

---

# **vine Documentation**

***Release 0.9.0***

**Ask Solem  
Contributors**

March 02, 2016



<b>1</b>	<b>About</b>	<b>3</b>
<b>2</b>	<b>Contents</b>	<b>5</b>
2.1	API Reference . . . . .	5
2.2	Changes . . . . .	8
2.3	1.0.0 . . . . .	8
<b>3</b>	<b>Indices and tables</b>	<b>9</b>
	<b>Python Module Index</b>	<b>11</b>



**Version** 1.0.0rc1

**Web** <http://vine.readthedocs.org/>

**Download** <http://pypi.python.org/pypi/vine/>

**Source** <http://github.com/celery/vine/>

**Keywords** promise, async, future



---

**About**

---





## 2.1 API Reference

**Release** 0.9

**Date** March 02, 2016

### 2.1.1 vine.promises

**class** vine.promises.**promise** (*fun=None, args=None, kwargs=None, callback=None, on\_error=None*)

Future evaluation.

This is a special implementation of promises in that it can be used both for “promise of a value” and lazy evaluation. The biggest upside for this is that everything in a promise can also be a promise, e.g. filters, callbacks and errbacks can all be promises.

Usage examples:

```
>>> from __future__ import print_statement # noqa
>>> p = promise()
>>> p.then(promise(print, ('OK',))) # noqa
>>> p.on_error = promise(print, ('ERROR',)) # noqa
>>> p(20)
OK, 20
>>> p.then(promise(print, ('hello',))) # noqa
hello, 20

>>> p.throw(KeyError('foo'))
ERROR, KeyError('foo')

>>> p2 = promise()
>>> p2.then(print) # noqa
>>> p2.cancel()
>>> p(30)
```

Example:

```
from vine import promise, wrap

class Protocol(object):
```

```
def __init__(self):
    self.buffer = []

def receive_message(self):
    return self.read_header().then(
        self.read_body).then(
            wrap(self.prepare_body))

def read(self, size, callback=None):
    callback = callback or promise()
    tell_eventloop_to_read(size, callback)
    return callback

def read_header(self, callback=None):
    return self.read(4, callback)

def read_body(self, header, callback=None):
    body_size, = unpack('>L', header)
    return self.read(body_size, callback)

def prepare_body(self, value):
    self.buffer.append(value)
```

**args**  
**cancel()**  
**cancelled**  
**failed**  
**fun**  
**kwargs**  
**listeners**  
**on\_error**  
**ready**  
**reason**  
**set\_error\_state** (*exc=None*)  
**then** (*callback, on\_error=None*)  
**throw** (*exc=None*)  
**throw1** (*exc*)  
**value**

### 2.1.2 vine.synchronization

**class** vine.synchronization.**barrier** (*promises=None, args=None, kwargs=None, callback=None, size=None*)

Synchronization primitive to call a callback after a list of promises have been fulfilled.

Example:

```

# Request supports the .then() method.
p1 = http.Request('http://a')
p2 = http.Request('http://b')
p3 = http.Request('http://c')
requests = [p1, p2, p3]

def all_done():
    pass # all requests complete

b = barrier(requests).then(all_done)

# oops, we forgot we want another request
b.add(http.Request('http://d'))

```

Note that you cannot add new promises to a barrier after the barrier is fulfilled.

```

add(p)

add_noincr(p)

cancel()

finalize()

then(callback, errback=None)

throw(*args, **kwargs)

throw1(*args, **kwargs)

```

### 2.1.3 vine.funtools

```

vine.funtools.maybe_promise(p)
vine.funtools.ensure_promise(p)
vine.funtools.ppartial(p, *args, **kwargs)
vine.funtools.preplace(p, *args, **kwargs)
vine.funtools.ready_promise(callback=None, *args)
vine.funtools.starpromise(fun, *args, **kwargs)
vine.funtools.transform(filter_, callback, *filter_args, **filter_kwargs)
    Filter final argument to a promise.

```

E.g. to coerce callback argument to int:

```
transform(int, callback)
```

or a more complex example extracting something from a dict and coercing the value to float:

```

def filter_key_value(key, filter_, mapping):
    return filter_(mapping[key])

def get_page_expires(self, url, callback=None):
    return self.request(
        'GET', url,
        callback=transform(get_key, callback, 'PageExpireValue', int),
    )

```

`vine.functools.wrap(p)`

Wrap promise so that if the promise is called with a promise as argument, we attach ourselves to that promise instead.

## 2.1.4 vine.abstract

`class vine.abstract.Thenable`

`cancel()`

`then(on_success, on_error=None)`

`throw(exc=None)`

## 2.1.5 vine.five

- `vine.five`

### vine.five

Compatibility implementations of features only available in newer Python versions.

`vine.five.items(seq)`

`vine.five.with_metaclass(Type, skip_attrs=set([u'__dict__', u'__weakref__']))`

Class decorator to set metaclass.

Works with both Python 2 and Python 3 and it does not add an extra class in the lookup order like `six.with_metaclass` does (that is – it copies the original class instead of using inheritance).

## 2.2 Changes

## 2.3 1.0.0

**release-date** TBA

**release-by**

---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`



## V

`vine.abstract`, 8  
`vine.five`, 8  
`vine.funtools`, 7  
`vine.promises`, 5  
`vine.synchronization`, 6





## A

add() (vine.synchronization.barrier method), 7  
add\_noincr() (vine.synchronization.barrier method), 7  
args (vine.promises.promise attribute), 6

## B

barrier (class in vine.synchronization), 6

## C

cancel() (vine.abstract.Thenable method), 8  
cancel() (vine.promises.promise method), 6  
cancel() (vine.synchronization.barrier method), 7  
cancelled (vine.promises.promise attribute), 6

## E

ensure\_promise() (in module vine.funtools), 7

## F

failed (vine.promises.promise attribute), 6  
finalize() (vine.synchronization.barrier method), 7  
fun (vine.promises.promise attribute), 6

## I

items() (in module vine.five), 8

## K

kwargs (vine.promises.promise attribute), 6

## L

listeners (vine.promises.promise attribute), 6

## M

maybe\_promise() (in module vine.funtools), 7

## O

on\_error (vine.promises.promise attribute), 6

## P

ppartial() (in module vine.funtools), 7

preplace() (in module vine.funtools), 7

promise (class in vine.promises), 5

## R

ready (vine.promises.promise attribute), 6  
ready\_promise() (in module vine.funtools), 7  
reason (vine.promises.promise attribute), 6

## S

set\_error\_state() (vine.promises.promise method), 6  
starpromise() (in module vine.funtools), 7

## T

then() (vine.abstract.Thenable method), 8  
then() (vine.promises.promise method), 6  
then() (vine.synchronization.barrier method), 7  
Thenable (class in vine.abstract), 8  
throw() (vine.abstract.Thenable method), 8  
throw() (vine.promises.promise method), 6  
throw() (vine.synchronization.barrier method), 7  
throw1() (vine.promises.promise method), 6  
throw1() (vine.synchronization.barrier method), 7  
transform() (in module vine.funtools), 7

## V

value (vine.promises.promise attribute), 6  
vine.abstract (module), 8  
vine.five (module), 8  
vine.funtools (module), 7  
vine.promises (module), 5  
vine.synchronization (module), 6

## W

with\_metaclass() (in module vine.five), 8  
wrap() (in module vine.funtools), 7