.0.0.

pydash.collections.**collect**(*collection*, *callback=None*)

Creates an array of values by running each element in the collection through the callback. The callback is invoked with three arguments: `(value, index|key, collection)`. If a property name is passed for callback, the created `pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Mapped list.

Return type list

See also:

- `map_()` (main definition)
- `collect()` (alias)

New in version 1.0.0.

pydash.collections.**contains**(*collection*, *target*, *from_index=0*)

Checks if a given value is present in a collection. If *from_index* is negative, it is used as the offset from the end of the collection.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **target** (*mixed*) – Target value to compare to.
- **from_index** (*int, optional*) – Offset to start search from.

Returns Whether *target* is in *collection*.

Return type bool

See also:

- `contains()` (main definition)
- `include()` (alias)

New in version 1.0.0.

pydash.collections.**count_by**(*collection*, *callback=None*)

Creates an object composed of keys generated from the results of running each element of *collection* through the callback.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Dict containing counts by key.

Return type dict

New in version 1.0.0.

pydash.collections.**deep_pluck**(*collection*, *path*)

Like pluck but works with deep paths.

Parameters

- **collection** (*list|dict*) – list of dicts
- **path** (*str|list*) – collection's path to pluck

Returns plucked list

Return type list

New in version 2.2.0.

pydash.collections.**detect**(*collection*, *callback=None*)

Iterates over elements of a collection, returning the first element that the callback returns truthy for.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns First element found or `None`.

Return type mixed

See also:

- `find()` (main definition)
- `detect()` (alias)
- `find_where()` (alias)

New in version 1.0.0.

pydash.collections.**each**(*collection*, *callback=None*)

Iterates over elements of a collection, executing the callback for each element.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns list|dict: *collection*

See also:

- `for_each()` (main definition)

- each() (alias)

New in version 1.0.0.

pydash.collections.**each_right**(*collection*, *callback*)

This method is like for_each() except that it iterates over elements of a *collection* from right to left.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns list|dict: *collection*

See also:

- for_each_right() (main definition)
- each_right() (alias)

New in version 1.0.0.

pydash.collections.**every**(*collection*, *callback=None*)

Checks if the callback returns a truthy value for all elements of a collection. The callback is invoked with three arguments: (value, index|key, collection). If a property name is passed for callback, the created pluck() style callback will return the property value of the given element. If an object is passed for callback, the created where() style callback will return True for elements that have the properties of the given object, else False.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Whether all elements are truthy.

Return type bool

See also:

- every() (main definition)
- all_() (alias)

New in version 1.0.0.

pydash.collections.**filter_**(*collection*, *callback=None*)

Iterates over elements of a collection, returning an list of all elements the callback returns truthy for.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Filtered list.

Return type list

See also:

- select() (main definition)

- filter_() (alias)

New in version 1.0.0.

pydash.collections.**find**(*collection*, *callback=None*)

Iterates over elements of a collection, returning the first element that the callback returns truthy for.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns First element found or None.

Return type mixed

See also:

- find() (main definition)
- detect() (alias)
- find_where() (alias)

New in version 1.0.0.

pydash.collections.**find_last**(*collection*, *callback=None*)

This method is like find() except that it iterates over elements of a *collection* from right to left.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Last element found or None.

Return type mixed

New in version 1.0.0.

pydash.collections.**find_where**(*collection*, *callback=None*)

Iterates over elements of a collection, returning the first element that the callback returns truthy for.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns First element found or None.

Return type mixed

See also:

- find() (main definition)
- detect() (alias)
- find_where() (alias)

New in version 1.0.0.

pydash.collections.**foldl**(*collection*, *callback=None*, *accumulator=None*)

Reduces a collection to a value which is the accumulated result of running each element in the collection through the callback, where each successive callback execution consumes the return value of the previous execution.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed*) – Callback applied per iteration.
- **accumulator** (*mixed, optional*) – Initial value of aggregator. Default is to use the result of the first iteration.

Returns Accumulator object containing results of reduction.

Return type mixed

See also:

- `reduce_()` (main definition)
- `foldl()` (alias)
- `inject()` (alias)

New in version 1.0.0.

pydash.collections.**foldr**(*collection*, *callback=None*, *accumulator=None*)

This method is like `reduce_()` except that it iterates over elements of a *collection* from right to left.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed*) – Callback applied per iteration.
- **accumulator** (*mixed, optional*) – Initial value of aggregator. Default is to use the result of the first iteration.

Returns Accumulator object containing results of reduction.

Return type mixed

See also:

- `reduce_right()` (main definition)
- `foldr()` (alias)

New in version 1.0.0.

pydash.collections.**for_each**(*collection*, *callback=None*)

Iterates over elements of a collection, executing the callback for each element.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns list|dict: *collection*

See also:

- `for_each()` (main definition)

- each() (alias)

New in version 1.0.0.

pydash.collections.**for_each_right**(*collection*, *callback*)

This method is like for_each() except that it iterates over elements of a *collection* from right to left.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns list|dict: *collection*

See also:

- for_each_right() (main definition)
- each_right() (alias)

New in version 1.0.0.

pydash.collections.**group_by**(*collection*, *callback=None*)

Creates an object composed of keys generated from the results of running each element of a *collection* through the callback.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Results of grouping by *callback*.

Return type dict

New in version 1.0.0.

pydash.collections.**include**(*collection*, *target*, *from_index=0*)

Checks if a given value is present in a collection. If *from_index* is negative, it is used as the offset from the end of the collection.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **target** (*mixed*) – Target value to compare to.
- **from_index** (*int, optional*) – Offset to start search from.

Returns Whether *target* is in *collection*.

Return type bool

See also:

- contains() (main definition)
- include() (alias)

New in version 1.0.0.

pydash.collections.**index_by**(*collection*, *callback=None*)

Creates an object composed of keys generated from the results of running each element of the collection through the given callback.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Results of indexing by *callback*.

Return type dict

New in version 1.0.0.

pydash.collections.**inject**(*collection*, *callback=None*, *accumulator=None*)

Reduces a collection to a value which is the accumulated result of running each element in the collection through the callback, where each successive callback execution consumes the return value of the previous execution.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed*) – Callback applied per iteration.
- **accumulator** (*mixed, optional*) – Initial value of aggregator. Default is to use the result of the first iteration.

Returns Accumulator object containing results of reduction.

Return type mixed

See also:

- `reduce_()` (main definition)
- `foldl()` (alias)
- `inject()` (alias)

New in version 1.0.0.

pydash.collections.**invoke**(*collection*, *method_name*, **args*, ***kargs*)

Invokes the method named by *method_name* on each element in the *collection* returning a list of the results of each invoked method.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **method_name** (*str*) – Name of method to invoke.
- **args** (*optional*) – Arguments to pass to method call.
- **kargs** (*optional*) – Keyword arguments to pass to method call.

Returns List of results of invoking method of each item.

Return type list

New in version 1.0.0.

pydash.collections.**map_**(*collection*, *callback=None*)

Creates an array of values by running each element in the collection through the callback. The callback is invoked with three arguments: `(value, index|key, collection)`. If a property name is passed for callback, the created `pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Mapped list.

Return type list

See also:

- `map_()` (main definition)
- `collect()` (alias)

New in version 1.0.0.

pydash.collections.**mapiter**(*collection*, *callback=None*)

Like `map_()` except returns a generator.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Each mapped item.

Return type generator

New in version 2.1.0.

pydash.collections.**max_**(*collection*, *callback=None*)

Retrieves the maximum value of a *collection*.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Maximum value.

Return type mixed

New in version 1.0.0.

pydash.collections.**min_**(*collection*, *callback=None*)

Retrieves the minimum value of a *collection*.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Minimum value.

Return type mixed

New in version 1.0.0.

pydash.collections.**partition**(*collection*, *callback=None*)

Creates an array of elements split into two groups, the first of which contains elements the *callback* returns truthy for, while the second of which contains elements the *callback* returns falsey for. The *callback* is invoked with three arguments: `(value, index|key, collection)`.

If a property name is provided for *callback* the created `pluck()` style callback returns the property value of the given element.

If an object is provided for *callback* the created `where()` style callback returns `True` for elements that have the properties of the given object, else `False`.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns List of grouped elements.

Return type list

New in version 1.1.0.

`pydash.collections.`**`pluck`**(*collection*, *key*)

Retrieves the value of a specified property from all elements in the collection.

Parameters

- **collection** (*list|dict*) – list of dicts
- **key** (*str*) – collection's key to pluck

Returns plucked list

Return type list

New in version 1.0.0.

`pydash.collections.`**`reduce_`**(*collection*, *callback=None*, *accumulator=None*)

Reduces a collection to a value which is the accumulated result of running each element in the collection through the callback, where each successive callback execution consumes the return value of the previous execution.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed*) – Callback applied per iteration.
- **accumulator** (*mixed, optional*) – Initial value of aggregator. Default is to use the result of the first iteration.

Returns Accumulator object containing results of reduction.

Return type mixed

See also:

- `reduce_()` (main definition)
- `foldl()` (alias)
- `inject()` (alias)

New in version 1.0.0.

`pydash.collections.`**`reduce_right`**(*collection*, *callback=None*, *accumulator=None*)

This method is like `reduce_()` except that it iterates over elements of a *collection* from right to left.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed*) – Callback applied per iteration.
- **accumulator** (*mixed, optional*) – Initial value of aggregator. Default is to use the result of the first iteration.

Returns Accumulator object containing results of reduction.

Return type mixed

See also:

- `reduce_right()` (main definition)
- `foldr()` (alias)

New in version 1.0.0.

`pydash.collections.`**`reductions`**(*collection*, *callback=None*, *accumulator=None*, *from_right=False*)

This function is like `reduce_()` except that it returns a list of every intermediate value in the reduction operation.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed*) – Callback applied per iteration.
- **accumulator** (*mixed, optional*) – Initial value of aggregator. Default is to use the result of the first iteration.

Returns Results of each reduction operation.

Return type list

Note: The last element of the returned list would be the result of using `reduce_()`.

New in version 2.0.0.

`pydash.collections.`**`reductions_right`**(*collection*, *callback=None*, *accumulator=None*)

This method is like `reductions()` except that it iterates over elements of a *collection* from right to left.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed*) – Callback applied per iteration.
- **accumulator** (*mixed, optional*) – Initial value of aggregator. Default is to use the result of the first iteration.

Returns Results of each reduction operation.

Return type list

Note: The last element of the returned list would be the result of using `reduce_()`.

New in version 2.0.0.

`pydash.collections.`**`reject`**(*collection*, *callback=None*)

The opposite of `filter_()` this method returns the elements of a collection that the callback does **not** return truthy for.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Rejected elements of *collection*.

Return type list

New in version 1.0.0.

pydash.collections.**sample**(*collection*, *n=None*)
Retrieves a random element or *n* random elements from a *collection*.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **n** (*int, optional*) – Number of random samples to return.

Returns list|mixed: List of sampled collection value if *n* is provided, else single value from collection if *n* is `None`.

New in version 1.0.0.

pydash.collections.**select**(*collection*, *callback=None*)
Iterates over elements of a collection, returning an list of all elements the callback returns truthy for.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Filtered list.

Return type list

See also:

- `select()` (main definition)
- `filter_()` (alias)

New in version 1.0.0.

pydash.collections.**shuffle**(*collection*)
Creates a list of shuffled values, using a version of the Fisher-Yates shuffle.

Parameters **collection** (*list|dict*) – Collection to iterate over.

Returns Shuffled list of values.

Return type list

New in version 1.0.0.

pydash.collections.**size**(*collection*)
Gets the size of the *collection* by returning *len(collection)* for iterable objects.

Parameters **collection** (*list|dict*) – Collection to iterate over.

Returns Collection length.

Return type int

New in version 1.0.0.

pydash.collections.**some**(*collection*, *callback=None*)
Checks if the callback returns a truthy value for any element of a collection. The callback is invoked with three arguments: `(value, index|key, collection)`. If a property name is passed for callback, the created `pluck()` style callback will return the property value of the given element. If an object is passed for callback, the created `where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callbacked** (*mixed, optional*) – Callback applied per iteration.

Returns Whether any of the elements are truthy.

Return type bool

See also:

- `some()` (main definition)
- `any_()` (alias)

New in version 1.0.0.

`pydash.collections.`**`sort_by`**(*collection*, *callback=None*)

Creates a list of elements, sorted in ascending order by the results of running each element in a *collection* through the callback.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Sorted list.

Return type list

New in version 1.0.0.

`pydash.collections.`**`to_list`**(*collection*)

Converts the collection to a list.

Parameters **collection** (*list|dict*) – Collection to iterate over.

Returns Collection converted to list.

Return type list

New in version 1.0.0.

`pydash.collections.`**`where`**(*collection*, *properties*)

Examines each element in a collection, returning an array of all elements that have the given properties.

Parameters

- **collection** (*list|dict*) – Collection to iterate over.
- **properties** (*dict*) – property values to filter by

Returns filtered list.

Return type list

New in version 1.0.0.

4.1.5 Functions

Functions that wrap other functions.

New in version 1.0.0.

pydash.functions.**after**(*n*, *func*)
Creates a function that executes *func*, with the arguments of the created function, only after being called *n* times.

Parameters

- **n** (*int*) – Number of times *func* must be called before it is executed.
- **func** (*function*) – Function to execute.

Returns Function wrapped in an `After` context.

Return type After

New in version 1.0.0.

pydash.functions.**before**(*n*, *func*)
Creates a function that executes *func*, with the arguments of the created function, until it has been called *n* times.

Parameters

- **n** (*int*) – Number of times *func* may be executed.
- **func** (*function*) – Function to execute.

Returns Function wrapped in an `Before` context.

Return type Before

New in version 1.1.0.

pydash.functions.**compose**(**funcs*)
This function is like `flow()` except that it creates a function that invokes the provided functions from right to left. For example, composing the functions `f()`, `g()`, and `h()` produces `f(g(h()))`.

Returns Function(s) wrapped in a `Compose` context.

Return type Compose

See also:

- `flow_right()` (main definition)
- `compose()` (alias)
- `pipe_right()` (alias)

New in version 1.0.0.

Changed in version 2.0.0: Added `flow_right()` and made `compose()` an alias.

Changed in version x.x.x: Added `pipe_right()` as alias.

pydash.functions.**conjoin**(**funcs*)
Creates a function that composes multiple predicate functions into a single predicate that tests whether **all** elements of an object pass each predicate.

Returns Function(s) wrapped in a `Conjoin` context.

Return type Conjoin

New in version 2.0.0.

pydash.functions.**curry**(*func*, *arity=None*)
Creates a function which accepts one or more arguments of *func* that when invoked either executes *func* returning its result, if all *func* arguments have been provided, or returns a function that accepts one or more of the remaining *func* arguments, and so on.

Parameters

- **func** (*function*) – Function to curry.
- **arity** (*int, optional*) – Number of function arguments that can be accepted by curried function. Default is to use the number of arguments that are accepted by *func*.

Returns Function wrapped in a `Curry` context.

Return type Curry

New in version 1.0.0.

`pydash.functions.`**`curry_right`**(*func*, *arity=None*)

This method is like `curry()` except that arguments are applied to *func* in the manner of `partial_right()` instead of `partial()`.

Parameters

- **func** (*function*) – Function to curry.
- **arity** (*int, optional*) – Number of function arguments that can be accepted by curried function. Default is to use the number of arguments that are accepted by *func*.

Returns Function wrapped in a `CurryRight` context.

Return type CurryRight

New in version 1.1.0.

`pydash.functions.`**`debounce`**(*func*, *wait*, *max_wait=False*)

Creates a function that will delay the execution of *func* until after *wait* milliseconds have elapsed since the last time it was invoked. Subsequent calls to the debounced function will return the result of the last *func* call.

Parameters

- **func** (*function*) – Function to execute.
- **wait** (*int*) – Milliseconds to wait before executing *func*.
- **max_wait** (*optional*) – Maximum time to wait before executing *func*.

Returns Function wrapped in a `Debounce` context.

Return type Debounce

New in version 1.0.0.

`pydash.functions.`**`delay`**(*func*, *wait*, **args*, ***kargs*)

Executes the *func* function after *wait* milliseconds. Additional arguments will be provided to *func* when it is invoked.

Parameters

- **func** (*function*) – Function to execute.
- **wait** (*int*) – Milliseconds to wait before executing *func*.

Returns Return from *func*.

Return type mixed

New in version 1.0.0.

`pydash.functions.`**`disjoin`**(**funcs*)

Creates a function that composes multiple predicate functions into a single predicate that tests whether **any** elements of an object pass each predicate.

Returns Function(s) wrapped in a `Disjoin` context.

Return type Disjoin

New in version 2.0.0.

`pydash.functions.`**`flow`**`(*funcs)`

Creates a function that is the composition of the provided functions, where each successive invocation is supplied the return value of the previous. For example, composing the functions `f()`, `g()`, and `h()` produces `h(g(f()))`.

Returns Function(s) wrapped in a `Compose` context.

Return type Compose

See also:

- `flow()` (main definition)
- `pipe()` (alias)

New in version 2.0.0.

Changed in version x.x.x: Added `pipe()` as alias.

`pydash.functions.`**`flow_right`**`(*funcs)`

This function is like `flow()` except that it creates a function that invokes the provided functions from right to left. For example, composing the functions `f()`, `g()`, and `h()` produces `f(g(h()))`.

Returns Function(s) wrapped in a `Compose` context.

Return type Compose

See also:

- `flow_right()` (main definition)
- `compose()` (alias)
- `pipe_right()` (alias)

New in version 1.0.0.

Changed in version 2.0.0: Added `flow_right()` and made `compose()` an alias.

Changed in version x.x.x: Added `pipe_right()` as alias.

`pydash.functions.`**`iterated`**`(func)`

Creates a function that is composed with itself. Each call to the iterated function uses the previous function call's result as input. Returned `Iterated` instance can be called with `(initial, n)` where *initial* is the initial value to seed *func* with and *n* is the number of times to call *func*.

Parameters **func** (*function*) – Function to iterate.

Returns Function wrapped in a `Iterated` context.

Return type Iterated

New in version 2.0.0.

`pydash.functions.`**`juxtapose`**`(*funcs)`

Creates a function whose return value is a list of the results of calling each *funcs* with the supplied arguments.

Returns Function wrapped in a `Juxtapose` context.

Return type Juxtapose

New in version 2.0.0.

pydash.functions.**negate**(*func*)

Creates a function that negates the result of the predicate *func*. The *func* function is executed with the arguments of the created function.

Parameters **func** (*function*) – Function to negate execute.

Returns Function wrapped in a `Negate` context.

Return type Negate

New in version 1.1.0.

pydash.functions.**once**(*func*)

Creates a function that is restricted to execute func once. Repeat calls to the function will return the value of the first call.

Parameters **func** (*function*) – Function to execute.

Returns Function wrapped in a `Once` context.

Return type Once

New in version 1.0.0.

pydash.functions.**partial**(*func*, **args*)

Creates a function that, when called, invokes *func* with any additional partial arguments prepended to those provided to the new function.

Parameters **func** (*function*) – Function to execute.

Returns Function wrapped in a `Partial` context.

Return type Partial

New in version 1.0.0.

pydash.functions.**partial_right**(*func*, **args*)

This method is like `partial()` except that partial arguments are appended to those provided to the new function.

Parameters **func** (*function*) – Function to execute.

Returns Function wrapped in a `Partial` context.

Return type Partial

New in version 1.0.0.

pydash.functions.**pipe**(**funcs*)

Creates a function that is the composition of the provided functions, where each successive invocation is supplied the return value of the previous. For example, composing the functions `f()`, `g()`, and `h()` produces `h(g(f()))`.

Returns Function(s) wrapped in a `Compose` context.

Return type Compose

See also:

- `flow()` (main definition)
- `pipe()` (alias)

New in version 2.0.0.

Changed in version x.x.x: Added `pipe()` as alias.

`pydash.functions.`**`pipe_right`**`(*funcs)`

This function is like `flow()` except that it creates a function that invokes the provided functions from right to left. For example, composing the functions `f()`, `g()`, and `h()` produces `f(g(h()))`.

Returns Function(s) wrapped in a `Compose` context.

Return type Compose

See also:

- `flow_right()` (main definition)
- `compose()` (alias)
- `pipe_right()` (alias)

New in version 1.0.0.

Changed in version 2.0.0: Added `flow_right()` and made `compose()` an alias.

Changed in version x.x.x: Added `pipe_right()` as alias.

`pydash.functions.`**`throttle`**`(func, wait)`

Creates a function that, when executed, will only call the *func* function at most once per every *wait* milliseconds. Subsequent calls to the throttled function will return the result of the last *func* call.

Parameters

- **func** (*function*) – Function to throttle.
- **wait** (*int*) – Milliseconds to wait before calling *func* again.

Returns Results of last *func* call.

Return type mixed

New in version 1.0.0.

`pydash.functions.`**`wrap`**`(value, func)`

Creates a function that provides value to the wrapper function as its first argument. Additional arguments provided to the function are appended to those provided to the wrapper function.

Parameters

- **value** (*mixed*) – Value provided as first argument to function call.
- **func** (*function*) – Function to execute.

Returns Function wrapped in a `Partial` context.

Return type Partial

New in version 1.0.0.

4.1.6 Numerical

Numerical/mathemetical related functions.

New in version 2.1.0.

pydash.numerical.**add**(*collection*, *callback=None*)

Sum each element in *collection*. If callback is passed, each element of *collection* is passed through a callback before the summation is computed.

Parameters

- **collection** (*list|dict*) – Collection to process.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Result of summation.

Return type number

See also:

- add() (main definition)
- sum_() (alias)

New in version 2.1.0.

pydash.numerical.**average**(*collection*, *callback=None*)

Calculate arithmetic mean of each element in *collection*. If callback is passed, each element of *collection* is passed through a callback before the mean is computed.

Parameters

- **collection** (*list|dict*) – Collection to process.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Result of mean.

Return type float

See also:

- average() (main definition)
- avg() (alias)
- mean() (alias)

New in version 2.1.0.

pydash.numerical.**avg**(*collection*, *callback=None*)

Calculate arithmetic mean of each element in *collection*. If callback is passed, each element of *collection* is passed through a callback before the mean is computed.

Parameters

- **collection** (*list|dict*) – Collection to process.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Result of mean.

Return type float

See also:

- average() (main definition)
- avg() (alias)

- `mean()` (alias)

New in version 2.1.0.

`pydash.numerical.`**`curve`**(*x*, *precision=0*)

Round number to precision.

Parameters

- **x** (*number*) – Number to round.
- **precision** (*int, optional*) – Rounding precision. Defaults to `0`.

Returns Rounded number.

Return type int

See also:

- `round_()` (main definition)
- `curve()` (alias)

New in version 2.1.0.

`pydash.numerical.`**`mean`**(*collection*, *callback=None*)

Calculate arithmetic mean of each element in *collection*. If callback is passed, each element of *collection* is passed through a callback before the mean is computed.

Parameters

- **collection** (*list|dict*) – Collection to process.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Result of mean.

Return type float

See also:

- `average()` (main definition)
- `avg()` (alias)
- `mean()` (alias)

New in version 2.1.0.

`pydash.numerical.`**`median`**(*collection*, *callback=None*)

Calculate median of each element in *collection*. If callback is passed, each element of *collection* is passed through a callback before the median is computed.

Parameters

- **collection** (*list|dict*) – Collection to process.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Result of median.

Return type float

New in version 2.1.0.

pydash.numerical.**moving_average**(*array*, *size*)

Calculate moving average of each element of *array*. If callback is passed, each element of *array* is passed through a callback before the moving average is computed.

Parameters

- **array** (*list*) – List to process.
- **size** (*int*) – Window size.

Returns Result of moving average.

Return type list

See also:

- moving_averge() (main definition)
- moving_avg() (alias)

New in version 2.1.0.

pydash.numerical.**moving_avg**(*array*, *size*)

Calculate moving average of each element of *array*. If callback is passed, each element of *array* is passed through a callback before the moving average is computed.

Parameters

- **array** (*list*) – List to process.
- **size** (*int*) – Window size.

Returns Result of moving average.

Return type list

See also:

- moving_averge() (main definition)
- moving_avg() (alias)

New in version 2.1.0.

pydash.numerical.**pow_**(*x*, *n*)

Calculate exponentiation of *x* raised to the *n* power.

Parameters

- **x** (*number*) – Base number.
- **n** (*number*) – Exponent.

Returns Result of calculation.

Return type number

See also:

- power() (main definition)
- pow_() (alias)

New in version 2.1.0.

pydash.numerical.**power**(*x*, *n*)
Calculate exponentiation of *x* raised to the *n* power.

Parameters

- **x** (*number*) – Base number.
- **n** (*number*) – Exponent.

Returns Result of calculation.

Return type number

See also:

- power() (main definition)
- pow_() (alias)

New in version 2.1.0.

pydash.numerical.**round_**(*x*, *precision=0*)
Round number to precision.

Parameters

- **x** (*number*) – Number to round.
- **precision** (*int, optional*) – Rounding precision. Defaults to 0.

Returns Rounded number.

Return type int

See also:

- round_() (main definition)
- curve() (alias)

New in version 2.1.0.

pydash.numerical.**scale**(*array*, *maximum=1*)
Scale list of value to a maximum number.

Parameters

- **array** (*list*) – Numbers to scale.
- **maximum** (*number*) – Maximum scale value.

Returns Scaled numbers.

Return type list

New in version 2.1.0.

pydash.numerical.**sigma**(*array*)
Calculate standard deviation of list of numbers.

Parameters **array** (*list*) – List to process.

Returns Calculated standard deviation.

Return type float

See also:

- std_deviation() (main definition)
- sigma() (alias)

New in version 2.1.0.

pydash.numerical.**slope**(*point1*, *point2*)
Calculate the slope between two points.

Parameters

- **point1** (*list|tuple*) – X and Y coordinates of first point.
- **point2** (*list|tuple*) – X and Y cooredinates of second point.

Returns Calculated slope.

Return type float

New in version 2.1.0.

pydash.numerical.**std_deviation**(*array*)
Calculate standard deviation of list of numbers.

Parameters **array** (*list*) – List to process.

Returns Calculated standard deviation.

Return type float

See also:

- std_deviation() (main definition)
- sigma() (alias)

New in version 2.1.0.

pydash.numerical.**sum_**(*collection*, *callback=None*)
Sum each element in *collection*. If callback is passed, each element of *collection* is passed through a callback before the summation is computed.

Parameters

- **collection** (*list|dict*) – Collection to process.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Result of summation.

Return type number

See also:

- add() (main definition)
- sum_() (alias)

New in version 2.1.0.

pydash.numerical.**transpose**(*array*)
Transpose the elements of *array*.

Parameters **array** (*list*) – List to process.

Returns Transposed list.

Return type list

New in version 2.1.0.

pydash.numerical.**variance**(*array*)
Calculate the variance of the elements in *array*.

Parameters **array** (*list*) – List to process.

Returns Calculated variance.

Return type float

New in version 2.1.0.

pydash.numerical.**zscore**(*collection*, *callback=None*)
Calculate the standard score assuming normal distribution. If callback is passed, each element of *collection* is passed through a callback before the standard score is computed.

Parameters

- **collection** (*list|dict*) – Collection to process.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Calculated standard score.

Return type float

New in version 2.1.0.

4.1.7 Objects

Functions that operate on lists, dicts, and other objects.

New in version 1.0.0.

pydash.objects.**assign**(*obj*, **sources*, ***kargs*)
Assigns own enumerable properties of source object(s) to the destination object. If *callback* is supplied, it is invoked with two arguments: `(obj_value, source_value)`.

Parameters

- **obj** (*dict*) – Destination object whose properties will be modified.
- **sources** (*dict*) – Source objects to assign to *obj*.

Keyword Arguments **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Modified *obj*.

Return type dict

Warning: *obj* is modified in place.

See also:

- `assign()` (main definition)
- `extend()` (alias)

New in version 1.0.0.

Changed in version 2.3.2: Apply `clone_deep()` to each *source* before assigning to *obj*.

pydash.objects.**callables**(*obj*)

Creates a sorted list of keys of an object that are callable.

Parameters **obj** (*list|dict*) – Object to inspect.

Returns All keys whose values are callable.

Return type list

See also:

- `functions()` (main definition)
- `methods()` (alias)

New in version 1.0.0.

Changed in version 2.0.0: Renamed `functions` to `callables`.

pydash.objects.**clone**(*value*, *is_deep=False*, *callback=None*)

Creates a clone of *value*. If *is_deep* is `True` nested valueects will also be cloned, otherwise they will be assigned by reference. If a callback is provided it will be executed to produce the cloned values. The callback is invoked with one argument: `(value)`.

Parameters

- **value** (*list|dict*) – Object to clone.
- **is_deep** (*bool, optional*) – Whether to perform deep clone.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns list|dict: Cloned object.

New in version 1.0.0.

pydash.objects.**clone_deep**(*value*, *callback=None*)

Creates a deep clone of *value*. If a callback is provided it will be executed to produce the cloned values. The callback is invoked with one argument: `(value)`.

Parameters

- **value** (*list|dict*) – Object to clone.
- **callback** (*mixed, optional*) – Callback applied per iteration.

Returns list|dict: Cloned object.

New in version 1.0.0.

pydash.objects.**deep_get**(*obj*, *path*)

Get the value at any depth of a nested object based on the path described by *path*. If path doesn't exist, `None` is returned.

Parameters

- **obj** (*list|dict*) – Object to process.
- **keys** (*str|list*) – List or `.` delimited string of keys describing path.

Returns Value of *obj* at path.

Return type mixed

New in version 2.2.0.

pydash.objects.**deep_has**(*obj*, *path*)
Checks if *path* exists as a key of *obj*.

Parameters

- **obj** (*mixed*) – Object to test.
- **path** (*mixed*) – Path to test for. Can be a list of nested keys or a `.` delimited string of path describing the path.

Returns Whether *obj* has *path*.

Return type bool

See also:

- `deep_has()` (main definition)
- `has_path()` (alias)

pydash.objects.**deep_set**(*obj*, *path*, *value*)
Sets the value of an object described by *path*. If any part of the object path doesn't exist, it will be created.

Parameters

- **obj** (*list|dict*) – Object to modify.
- **path** (*str | list*) – Target path to set value to.
- **value** (*mixed*) – Value to set.

Returns Modified *obj*.

Return type mixed

New in version 2.2.0.

pydash.objects.**deep_map_values**(*obj*, *callback=None*, *property_path=<pydash.helpers._NoValue object at 0x7fb0dc2c3ed0>*)
Map all non-object values in *obj* with return values from *callback*. The callback is invoked with two arguments: `(obj_value, property_path)` where `property_path` contains the list of path keys corresponding to the path of `obj_value`.

Parameters

- **obj** (*list|dict*) – Object to map.
- **callback** (*callable*) – Callback applied to each value.

Returns The modified object.

Return type mixed

Warning: *obj* is modified in place.

pydash.objects.**defaults**(*obj*, **sources*)
Assigns own enumerable properties of source object(s) to the destination object for all destination properties that resolve to undefined.

Parameters

- **obj** (*dict*) – Destination object whose properties will be modified.
- **sources** (*dict*) – Source objects to assign to *obj*.

Returns Modified *obj*.

Return type dict

Warning: *obj* is modified in place.

New in version 1.0.0.

pydash.objects.**extend**(*obj*, **sources*, ***kargs*)

Assigns own enumerable properties of source object(s) to the destination object. If *callback* is supplied, it is invoked with two arguments: `(obj_value, source_value)`.

Parameters

- **obj** (*dict*) – Destination object whose properties will be modified.
- **sources** (*dict*) – Source objects to assign to *obj*.

Keyword Arguments **callback** (*mixed, optional*) – Callback applied per iteration.

Returns Modified *obj*.

Return type dict

Warning: *obj* is modified in place.

See also:

- `assign()` (main definition)
- `extend()` (alias)

New in version 1.0.0.

Changed in version 2.3.2: Apply `clone_deep()` to each *source* before assigning to *obj*.

pydash.objects.**find_key**(*obj*, *callback=None*)

This method is like `pydash.arrays.find_index()` except that it returns the key of the first element that passes the callback check, instead of the element itself.

Parameters

- **obj** (*list|dict*) – Object to search.
- **callback** (*mixed*) – Callback applied per iteration.

Returns Found key or `None`.

Return type mixed

See also:

- `find_key()` (main definition)
- `find_last_key()` (alias)

New in version 1.0.0.

pydash.objects.**find_last_key**(*obj*, *callback=None*)

This method is like `pydash.arrays.find_index()` except that it returns the key of the first element that passes the callback check, instead of the element itself.

Parameters

- **obj** (*list|dict*) – Object to search.

- **callback** (*mixed*) – Callback applied per iteration.

Returns Found key or `None`.

Return type mixed

See also:

- `find_key()` (main definition)
- `find_last_key()` (alias)

New in version 1.0.0.

`pydash.objects.`**`for_in`**(*obj*, *callback=None*)

Iterates over own and inherited enumerable properties of *obj*, executing *callback* for each property.

Parameters

- **obj** (*list|dict*) – Object to process.
- **callback** (*mixed*) – Callback applied per iteration.

Returns list|dict: *obj*.

See also:

- `for_in()` (main definition)
- `for_own()` (alias)

New in version 1.0.0.

`pydash.objects.`**`for_in_right`**(*obj*, *callback=None*)

This function is like `for_in()` except it iterates over the properties in reverse order.

Parameters

- **obj** (*list|dict*) – Object to process.
- **callback** (*mixed*) – Callback applied per iteration.

Returns list|dict: *obj*.

See also:

- `for_in_right()` (main definition)
- `for_own_right()` (alias)

New in version 1.0.0.

`pydash.objects.`**`for_own`**(*obj*, *callback=None*)

Iterates over own and inherited enumerable properties of *obj*, executing *callback* for each property.

Parameters

- **obj** (*list|dict*) – Object to process.
- **callback** (*mixed*) – Callback applied per iteration.

Returns list|dict: *obj*.

See also:

- `for_in()` (main definition)

- `for_own()` (alias)

New in version 1.0.0.

`pydash.objects.`**`for_own_right`**(*obj*, *callback=None*)
This function is like `for_in()` except it iterates over the properties in reverse order.

Parameters

- **obj** (*list|dict*) – Object to process.
- **callback** (*mixed*) – Callback applied per iteration.

Returns list|dict: *obj*.

See also:

- `for_in_right()` (main definition)
- `for_own_right()` (alias)

New in version 1.0.0.

`pydash.objects.`**`get_path`**(*obj*, *path*, *default=None*)
Get the value at any depth of a nested object based on the path described by *path*. If path doesn't exist, `None` is returned.

Parameters

- **obj** (*list|dict*) – Object to process.
- **path** (*str|list*) – List or `.` delimited string of path describing path.

Returns Value of *obj* at path.

Return type mixed

New in version 2.0.0.

Changed in version 2.2.0: Support escaping "." delimiter in single string path key.

`pydash.objects.`**`has`**(*obj*, *key*)
Checks if *key* exists as a key of *obj*.

Parameters

- **obj** (*mixed*) – Object to test.
- **key** (*mixed*) – Key to test for.

Returns Whether *obj* has *key*.

Return type bool

New in version 1.0.0.

`pydash.objects.`**`has_path`**(*obj*, *path*)
Checks if *path* exists as a key of *obj*.

Parameters

- **obj** (*mixed*) – Object to test.
- **path** (*mixed*) – Path to test for. Can be a list of nested keys or a `.` delimited string of path describing the path.

Returns Whether *obj* has *path*.

Return type bool

See also:

- `deep_has()` (main definition)
- `has_path()` (alias)

`pydash.objects.`**`invert`**(*obj*, *multivalue=False*)

Creates an object composed of the inverted keys and values of the given object.

Parameters

- **obj** (*dict*) – dict to invert
- **multivalue** (*bool, optional*) – Whether to return inverted values as lists. Defaults to `False`.

Returns Inverted dict

Return type dict

Note: Assumes *dict* values are hashable as *dict* keys.

New in version 1.0.0.

Changed in version 2.0.0: Added `multivalue` argument.

`pydash.objects.`**`keys`**(*obj*)

Creates a list composed of the keys of *obj*.

Parameters **obj** (*mixed*) – Object to extract keys from.

Returns List of keys.

Return type list

See also:

- `keys()` (main definition)
- `keys_in()` (alias)

New in version 1.0.0.

Changed in version 1.1.0: Added `keys_in()` as alias.

`pydash.objects.`**`keys_in`**(*obj*)

Creates a list composed of the keys of *obj*.

Parameters **obj** (*mixed*) – Object to extract keys from.

Returns List of keys.

Return type list

See also:

- `keys()` (main definition)
- `keys_in()` (alias)

New in version 1.0.0.

Changed in version 1.1.0: Added `keys_in()` as alias.

pydash.objects.**map_values**(*obj*, *callback=None*)

Creates an object with the same keys as *obj* and values generated by running each property of *obj* through the *callback*. The callback is invoked with three arguments: `(value, key, object)`. If a property name is provided for *callback* the created `pydash.collections.pluck()` style callback will return the property value of the given element. If an object is provided for callback the created `pydash.collections.where()` style callback will return `True` for elements that have the properties of the given object, else `False`.

Parameters

- **obj** (*list|dict*) – Object to map.
- **callback** (*mixed*) – Callback applied per iteration.

Returns list|dict: Results of running *obj* through *callback*.

New in version 1.0.0.

pydash.objects.**merge**(*obj*, **sources*, ***kargs*)

Recursively merges own enumerable properties of the source object(s) that don't resolve to undefined into the destination object. Subsequent sources will overwrite property assignments of previous sources. If a callback is provided it will be executed to produce the merged values of the destination and source properties. If the callback returns undefined merging will be handled by the method instead. The callback is invoked with at least two arguments: `(obj_value, *source_value)`.

Parameters

- **obj** (*dict*) – Destination object to merge source(s) into.
- **sources** (*dict*) – Source objects to merge from. subsequent sources overwrite previous ones.

Keyword Arguments **callback** (*function, optional*) – Callback function to handle merging (must be passed in as keyword argument).

Returns Merged object.

Return type dict

Warning: *obj* is modified in place.

New in version 1.0.0.

Changed in version 2.3.2: Apply `clone_deep()` to each *source* before assigning to *obj*.

Changed in version 2.3.2: Allow *callback* to be passed by reference if it is the last positional argument.

pydash.objects.**methods**(*obj*)

Creates a sorted list of keys of an object that are callable.

Parameters **obj** (*list|dict*) – Object to inspect.

Returns All keys whose values are callable.

Return type list

See also:

- `functions()` (main definition)
- `methods()` (alias)

New in version 1.0.0.

Changed in version 2.0.0: Renamed `functions` to `callables`.

pydash.objects.**omit**(*obj*, *callback=None*, **properties*)

Creates a shallow clone of object excluding the specified properties. Property names may be specified as individual arguments or as lists of property names. If a callback is provided it will be executed for each property of object omitting the properties the callback returns truthy for. The callback is invoked with three arguments: `(value, key, object)`.

Parameters

- **obj** (*mixed*) – Object to process.
- **properties** (*str*) – Property values to omit.
- **callback** (*mixed, optional*) – Callback used to determine whic properties to omit.

Returns Results of omitting properties.

Return type dict

New in version 1.0.0.

pydash.objects.**pairs**(*obj*)

Creates a two dimensional list of an object's key-value pairs, i.e. `[[key1, value1], [key2, value2]]`.

Parameters **obj** (*mixed*) – Object to process.

Returns Two dimensional list of object's key-value pairs.

Return type list

New in version 1.0.0.

pydash.objects.**parse_int**(*value*, *radix=None*)

Converts the given *value* into an integer of the specified *radix*. If *radix* is falsey a radix of `10` is used unless the *value* is a hexadecimal, in which case a radix of 16 is used.

Parameters

- **value** (*mixed*) – Value to parse.
- **radix** (*int, optional*) – Base to convert to.

Returns Integer if parsable else `None`.

Return type mixed

New in version 1.0.0.

pydash.objects.**pick**(*obj*, *callback=None*, **properties*)

Creates a shallow clone of object composed of the specified properties. Property names may be specified as individual arguments or as lists of property names. If a callback is provided it will be executed for each property of object picking the properties the callback returns truthy for. The callback is invoked with three arguments: `(value, key, object)`.

Parameters

- **obj** (*list|dict*) – Object to pick from.
- **properties** (*str*) – Property values to pick.
- **callback** (*mixed, optional*) – Callback used to determine whic properties to pick.

Returns Results of picking properties.

Return type dict

New in version 1.0.0.

pydash.objects.**rename_keys**(*obj*, *key_map*)
Rename the keys of *obj* using *key_map* and return new object.

Parameters

- **obj** (*dict*) – Object to rename.
- **key_map** (*dict*) – Renaming map whose keys correspond to existing keys in *obj* and whose values are the new key name.

Returns Renamed *obj*.

Return type dict

New in version 2.0.0.

pydash.objects.**set_path**(*obj*, *value*, *keys*, *default=None*)
Sets the value of an object described by *keys*. If any part of the object path doesn't exist, it will be created with *default*.

Parameters

- **obj** (*list|dict*) – Object to modify.
- **value** (*mixed*) – Value to set.
- **keys** (*list*) – Target path to set value to.
- **default** (*callable*) – Callable that returns default value to assign if path part is not set. Defaults to `{}` is *obj* is a `dict` or `[]` if *obj* is a `list`.

Returns Modified *obj*.

Return type mixed

New in version 2.0.0.

pydash.objects.**to_string**(*obj*)
Converts an object to string.

Parameters **obj** (*mixed*) – Object to convert.

Returns String representation of *obj*.

Return type str

New in version 2.0.0.

pydash.objects.**transform**(*obj*, *callback=None*, *accumulator=None*)
An alernative to `pydash.collections.reduce()`, this method transforms *obj* to a new accumulator object which is the result of running each of its properties through a callback, with each callback execution potentially mutating the accumulator object. The callback is invoked with four arguments: `(accumulator, value, key, object)`. Callbacks may exit iteration early by explicitly returning `False`.

Parameters

- **obj** (*list|dict*) – Object to process.
- **callback** (*mixed*) – Callback applied per iteration.
- **accumulator** (*mixed, optional*) – Accumulated object. Defaults to `list`.

Returns Accumulated object.

Return type mixed

New in version 1.0.0.

pydash.objects.**update_path**(*obj*, *callback*, *keys*, *default=None*)

Update the value of an object described by *keys* using *callback*. If any part of the object path doesn't exist, it will be created with *default*. The callback is invoked with the last key value of *obj*: `(value)`

Parameters

- **obj** (*list|dict*) – Object to modify.
- **callback** (*callable*) – Function that returns updated value.
- **keys** (*list*) – A list of string keys that describe the object path to modify.
- **default** (*mixed*) – Default value to assign if path part is not set. Defaults to `{}` if *obj* is a `dict` or `[]` if *obj* is a `list`.

Returns Updated *obj*.

Return type mixed

New in version 2.0.0.

pydash.objects.**values**(*obj*)

Creates a list composed of the values of *obj*.

Parameters **obj** (*mixed*) – Object to extract values from.

Returns List of values.

Return type list

See also:

- `values()` (main definition)
- `values_in()` (alias)

New in version 1.0.0.

Changed in version 1.1.0: Added `values_in()` as alias.

pydash.objects.**values_in**(*obj*)

Creates a list composed of the values of *obj*.

Parameters **obj** (*mixed*) – Object to extract values from.

Returns List of values.

Return type list

See also:

- `values()` (main definition)
- `values_in()` (alias)

New in version 1.0.0.

Changed in version 1.1.0: Added `values_in()` as alias.

4.1.8 Predicates

Predicate functions that return boolean evaluations of objects.

New in version 2.0.0.

pydash.predicates.**is_associative**(*value*)
Checks if *value* is an associative object meaning that it can be accessed via an index or key

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is associative.

Return type bool

New in version 2.0.0.

pydash.predicates.**is_boolean**(*value*)
Checks if *value* is a boolean value.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is a boolean.

Return type bool

New in version 1.0.0.

pydash.predicates.**is_date**(*value*)
Check if *value* is a date object.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is a date object.

Return type bool

Note: This will also return `True` for datetime objects.

New in version 1.0.0.

pydash.predicates.**is_decreasing**(*value*)
Check if *value* is monotonically increasing.

Parameters **value** (*list*) – Value to check.

Returns Whether *value* is monotonically increasing.

Return type bool

New in version 2.0.0.

pydash.predicates.**is_empty**(*value*)
Checks if *value* is empty.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is empty.

Return type bool

Note: Returns `True` for booleans and numbers.

New in version 1.0.0.

pydash.predicates.**is_equal**(*a*, *b*, *callback=None*)
Performs a comparison between two values to determine if they are equivalent to each other. If a callback is provided it will be executed to compare values. If the callback returns `None`, comparisons will be handled by the method instead. The callback is invoked with two arguments: `(a, b)`.

Parameters

- **a** (*list|dict*) – Object to compare.
- **b** (*list|dict*) – Object to compare.
- **callback** (*mixed, optional*) – Callback used to compare values from *a* and *b*.

Returns Whether *a* and *b* are equal.

Return type bool

New in version 1.0.0.

pydash.predicates.**is_error**(*value*)
Checks if *value* is an `Exception`.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is an exception.

Return type bool

New in version 1.1.0.

pydash.predicates.**is_even**(*value*)
Checks if *value* is even.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is even.

Return type bool

New in version 2.0.0.

pydash.predicates.**is_float**(*value*)
Checks if *value* is a float.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is a float.

Return type bool

New in version 2.0.0.

pydash.predicates.**is_function**(*value*)
Checks if *value* is a function.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is callable.

Return type bool

New in version 1.0.0.

pydash.predicates.**is_increasing**(*value*)
Check if *value* is monotonically increasing.

Parameters **value** (*list*) – Value to check.

Returns Whether *value* is monotonically increasing.

Return type bool

New in version 2.0.0.

pydash.predicates.**is_indexed**(*value*)
Checks if *value* is integer indexed, i.e., `list` or `str`.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is integer indexed.

Return type bool

New in version 2.0.0.

pydash.predicates.**is_instance_of**(*value*, *types*)
Checks if *value* is an instance of *types*.

Parameters

- **value** (*mixed*) – Value to check.
- **types** (*mixed*) – Types to check against. Pass as `tuple` to check if *value* is one of multiple types.

Returns Whether *value* is an instance of *types*.

Return type bool

New in version 2.0.0.

pydash.predicates.**is_integer**(*value*)
Checks if *value* is a integer.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is a integer.

Return type bool

New in version 2.0.0.

pydash.predicates.**is_json**(*value*)
Checks if *value* is a valid JSON string.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is JSON.

Return type bool

New in version 2.0.0.

pydash.predicates.**is_list**(*value*)
Checks if *value* is a list.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is a list.

Return type bool

New in version 1.0.0.

pydash.predicates.**is_monotone**(*value*, *op*)
Checks if *value* is monotonic when *operator* used for comparison.

Parameters

- **value** (*list*) – Value to check.
- **op** (*function*) – Operation to used for comparison.

Returns Whether *value* is monotone.

Return type bool

New in version 2.0.0.

pydash.predicates.**is_nan**(*value*)
Checks if *value* is not a number.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is not a number.

Return type bool

New in version 1.0.0.

pydash.predicates.**is_negative**(*value*)
Checks if *value* is negative.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is negative.

Return type bool

New in version 2.0.0.

pydash.predicates.**is_none**(*value*)
Checks if *value* is *None*.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is `None`.

Return type bool

New in version 1.0.0.

pydash.predicates.**is_number**(*value*)
Checks if *value* is a number.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is a number.

Return type bool

Note: Returns `True` for `int`, `long` (PY2), `float`, and `decimal.Decimal`.

New in version 1.0.0.

pydash.predicates.**is_object**(*value*)
Checks if *value* is a `list` or `dict`.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is `list` or `dict`.

Return type bool

New in version 1.0.0.

pydash.predicates.**is_odd**(*value*)
Checks if *value* is odd.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is odd.

Return type bool

New in version 2.0.0.

pydash.predicates.**is_plain_object**(*value*)
Checks if *value* is a `dict`.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is a `dict`.

Return type bool

New in version 1.0.0.

pydash.predicates.**is_positive**(*value*)
Checks if *value* is positive.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is positive.

Return type bool

New in version 2.0.0.

pydash.predicates.**is_re**(*value*)
Checks if *value* is a `RegExp` object.

Parameters **value** (*mxied*) – Value to check.

Returns Whether *value* is a RegExp object.

Return type bool

See also:

- `is_reg_exp()` (main definition)
- `is_re()` (alias)

New in version 1.1.0.

pydash.predicates.**is_reg_exp**(*value*)
Checks if *value* is a `RegExp` object.

Parameters **value** (*mxied*) – Value to check.

Returns Whether *value* is a RegExp object.

Return type bool

See also:

- `is_reg_exp()` (main definition)
- `is_re()` (alias)

New in version 1.1.0.

pydash.predicates.**is_strictly_decreasing**(*value*)
Check if *value* is strictly decreasing.

Parameters **value** (*list*) – Value to check.

Returns Whether *value* is strictly decreasing.

Return type bool

New in version 2.0.0.

pydash.predicates.**is_strictly_increasing**(*value*)
Check if *value* is strictly increasing.

Parameters **value** (*list*) – Value to check.

Returns Whether *value* is strictly increasing.

Return type bool

New in version 2.0.0.

pydash.predicates.**is_string**(*value*)
Checks if *value* is a string.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is a string.

Return type bool

New in version 1.0.0.

pydash.predicates.**is_zero**(*value*)
Checks if *value* is `0`.

Parameters **value** (*mixed*) – Value to check.

Returns Whether *value* is `0`.

Return type bool

New in version 2.0.0.

4.1.9 Strings

String functions.

New in version 1.1.0.

pydash.strings.**camel_case**(*text*)
Converts *text* to camel case.

Parameters **text** (*str*) – String to convert.

Returns String converted to camel case.

Return type str

New in version 1.1.0.

pydash.strings.**capitalize**(*text*)
Capitalizes the first character of *text*.

Parameters **text** (*str*) – String to capitalize.

Returns Capitalized string.

Return type str

New in version 1.1.0.

pydash.strings.**deburr**(*text*)
Deburrs *text* by converting latin-1 supplementary letters to basic latin letters.

Parameters **text** (*str*) – String to deburr.

Returns Deburred string.

Return type str

New in version 2.0.0.

pydash.strings.**ends_with**(*text*, *target*, *position=None*)

Checks if *text* ends with a given target string.

Parameters

- **text** (*str*) – String to check.
- **target** (*str*) – String to check for.
- **position** (*int, optional*) – Position to search from. Defaults to end of *text*.

Returns Whether *text* ends with *target*.

Return type bool

New in version 1.1.0.

pydash.strings.**ensure_ends_with**(*text*, *suffix*)

Append a given suffix to a string, but only if the source string does not end with that suffix.

Parameters

- **text** (*str*) – Source string to append *suffix* to. Must not be *None*.
- **suffix** (*str*) – String to append to the source string if the source string does not end with *suffix* Must not be *None*.

Returns source string possibly extended by *suffix* Must not be *None*.

Return type str

New in version 2.4.0.

pydash.strings.**ensure_starts_with**(*text*, *prefix*)

Prepend a given prefix to a string, but only if the source string does not start with that prefix.

Parameters

- **text** (*str*) – Source string to prepend *prefix* to. Must not be *None*.
- **suffix** (*str*) – String to prepend to the source string if the source string does not start with *prefix*. Must not be *None*.

Returns source string possibly prefixed by *prefix*

Return type str

New in version 2.4.0.

pydash.strings.**escape**(*text*)

Converts the characters `&`, `<`, `>`, `"`, `'`, and `` ` `` in *text* to their corresponding HTML entities.

Parameters **text** (*str*) – String to escape.

Returns HTML escaped string.

Return type str

New in version 1.0.0.

Changed in version 1.1.0: Moved function to Strings module.

pydash.strings.**escape_reg_exp**(*text*)
Escapes the RegExp special characters in *text*.

Parameters **text** (*str*) – String to escape.

Returns RegExp escaped string.

Return type str

New in version 1.1.0.

pydash.strings.**escape_re**(*text*)
Escapes the RegExp special characters in *text*.

Parameters **text** (*str*) – String to escape.

Returns RegExp escaped string.

Return type str

New in version 1.1.0.

pydash.strings.**explode**(*text*, *delimiter=None*)
Splits *text* on *delimiter*. If *delimiter* not provided or `None`, then *text* is split on every character.

Parameters

- **text** (*str*) – String to explode.
- **delimiter** (*str, optional*) – Delimiter string to split on. Defaults to `None`.

Returns Exploded string.

Return type list

New in version 2.0.0.

pydash.strings.**implode**(*array*, *delimiter=''*)
Joins an iterable into a string using *delimiter* between each element.

Parameters

- **array** (*iterable*) – Iterable to implode.
- **delimiter** (*str*) – Delimiter to using when joining. Defaults to `''`.

Returns Imploded iterable.

Return type str

New in version 2.0.0.

pydash.strings.**js_match**(*reg_exp*, *text*)
Return list of matches using Javascript style regular expression.

Parameters

- **reg_exp** (*str*) – Javascript style regular expression.
- **text** (*str*) – String to evaluate.

Returns List of matches.

Return type list

New in version 2.0.0.

pydash.strings.**js_replace**(*reg_exp*, *text*, *repl*)
Replace *text* with *repl* using Javascript style regular expression to find matches.

Parameters

- **reg_exp** (*str*) – Javascript style regular expression.
- **text** (*str*) – String to evaluate.
- **repl** (*str*) – Replacement string.

Returns Modified string.

Return type str

New in version 2.0.0.

pydash.strings.**kebab_case**(*text*)

Converts *text* to kebab case (a.k.a. spinal case).

Parameters **text** (*str*) – String to convert.

Returns String converted to kebab case.

Return type str

New in version 1.1.0.

pydash.strings.**pad**(*text*, *length*, *chars=' '*)

Pads *text* on the left and right sides if it is shorter than the given padding length. The *chars* string may be truncated if the number of padding characters can't be evenly divided by the padding length.

Parameters

- **text** (*str*) – String to pad.
- **length** (*int*) – Amount to pad.
- **chars** (*str, optional*) – Chars to pad with. Defaults to `" "`.

Returns Padded string.

Return type str

New in version 1.1.0.

pydash.strings.**pad_left**(*text*, *length*, *chars=' '*)

Pads *text* on the left side if it is shorter than the given padding length. The *chars* string may be truncated if the number of padding characters can't be evenly divided by the padding length.

Parameters

- **text** (*str*) – String to pad.
- **length** (*int*) – Amount to pad.
- **chars** (*str, optional*) – Chars to pad with. Defaults to `" "`.

Returns Padded string.

Return type str

New in version 1.1.0.

pydash.strings.**pad_right**(*text*, *length*, *chars=' '*)

Pads *text* on the right side if it is shorter than the given padding length. The *chars* string may be truncated if the number of padding characters can't be evenly divided by the padding length.

Parameters

- **text** (*str*) – String to pad.

- **length** (*int*) – Amount to pad.
- **chars** (*str, optional*) – Chars to pad with. Defaults to `" "`.

Returns Padded string.

Return type str

New in version 1.1.0.

`pydash.strings.quote` (*text*, *quote_char=""*)
Quote a string with another string.

Parameters

- **text** (*str*) – String to be quoted
- **quote_char** (*str*) – the quote character. Defaults to *"*

Returns the quoted string.

Return type str

New in version 2.4.0.

`pydash.strings.repeat` (*text*, *n=0*)
Repeats the given string *n* times.

Parameters

- **text** (*str*) – String to repeat.
- **n** (*int, optional*) – Number of times to repeat the string.

Returns Repeated string.

Return type str

New in version 1.1.0.

`pydash.strings.snake_case` (*text*)
Converts *text* to snake case.

Parameters **text** (*str*) – String to convert.

Returns String converted to snake case.

Return type str

New in version 1.1.0.

`pydash.strings.starts_with` (*text*, *target*, *position=None*)
Checks if *text* starts with a given target string.

Parameters

- **text** (*str*) – String to check.
- **target** (*str*) – String to check for.
- **position** (*int, optional*) – Position to search from. Defaults to beginning of *text*.

Returns Whether *text* starts with *target*.

Return type bool

New in version 1.1.0.

`pydash.strings.surround` (*text*, *wrapper*)
Surround a string with another string.

Parameters

- **text** (*str*) – String to surround with *wrapper*
- **wrapper** (*str*) – String by which *text* is to be surrounded

Returns surrounded string.

Return type str

New in version 2.4.0.

pydash.strings.**trim**(*text*, *chars=None*)

Removes leading and trailing whitespace or specified characters from *text*.

Parameters

- **text** (*str*) – String to trim.
- **chars** (*str, optional*) – Specific characters to remove.

Returns Trimmed string.

Return type str

New in version 1.1.0.

pydash.strings.**trim_left**(*text*, *chars=None*)

Removes leading whitespace or specified characters from *text*.

Parameters

- **text** (*str*) – String to trim.
- **chars** (*str, optional*) – Specific characters to remove.

Returns Trimmed string.

Return type str

New in version 1.1.0.

pydash.strings.**trim_right**(*text*, *chars=None*)

Removes trailing whitespace or specified characters from *text*.

Parameters

- **text** (*str*) – String to trim.
- **chars** (*str, optional*) – Specific characters to remove.

Returns Trimmed string.

Return type str

New in version 1.1.0.

pydash.strings.**trunc**(*text*, *length=30*, *omission='...'*, *separator=None*)

Truncates *text* if it is longer than the given maximum string length. The last characters of the truncated string are replaced with the omission string which defaults to `...`.

Parameters

- **text** (*str*) – String to truncate.
- **length** (*int, optional*) – Maximum string length. Defaults to `30`.
- **omission** (*str, optional*) – String to indicate text is omitted.
- **separator** (*mixed, optional*) – Separator pattern to truncate to.

Returns Truncated string.

Return type str

New in version 1.1.0.

pydash.strings.**unescape**(*text*)
The inverse of escape(). This method converts the HTML entities &, <, >, ", ', and ` in *text* to their corresponding characters.

Parameters **text** (*str*) – String to unescape.

Returns HTML unescaped string.

Return type str

New in version 1.0.0.

Changed in version 1.1.0: Moved to Strings module.

pydash.strings.**url**(**paths*, ***params*)
Combines a series of URL paths into a single URL. Optionally, pass in keyword arguments to append query parameters.

Parameters **paths** (*str*) – URL paths to combine.

Keyword Arguments **params** (*str, optional*) – Query parameters.

Returns URL string.

Return type str

New in version 2.2.0.

pydash.strings.**words**(*text*)
Return list of words contained in *text*.

Parameters **text** (*str*) – String to split.

Returns List of words.

Return type list

New in version 2.0.0.

4.1.10 Utilities

Utility functions.

New in version 1.0.0.

pydash.utilities.**attempt**(*func*, **args*, ***kargs*)
Attempts to execute *func*, returning either the result or the caught error object.

Parameters **func** (*function*) – The function to attempt.

Returns Returns the *func* result or error object.

Return type mixed

New in version 1.1.0.

pydash.utilities.**constant**(*value*)
Creates a function that returns *value*.

Parameters **value** (*mixed*) – Constant value to return.

Returns Function that always returns *value*.

Return type function

New in version 1.0.0.

pydash.utilities.**callback**(*func*)

Return a pydash style callback. If *func* is a property name the created callback will return the property value for a given element. If *func* is an object the created callback will return `True` for elements that contain the equivalent object properties, otherwise it will return `False`.

Parameters **func** (*mixed*) – Object to create callback function from.

Returns Callback function.

Return type function

See also:

- `callback()` (main definition)
- `iteratee()` (alias)

New in version 1.0.0.

Changed in version 2.0.0: Rename `create_callback()` to `iteratee()`.

pydash.utilities.**deep_property**(*path*)

Creates a `pydash.collections.pluck()` style function, which returns the key value of a given object.

Parameters **key** (*mixed*) – Key value to fetch from object.

Returns Function that returns object's key value.

Return type function

See also:

- `property_()` (main definition)
- `prop()` (alias)

New in version 1.0.0.

pydash.utilities.**deep_prop**(*path*)

Creates a `pydash.collections.pluck()` style function, which returns the key value of a given object.

Parameters **key** (*mixed*) – Key value to fetch from object.

Returns Function that returns object's key value.

Return type function

See also:

- `property_()` (main definition)
- `prop()` (alias)

New in version 1.0.0.

pydash.utilities.**identity**(**args*)

Return the first argument provided to it.

Returns First argument or `None`.

Return type mixed

New in version 1.0.0.

pydash.utilities.**iteratee**(*func*)

Return a pydash style callback. If *func* is a property name the created callback will return the property value for a given element. If *func* is an object the created callback will return `True` for elements that contain the equivalent object properties, otherwise it will return `False`.

Parameters **func** (*mixed*) – Object to create callback function from.

Returns Callback function.

Return type function

See also:

- `callback()` (main definition)
- `iteratee()` (alias)

New in version 1.0.0.

Changed in version 2.0.0: Rename `create_callback()` to `iteratee()`.

pydash.utilities.**matches**(*source*)

Creates a `pydash.collections.where()` style predicate function which performs a deep comparison between a given object and the *source* object, returning `True` if the given object has equivalent property values, else `False`.

Parameters **source** (*dict*) – Source object used for comparision.

Returns Function that compares a `dict` to *source* and returns whether the two objects contain the same items.

Return type function

New in version 1.0.0.

pydash.utilities.**memoize**(*func*, *resolver=None*)

Creates a function that memoizes the result of *func*. If *resolver* is provided it will be used to determine the cache key for storing the result based on the arguments provided to the memoized function. By default, all arguments provided to the memoized function are used as the cache key. The result cache is exposed as the cache property on the memoized function.

Parameters

- **func** (*function*) – Function to memoize.
- **resolver** (*function, optional*) – Function that returns the cache key to use.

Returns Memoized function.

Return type function

New in version 1.0.0.

pydash.utilities.**noop**(**args*, ***kargs*)

A no-operation function.

New in version 1.0.0.

pydash.utilities.**now**()

Return the number of milliseconds that have elapsed since the Unix epoch (1 January 1970 00:00:00 UTC).

Returns Milliseconds since Unix epoch.

Return type int

New in version 1.0.0.

pydash.utilities.**property_**(*key*)

Creates a pydash.collections.pluck() style function, which returns the key value of a given object.

Parameters **key** (*mixed*) – Key value to fetch from object.

Returns Function that returns object's key value.

Return type function

See also:

- property_() (main definition)
- prop() (alias)

New in version 1.0.0.

pydash.utilities.**prop**(*key*)

Creates a pydash.collections.pluck() style function, which returns the key value of a given object.

Parameters **key** (*mixed*) – Key value to fetch from object.

Returns Function that returns object's key value.

Return type function

See also:

- property_() (main definition)
- prop() (alias)

New in version 1.0.0.

pydash.utilities.**random**(*start=0*, *stop=1*, *floating=False*)

Produces a random number between *start* and *stop* (inclusive). If only one argument is provided a number between 0 and the given number will be returned. If floating is truthy or either *start* or *stop* are floats a floating-point number will be returned instead of an integer.

Parameters

- **start** (*int*) – Minimum value.
- **stop** (*int*) – Maximum value.
- **floating** (*bool, optional*) – Whether to force random value to `float`. Default is `False`.

Returns int|float: Random value.

New in version 1.0.0.

pydash.utilities.**range_**(**args*)

Creates a list of numbers (positive and/or negative) progressing from start up to but not including end. If start is less than stop a zero-length range is created unless a negative step is specified.

Parameters

- **stop** (*int*) – Integer - 1 to stop at. Defaults to `1`.
- **start** (*int, optional*) – Integer to start with. Defaults to `0`.

- **step** (*int, optional*) – If positive the last element is the largest `start + i * step` less than *stop*. If negative the last element is the smallest `start + i * step` greater than *stop*. Defaults to `1`.

Returns List of integers in range

Return type list

New in version 1.0.0.

Changed in version 1.1.0: Moved to Utilities module.

`pydash.utilities.`**`result`**(*obj*, *key*, *default=None*)

Return the value of property *key* on *obj*. If *key* value is a function it will be invoked and its result returned, else the property value is returned. If *obj* is falsey then *default* is returned.

Parameters

- **obj** (*list|dict*) – Object to retrieve result from.
- **key** (*mixed*) – Key or index to get result from.
- **default** (*mixed, optional*) – Default value to return if *obj* is falsey. Defaults to `None`.

Returns Result of `obj[key]` or `None`.

Return type mixed

New in version 1.0.0.

Changed in version 2.0.0: Added `default` argument.

`pydash.utilities.`**`times`**(*n*, *callback*)

Executes the callback *n* times, returning a list of the results of each callback execution. The callback is invoked with one argument: `(index)`.

Parameters

- **n** (*int*) – Number of times to execute *callback*.
- **callback** (*function*) – Function to execute.

Returns A list of results from calling *callback*.

Return type list

New in version 1.0.0.

`pydash.utilities.`**`unique_id`**(*prefix=None*)

Generates a unique ID. If *prefix* is provided the ID will be appended to it.

Parameters **prefix** (*str, optional*) – String prefix to prepend to ID value.

Returns ID value.

Return type str

New in version 1.0.0.

CHAPTER 5

Project Info

5.1 License

5.2 Changelog

5.2.1 v2.4.2 (2015-02-03)

- Fix `remove` so that array is modified after callback iteration.

5.2.2 v2.4.1 (2015-01-11)

- Fix `kebab_case` so that it casts string to lower case.

5.2.3 v2.4.0 (2015-01-07)

- Add `ensure_ends_with`.
- Add `ensure_starts_with`.
- Add `quote`.

- Add `surround`.

5.2.4 v2.3.2 (2014-12-10)

- Fix `merge` and `assign`/`extend` so they apply `clone_deep` to source values before assigning to destination object.
- Make `merge` accept a callback as a positional argument if it is last.

5.2.5 v2.3.1 (2014-12-07)

- Add `pipe` and `pipe_right` as aliases of `flow` and `flow_right`.
- Fix `merge` so that trailing `{}` or `[]` don't overwrite previous source values.
- Make `py_` an alias for `_`.

5.2.6 v2.3.0 (2014-11-10)

- Support `type` callbacks (e.g. `int`, `float`, `str`, etc.) by only passing a single callback argument when invoking the callback.
- Drop official support for Python 3.2. Too many testing dependencies no longer work on it.

5.2.7 v2.2.0 (2014-10-28)

- Add `append`.
- Add `deep_get`.
- Add `deep_has`.
- Add `deep_map_values`.
- Add `deep_set`.
- Add `deep_pluck`.
- Add `deep_property`.
- Add `join`.
- Add `pop`.
- Add `push`.
- Add `reverse`.
- Add `shift`.
- Add `sort`.
- Add `splice`.
- Add `unshift`.
- Add `url`.
- Fix bug in `snake_case` that resulted in returned string not being converted to lower case.
- Fix bug in chaining method access test which skipped the actual test.

- Make _ instance alias method access to methods with a trailing underscore in their name. For example, `_.map()` becomes an alias for `map_()`.
- Make `deep_prop` an alias of `deep_property`.
- Make `has` work with deep paths.
- Make `has_path` an alias of `deep_has`.
- Make `get_path` handle escaping the `.` delimiter for string keys.
- Make `get_path` handle list indexing using strings such as `'0.1.2'` to access `'value'` in `[[0, [0, 0, 'value']]]`.
- Make `concat` an alias of `cat`.

5.2.8 v2.1.0 (2014-09-17)

- Add `add`, `sum_`.
- Add `average`, `avg`, `mean`.
- Add `mapiter`.
- Add `median`.
- Add `moving_average`, `moving_avg`.
- Add `power`, `pow_`.
- Add `round_`, `curve`.
- Add `scale`.
- Add `slope`.
- Add `std_deviation`, `sigma`.
- Add `transpose`.
- Add `variance`.
- Add `zscore`.

5.2.9 v2.0.0 (2014-09-11)

- Add _ instance that supports both method chaining and module method calling.
- Add `cat`.
- Add `conjoin`.
- Add `deburr`.
- Add `disjoin`.
- Add `explode`.
- Add `flatten_deep`.
- Add `flow`.
- Add `flow_right`.
- Add `get_path`.

- Add `has_path`.
- Add `implode`.
- Add `intercalate`.
- Add `interleave`.
- Add `intersperse`.
- Add `is_associative`.
- Add `is_even`.
- Add `is_float`.
- Add `is_decreasing`.
- Add `is_increasing`.
- Add `is_indexed`.
- Add `is_instance_of`.
- Add `is_integer`.
- Add `is_json`.
- Add `is_monotone`.
- Add `is_negative`.
- Add `is_odd`.
- Add `is_positive`.
- Add `is_strictly_decreasing`.
- Add `is_strictly_increasing`.
- Add `is_zero`.
- Add `iterated`.
- Add `js_match`.
- Add `js_replace`.
- Add `juxtapose`.
- Add `mapcat`.
- Add `reductions`.
- Add `reductions_right`.
- Add `rename_keys`.
- Add `set_path`.
- Add `split_at`.
- Add `thru`.
- Add `to_string`.
- Add `update_path`.
- Add `words`.

- Make callback function calling adapt to argspec of given callback function. If, for example, the full callback signature is `(item, index, obj)` but the passed in callback only supports `(item)`, then only `item` will be passed in when callback is invoked. Previously, callbacks had to support all arguments or implement star-args.
- Make `chain` lazy and only compute the final value when `value` called.
- Make `compose` an alias of `flow_right`.
- Make `flatten` shallow by default, remove callback option, and add `is_deep` option. (**breaking change**)
- Make `is_number` return `False` for boolean `True` and `False`. (**breaking change**)
- Make `invert` accept `multivalue` argument.
- Make `result` accept `default` argument.
- Make `slice_` accept optional `start` and `end` arguments.
- Move files in `pydash/api/` to `pydash/`. (**possible breaking change**)
- Move predicate functions from `pydash.api.objects` to `pydash.api.predicates`. (**possible breaking change**)
- Rename `create_callback` to `iteratee`. (**breaking change**)
- Rename `functions` to `callables` in order to allow `functions.py` to exist at the root of the pydash module folder. (**breaking change**)
- Rename *private* utility function `_iter_callback` to `itercallback`. (**possible breaking change**)
- Rename *private* utility function `_iter_list_callback` to `iterlist_callback`. (**possible breaking change**)
- Rename *private* utility function `_iter_dict_callback` to `iterdict_callback`. (**possible breaking change**)
- Rename *private* utility function `_iterate` to `iterator`. (**possible breaking change**)
- Rename *private* utility function `_iter_dict` to `iterdict`. (**possible breaking change**)
- Rename *private* utility function `_iter_list` to `iterlist`. (**possible breaking change**)
- Rename *private* utility function `_iter_unique` to `iterunique`. (**possible breaking change**)
- Rename *private* utility function `_get_item` to `getitem`. (**possible breaking change**)
- Rename *private* utility function `_set_item` to `setitem`. (**possible breaking change**)
- Rename *private* utility function `_deprecated` to `deprecated`. (**possible breaking change**)
- Undeprecate `tail` and make alias of `rest`.

5.2.10 v1.1.0 (2014-08-19)

- Add `attempt`.
- Add `before`.
- Add `camel_case`.
- Add `capitalize`.
- Add `chunk`.
- Add `curry_right`.
- Add `drop_right`.

- Add `drop_right_while`.
- Add `drop_while`.
- Add `ends_with`.
- Add `escape_reg_exp` and `escape_re`.
- Add `is_error`.
- Add `is_reg_exp` and `is_re`.
- Add `kebab_case`.
- Add `keys_in` as alias of `keys`.
- Add `negate`.
- Add `pad`.
- Add `pad_left`.
- Add `pad_right`.
- Add `partition`.
- Add `pull_at`.
- Add `repeat`.
- Add `slice_`.
- Add `snake_case`.
- Add `sorted_last_index`.
- Add `starts_with`.
- Add `take_right`.
- Add `take_right_while`.
- Add `take_while`.
- Add `trim`.
- Add `trim_left`.
- Add `trim_right`.
- Add `trunc`.
- Add `values_in` as alias of `values`.
- Create `pydash.api.strings` module.
- Deprecate `tail`.
- Modify `drop` to accept `n` argument and remove as alias of `rest`.
- Modify `take` to accept `n` argument and remove as alias of `first`.
- Move `escape` and `unescape` from `pydash.api.utilities` to `pydash.api.strings`. (**possible breaking change**)
- Move `range_` from `pydash.api.arrays` to `pydash.api.utilities`. (**possible breaking change**)

5.2.11 v1.0.0 (2014-08-05)

- Add Python 2.6 and Python 3 support.
- Add `after`.
- Add `assign` and `extend`. Thanks nathancahill!
- Add `callback` and `create_callback`.
- Add `chain`.
- Add `clone`.
- Add `clone_deep`.
- Add `compose`.
- Add `constant`.
- Add `count_by`. Thanks nathancahill!
- Add `curry`.
- Add `debounce`.
- Add `defaults`. Thanks nathancahill!
- Add `delay`.
- Add `escape`.
- Add `find_key`. Thanks nathancahill!
- Add `find_last`. Thanks nathancahill!
- Add `find_last_index`. Thanks nathancahill!
- Add `find_last_key`. Thanks nathancahill!
- Add `for_each`. Thanks nathancahill!
- Add `for_each_right`. Thanks nathancahill!
- Add `for_in`. Thanks nathancahill!
- Add `for_in_right`. Thanks nathancahill!
- Add `for_own`. Thanks nathancahill!
- Add `for_own_right`. Thanks nathancahill!
- Add `functions_` and `methods`. Thanks nathancahill!
- Add `group_by`. Thanks nathancahill!
- Add `has`. Thanks nathancahill!
- Add `index_by`. Thanks nathancahill!
- Add `identity`.
- Add `inject`.
- Add `invert`.
- Add `invoke`. Thanks nathancahill!
- Add `is_list`. Thanks nathancahill!
- Add `is_boolean`. Thanks nathancahill!

- Add `is_empty`. Thanks nathancahill!
- Add `is_equal`.
- Add `is_function`. Thanks nathancahill!
- Add `is_none`. Thanks nathancahill!
- Add `is_number`. Thanks nathancahill!
- Add `is_object`.
- Add `is_plain_object`.
- Add `is_string`. Thanks nathancahill!
- Add `keys`.
- Add `map_values`.
- Add `matches`.
- Add `max_`. Thanks nathancahill!
- Add `memoize`.
- Add `merge`.
- Add `min_`. Thanks nathancahill!
- Add `noop`.
- Add `now`.
- Add `omit`.
- Add `once`.
- Add `pairs`.
- Add `parse_int`.
- Add `partial`.
- Add `partial_right`.
- Add `pick`.
- Add `property_` and `prop`.
- Add `pull`. Thanks nathancahill!
- Add `random`.
- Add `reduce_` and `foldl`.
- Add `reduce_right` and `foldr`.
- Add `reject`. Thanks nathancahill!
- Add `remove`.
- Add `result`.
- Add `sample`.
- Add `shuffle`.
- Add `size`.
- Add `sort_by`. Thanks nathancahill!

- Add `tap`.
- Add `throttle`.
- Add `times`.
- Add `transform`.
- Add `to_list`. Thanks nathancahill!
- Add `unescape`.
- Add `unique_id`.
- Add `values`.
- Add `wrap`.
- Add `xor`.

5.2.12 v0.0.0 (2014-07-22)

- Add `all_`.
- Add `any_`.
- Add `at`.
- Add `bisect_left`.
- Add `collect`.
- Add `collections`.
- Add `compact`.
- Add `contains`.
- Add `detect`.
- Add `difference`.
- Add `drop`.
- Add `each`.
- Add `each_right`.
- Add `every`.
- Add `filter_`.
- Add `find`.
- Add `find_index`.
- Add `find_where`.
- Add `first`.
- Add `flatten`.
- Add `head`.
- Add `include`.
- Add `index_of`.
- Add `initial`.

- Add `intersection`.
- Add `last`.
- Add `last_index_of`.
- Add `map_`.
- Add `object_`.
- Add `pluck`.
- Add `range_`.
- Add `rest`.
- Add `select`.
- Add `some`.
- Add `sorted_index`.
- Add `tail`.
- Add `take`.
- Add `union`.
- Add `uniq`.
- Add `unique`.
- Add `unzip`.
- Add `where`.
- Add `without`.
- Add `zip_`.
- Add `zip_object`.

5.3 Authors

5.3.1 Lead

- Derrick Gilland, dgilland@gmail.com, dgilland@github

5.3.2 Contributors

- Nathan Cahill, nathan@nathancahill.com, nathancahill@github
- Klaus Sevensleeper, k7sleeper@gmail.com, k7sleeper@github

5.4 How to Contribute

- Overview
- Guidelines
- Branching

5.4.1 Overview

1. Fork the repo.
2. Build development environment run tests to ensure a clean, working slate.
3. Improve/fix the code.
4. Add test cases if new functionality introduced or bug fixed (100% test coverage).
5. Ensure tests pass.
6. Add yourself to `AUTHORS.rst`.
7. Push to your fork and submit a pull request to the `develop` branch.

5.4.2 Guidelines

Some simple guidelines to follow when contributing code:

- Adhere to PEP8.
- Clean, well documented code.
- All tests must pass.
- 100% test coverage.

5.4.3 Branching

There are two main development branches: `master` and `develop`. `master` represents the currently released version while `develop` is the latest development work. When submitting a pull request, be sure to submit to `develop`. The originating branch you submit from can be any name though.

5.4.4 Continuous Integration

Integration testing is provided by Travis-CI at https://travis-ci.org/dgilland/pydash.

Test coverage reporting is provided by Coveralls at https://coveralls.io/r/dgilland/pydash.

5.4.5 Project CLI

Some useful CLI commands when working on the project are below. **NOTE:** All commands are run from the root of the project and require `make`.

make build

Run the `clean` and `install` commands.

```
make build
```

make install

Install Python dependencies into virtualenv located at `env/`.

```
make install
```

make clean

Remove build/test related temporary files like `env/`, `.tox`, `.coverage`, and `__pycache__`.

```
make clean
```

make test

Run unittests under the virtualenv's default Python version. Does not test all support Python versions. To test all supported versions, see make test-full.

```
make test
```

make test-full

Run unittest and linting for all supported Python versions. **NOTE:** This will fail if you do not have all Python versions installed on your system. If you are on an Ubuntu based system, the Dead Snakes PPA is a good resource for easily installing multiple Python versions. If for whatever reason you're unable to have all Python versions on your development machine, note that Travis-CI will run full integration tests on all pull requests.

```
make test-full
```

make lint

Run `make pylint` and `make pep8` commands.

```
make lint
```

make pylint

Run `pylint` compliance check on code base.

```
make pylint
```

make pep8

Run PEP8 compliance check on code base.

```
make pep8
```

make docs

Build documentation to `docs/_build/`.

```
make docs
```

5.5 Kudos

Thank you to Lo-Dash for providing such a great library to port.

CHAPTER 6

Indices and Tables

- *genindex*
- *modindex*
- *search*

p

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z