
Python

Apr 17, 2020

Contents

1	conf	3
1.1	Module Contents	3
2	setup	5
2.1	Module Contents	5
3	conftest	7
4	wtypes	9
4.1	Subpackages	9
4.2	Submodules	9
4.3	Package Contents	30
	Python Module Index	47
	Index	49

This page contains auto-generated API reference documentation¹.

¹ Created with sphinx-autoapi

1.1 Module Contents

```
conf.master_doc = index
conf.extensions
conf.autoapi_type = python
conf.autoapi_dirs = ['.']
conf.SUFFIX
conf.html_static_path = []
conf.using_rtd_theme
conf.theme
conf.websupport2_base_url = https://readthedocs.org/websupport
conf.context
conf.html_context
conf.extensions = ['readthedocs_ext.readthedocs']
conf.project_language = en
conf.language_user
conf.latex_engine_user
conf.latex_elements_user
conf.latex_use_xindy = False
conf.chinese
conf.japanese
conf.latex_engine
```

setup

2.1 Module Contents

`setup.flit_data`

`setup.metadata`

CHAPTER 3

conftest

extended python types for the web and json

Notes

`wtypes.simpleTypes`

The limited set of base types provide by jsonschema.

Type list

4.1 Subpackages

4.1.1 `wtypes.widgets`

Widgets features for wtypes.

Submodules

`wtypes.widgets.ipynbwidgets`

4.2 Submodules

4.2.1 `wtypes.base`

An extended type and trait system for python.

Notes

Module Contents

```
wtypes.base.__version__ = 0.0.1
wtypes.base.ValidationError
class wtypes.base._Implementation
    An implementation of the pluggy wtypes spec.
```

Notes

The implementation needs to be registered with the plugin manager.

validate_type (*type*)

validate_object (*object*, *schema*)

```
wtypes.base.istype (object, cls)
    instance(object, type) and issubclass(object, cls)
```

Examples

```
>>> assert istype(int, int)
>>> assert not istype(10, int)
```

```
class wtypes.base._NoTitle
    A subclass suppresses the class name when combining schema
```

```
class wtypes.base._NoInit
    A subclass to restrict initializing an object from the type.
```

```
class wtypes.base._ContextMeta
    Bases: abc.ABCMeta
    Meta operations for the context..
    _context
        The schema the object validates against.
        Type dict
    _type
        The schema the object validates against.
        Type type
    _schema
        Type dict
```

Notes

A context type cannot be verified as it only describes, although some descriptors like SHACL can validate

_schema

_context

`__type`

`__merge_args` (*cls*)

`__merge_context` (*cls*)

`__merge_annotations` (*cls*)
Merge annotations from the module resolution order.

`__merge_schema` (*cls*)
Merge schema from the module resolution order.

`__merge_types` (*cls*)
Merge schema from the module resolution order.

`__matmul__` (*cls, object*)

`validate` (*cls, object*)
A context type does not validate.

`__instancecheck__` (*cls, object*)

`create` (*cls, name: str, **schema*)
Create a new schema type.

Parameters

- **`name`** (*str*) – The title of the new type/schema
- **`**schema`** (*dict*) – Extra features to add include in the schema.

Returns

Return type `type`

`__add__` (*cls, object*)

`__neg__` (*cls*)
The Not version of a type.

`__pos__` (*cls*)
The type.

`__and__` (*cls, object*)
AllOf the conditions

`__sub__` (*cls, object*)
AnyOf the conditions

`__or__` (*cls, object*)
OneOf the conditions

`class wtypes.base._SchemaMeta`

Bases: `wtypes.base._ContextMeta`

Meta operations for wtypes.

The `_SchemaMeta` ensures that a type's extended schema is validate. Types cannot be generated with invalid schema.

`__schema`
The schema the object validates against.

Type `dict`

`__neg__` (*cls*)
The Not version of a type.

`__pos__ (cls)`
The type.

`__and__ (cls, object)`
AllOf the conditions

`__sub__ (cls, object)`
AnyOf the conditions

`__or__ (cls, object)`
OneOf the conditions

`validate (cls, object)`
Validate an object against type's schema.

Note: `isinstance` can used for validation, too.

Parameters `object` – An object to validate.

Raises `jsonschema.ValidationError` – The `jsonschema` module validation throws an exception on failure, otherwise the returns a `None` type.

class `wtypes.base._ConstType`
Bases: `wtypes.base._SchemaMeta`
`ConstType` permits bracketed syntax for defining complex types.

Note: The bracketed notebook should differentiate actions on types versus those on objects.

`__getitem__ (cls, object)`

class `wtypes.base._ContainerType`
Bases: `wtypes.base._ConstType`
`ContainerType` extras schema from bracketed arguments to define complex types.

`__getitem__ (cls, object)`

`wtypes.base._python_to_wtype (object)`

`wtypes.base._get_schema_from_typeish (object, key='anyOf')`
infer a schema from an object.

`wtypes.base._lower_key (str)`

`wtypes.base._object_to_webtype (object)`

`wtypes.base._construct_title (cls)`

class `wtypes.base.Trait`
A trait is an object validated by a validate `jsonschema`.

`_schema`

`_context`

classmethod `_resolve_defaults (cls, *args, **kwargs)`

`wtypes.base.get_jawn (thing, key, object)`

class wtypes.base.**Description**

Bases: *wtypes.base._NoInit*, *wtypes.base.Trait*, *wtypes.base._NoTitle*

An empty type with a description

Examples

```
>>> yo = Description['yo']
>>> yo._schema.toDict()
{'description': 'yo'}
```

class wtypes.base.**Examples**

Bases: *wtypes.base._NoInit*, *wtypes.base.Trait*

class wtypes.base.**Default**

Bases: *wtypes.base._NoInit*, *wtypes.base.Trait*

class wtypes.base.**Title**

Bases: *wtypes.base._NoInit*, *wtypes.base.Trait*, *wtypes.base._NoTitle*

An empty type with a title

Examples

```
>>> holla = Title['holla']
>>> holla._schema.toDict()
{'title': 'holla'}
```

class wtypes.base.**Const**

Bases: *wtypes.base._NoInit*, *wtypes.base.Trait*

A constant

Examples

```
>>> Const[10]._schema.toDict()
{'const': 10}
```

```
>>> assert isinstance('thing', Const['thing'])
>>> assert not isinstance('jawn', Const['thing']), "Because the compiler is from_
↪Philly."
```

class wtypes.base.**Bool**

Bases: *wtypes.base.Trait*

Boolean type

Examples

```
>>> Bool(), Bool(True), Bool(False)
(False, True, False)
>>> assert (Bool + Default[True])()
```

Note: It is not possible to base class `bool` so object creation is customized.

class `wtypes.base.Null`
Bases: `wtypes.base.Trait`
nil, none, null type

Examples

```
>>> Null(None)
>>> assert (Null + Default[None]()) is None
```

class `wtypes.base._NumericSchema`
Bases: `wtypes.base._SchemaMeta`
Meta operations for numerical types

- `__rgt__`
- `__rge__`
- `__rlt__`
- `__rle__`
- `__ge__` (*cls, object*)
Inclusive minimum
- `__gt__` (*cls, object*)
Exclusive minimum
- `__le__` (*cls, object*)
Inclusive maximum
- `__lt__` (*cls, object*)
Exclusive maximum
- `__truediv__` (*cls, object*)
multiple of a number

class `wtypes.base.Integer`
Bases: `wtypes.base.Trait`, `int`
integer type

Examples

```
>>> assert isinstance(10, Integer)
>>> assert not isinstance(10.1, Integer)
>>> (Integer+Default[9])(9)
9
```

```
>>> bounded = (10< Integer)< 100
>>> bounded._schema.toDict()
{'type': 'integer', 'exclusiveMinimum': 10, 'exclusiveMaximum': 100}
>>> assert isinstance(12, bounded)
```

(continues on next page)

(continued from previous page)

```
>>> assert not isinstance(0, bounded)
>>> assert (Integer/3)(9) == 9
```

class `wtypes.base.Float`

Bases: `wtypes.base.Trait`, `float`

float type

```
>>> assert isinstance(10, Float)
>>> assert isinstance(10.1, Float)
```

Symbollic conditions.

```
>>> bounded = (10< Float)< 100
>>> bounded._schema.toDict()
{'type': 'number', 'exclusiveMinimum': 10, 'exclusiveMaximum': 100}
```

```
>>> assert isinstance(12.1, bounded)
>>> assert not isinstance(0.1, bounded)
```

Multiples

```
>>> assert (Float+MultipleOf[3])(9) == 9
```

class `wtypes.base.MultipleOf`

Bases: `wtypes.base._NoInit`, `wtypes.base.Trait`

A multipleof constraint for numeric types.

class `wtypes.base.Minimum`

Bases: `wtypes.base._NoInit`, `wtypes.base.Trait`

A minimum constraint for numeric types.

class `wtypes.base.ExclusiveMinimum`

Bases: `wtypes.base._NoInit`, `wtypes.base.Trait`

A exclusive minimum constraint for numeric types.

class `wtypes.base.Maximum`

Bases: `wtypes.base._NoInit`, `wtypes.base.Trait`

A exclusive maximum constraint for numeric types.

class `wtypes.base.ExclusiveMaximum`

Bases: `wtypes.base._NoInit`, `wtypes.base.Trait`

A exclusive maximum constraint for numeric types.

class `wtypes.base.Properties`

Bases: `wtypes.base.Trait`, `wtypes.base._NoInit`, `wtypes.base._NoTitle`

Object properties.

class `wtypes.base._ObjectSchema`

Bases: `wtypes.base._SchemaMeta`

Meta operations for the object schema.

`__getitem__` (*cls*, *object*)

Examples

```
>>> Dict[wtypes.Forward[range], int].__annotations__
{'': typing.Union[abc.Forward, int]}
>>> Dict[wtypes.Forward[range], int]._schema.toDict()
{'type': 'object', 'properties': {}, 'additionalProperties': {'anyOf': [{'type':
↪ 'integer'}]}}
```

class wtypes.base._Object

Base class for validating object types.

classmethod __init_subclass__(cls, **schema)

classmethod from_config_file(cls, *object)

class wtypes.base.Dict

Bases: *wtypes.base.Trait*, dict, *wtypes.base._Object*

dict type

Examples

```
>>> assert isinstance(Dict, __import__('collections').abc.MutableMapping)
>>> assert (Dict + Default[{'b': 'foo'}])() == {'b': 'foo'}
>>> assert (Dict + Default[{'b': 'foo'}])({'a': 'bar'}) == {'b': 'foo', 'a': 'bar'
↪ }
```

```
>>> assert isinstance({}, Dict)
>>> assert not isinstance([], Dict)
```

```
>>> assert isinstance({'a': 1}, Dict + Required['a',])
>>> assert not isinstance({}, Dict + Required['a',])
```

```
>>> assert not isinstance({'a': 'b'}, Dict[Integer, Float])
>>> assert Dict[Integer]({'a': 1}) == {'a': 1}
```

```
>>> Dict[{'a': int}]._schema.toDict()
{'type': 'object', 'properties': {'a': {'type': 'integer'}}}
>>> Dict[{'a': int}]({'a': 1})
{'a': 1}
```

__setitem__ (self, key, object)

Only test the key being set to avoid invalid state.

update (self, *args, **kwargs)

class wtypes.base.Bunch

Bases: *wtypes.base.Dict*, *munch.Munch*

Bunch type

Examples

```
>>> Bunch[{'a': int}]._schema.toDict()
{'type': 'object', 'properties': {'a': {'type': 'integer'}}}
>>> Bunch[{'a': int}]({'a': 1}).toDict()
{'a': 1}
```

class wtypes.base.**AdditionalProperties**

Bases: *wtypes.base.Trait*, *wtypes.base._NoInit*, *wtypes.base._NoTitle*

Additional object properties.

class wtypes.base.**Required**

Bases: *wtypes.base.Trait*, *wtypes.base._NoInit*, *wtypes.base._NoTitle*

Required properties.

class wtypes.base.**minProperties**

Bases: *wtypes.base.Trait*, *wtypes.base._NoInit*, *wtypes.base._NoTitle*

Minimum number of properties.

class wtypes.base.**maxProperties**

Bases: *wtypes.base.Trait*, *wtypes.base._NoInit*, *wtypes.base._NoTitle*

Maximum number of properties.

class wtypes.base.**PropertyNames**

Bases: *wtypes.base.Trait*, *wtypes.base._NoInit*, *wtypes.base._NoTitle*

Property name constraints.

class wtypes.base.**Dependencies**

Bases: *wtypes.base.Trait*, *wtypes.base._NoInit*, *wtypes.base._NoTitle*

Properties dependencies.

class wtypes.base.**PatternProperties**

Bases: *wtypes.base.Trait*, *wtypes.base._NoInit*, *wtypes.base._NoTitle*

Pattern properties names.

class wtypes.base.**_StringSchema**

Bases: *wtypes.base._SchemaMeta*

Meta operations for strings types.

__mod__ (*cls*, *object*)

A pattern string type.

__gt__ (*cls*, *object*)

Minumum string length

__lt__ (*cls*, *object*)

Maximum string length

class wtypes.base.**String**

Bases: *wtypes.base.Trait*, *str*

string type.

Examples

```
>>> assert isinstance('abc', String)
>>> assert (String+Default['abc'])() == 'abc'
```

String patterns

```
>>> assert isinstance('abc', String%"^a")
>>> assert not isinstance('abc', String%"^b")
```

String constraints

```
>>> assert isinstance('abc', (2<String)<10)
>>> assert not isinstance('a', (2<String)<10)
>>> assert not isinstance('a'*100, (2<String)<10)
```

class `wtypes.base.MinLength`

Bases: `wtypes.base.Trait`, `wtypes.base._NoInit`, `wtypes.base._NoTitle`

Minimum length of a string type.

class `wtypes.base.MaxLength`

Bases: `wtypes.base.Trait`, `wtypes.base._NoInit`, `wtypes.base._NoTitle`

Maximum length of a string type.

class `wtypes.base.ContentMediaType`

Bases: `wtypes.base.Trait`, `wtypes.base._NoInit`, `wtypes.base._NoTitle`

Content type of a string.

class `wtypes.base.Pattern`

Bases: `wtypes.base.Trait`, `wtypes.base._NoInit`

A regular expression pattern.

class `wtypes.base.Format`

Bases: `wtypes.base.Trait`, `wtypes.base._NoInit`, `wtypes.base._NoTitle`

class `wtypes.base._ListSchema`

Bases: `wtypes.base._SchemaMeta`

Meta operations for list types.

__getitem__ (*cls, object*)

List meta operations for bracketed type notation.

Examples

```
>>> List[wtypes.Forward[range], int]._type
typing.List[typing.Union[abc.Forward, int]]
```

```
>>> Tuple[wtypes.Forward[range], int]._type
typing.Tuple[abc.Forward, int]
```

__gt__ (*cls, object*)

Minimum array length

__lt__ (*cls, object*)

Maximum array length

```
class wtypes.base.List
    Bases: wtypes.base.Trait, list
    List type
```

Examples

List

```
>>> assert isinstance([], List)
>>> assert not isinstance({}, List)
```

Typed list

```
>>> assert List[Integer]([1, 2, 3])
>>> assert isinstance([1], List[Integer])
>>> assert not isinstance([1.1], List[Integer])
```

```
>>> List[Integer, String]._schema.toDict()
{'type': 'array', 'items': {'anyOf': [{'type': 'integer'}, {'type': 'string'}]}}
```

Tuple

```
>>> assert List[Integer, String]([1, 'abc', 2])
>>> assert isinstance([1, '1'], List[Integer, String])
>>> assert not isinstance([1, {}], List[Integer, String])
```

```
_verify_item(self, object, id=None)
    Elemental verification for interactive type checking.
```

```
__setitem__(self, id, object)
```

```
append(self, object)
```

```
insert(self, id, object)
```

```
extend(self, object)
```

```
pop(self, index=-1)
```

```
class wtypes.base.Unique
    Bases: wtypes.base.List
    Unique list type
```

Examples

```
>>> assert Unique(list('abc'))
>>> assert isinstance([1,2], Unique)
>>> assert not isinstance([1,1], Unique)
```

```
class wtypes.base.Tuple
    Bases: wtypes.base.List
    tuple type
```

Note: There are no tuples in json, they are typed lists.

```
>>> assert Tuple.__schema == List.__schema
```

Examples

```
>>> assert isinstance([1,2], Tuple)
>>> assert Tuple[Integer, String]([1, 'abc'])
>>> Tuple[Integer, String].__schema.toDict()
{'type': 'array', 'items': [{'type': 'integer'}, {'type': 'string'}]}
```

```
>>> assert isinstance([1,'1'], Tuple[Integer, String])
>>> assert not isinstance([1,1], Tuple[Integer, String])
```

class wtypes.base.UniqueItems

Bases: *wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle*

Schema for unique items in a list.

class wtypes.base.Contains

Bases: *wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle*

class wtypes.base.Items

Bases: *wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle*

class wtypes.base.AdditionalItems

Bases: *wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle*

class wtypes.base.MinItems

Bases: *wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle*

Minimum length of an array.

class wtypes.base.MaxItems

Bases: *wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle*

Maximum length of an array.

class wtypes.base.Enum

Bases: *wtypes.base.Trait*

An enumerate type that is restricted to its inputs.

Examples

```
>>> assert Enum['cat', 'dog']('cat')
>>> assert isinstance('cat', Enum['cat', 'dog'])
>>> assert not isinstance('', Enum['cat', 'dog'])
```

class wtypes.base.ContentEncoding

Bases: *Enum['7bit 8bit binary quoted-printable base64'.split()], wtypes.base._NoInit, wtypes.base._NoTitle*

Content encodings for a string.

class wtypes.base.If

Bases: *wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle*

if condition type

```

class wtypes.base.Then
    Bases: wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle
    then condition type

class wtypes.base.Else
    Bases: wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle
    else condition type

```

4.2.2 wtypes.combining_types

Module Contents

```

class wtypes.combining_types._NotType
    Bases: wtypes.base._ConstType
    validate(cls, object)
    __getitem__(cls, object)

class wtypes.combining_types.Not
    Bases: wtypes.base.Trait
    not schema.

```

Examples

```

>>> assert Not[wtypes.String](100) == 100
>>> assert not isinstance('abc', Not[wtypes.String])

```

Note: See the `__neg__` method for symbollic not composition.

```

class wtypes.combining_types._AnyOfType
    Bases: wtypes.base._ConstType
    validate(cls, object)
    __getitem__(cls, object)

class wtypes.combining_types.AnyOf
    Bases: wtypes.base.Trait
    anyOf combined schema.

```

Examples

```

>>> assert AnyOf[wtypes.Integer, wtypes.String]('abc')
>>> assert isinstance(10, AnyOf[wtypes.Integer, wtypes.String])
>>> assert not isinstance([], AnyOf[wtypes.Integer, wtypes.String])

```

```

class wtypes.combining_types._AllOfType
    Bases: wtypes.base._ConstType
    validate(cls, object)

```

```
__getitem__(cls, object)
```

```
class wtypes.combining_types.AllOf
```

```
Bases: wtypes.base.Trait
```

```
allOf combined schema.
```

Examples

```
>>> assert AllOf[wtypes.Float>0, wtypes.Integer/3](9)
>>> assert isinstance(9, AllOf[wtypes.Float>0, wtypes.Integer/3])
>>> assert not isinstance(-9, AllOf[wtypes.Float>0, wtypes.Integer/3])
```

```
class wtypes.combining_types._OneOfType
```

```
Bases: wtypes.base._ConstType
```

```
validate(cls, object)
```

```
__getitem__(cls, object)
```

```
class wtypes.combining_types.OneOf
```

```
Bases: wtypes.base.Trait
```

```
oneOf combined schema.
```

Examples

```
>>> assert OneOf[wtypes.Float>0, wtypes.Integer/3](-9)
>>> assert isinstance(-9, OneOf[wtypes.Float>0, wtypes.Integer/3])
>>> assert not isinstance(9, OneOf[wtypes.Float>0, wtypes.Integer/3])
```

4.2.3 wtypes.content_types

Module Contents

```
class wtypes.content_types._Content
```

```
Bases: wtypes.String
```

```
__repr_data__(self)
```

```
__repr_metadata__(self)
```

```
__repr_mimebundle__(self, include=None, exclude=None)
```

```
class wtypes.content_types.TextPlain
```

```
Bases: wtypes.content_types._Content
```

```
class wtypes.content_types.TextMarkdown
```

```
Bases: wtypes.content_types.TextPlain
```

```
class wtypes.content_types.TextHtml
```

```
Bases: wtypes.content_types.TextPlain
```

4.2.4 wtypes.dataclass

Compatibility for wtyped dataclasses.

Module Contents

class `wtypes.dataclass.Setter`

`__setattr__` (*self, key, object*)
Only test the attribute being set to avoid invalid state.

class `wtypes.dataclass.DataClass`

Bases: `wtypes.dataclass.Setter`, `wtypes.Trait`, `wtypes.base._Object`

Validating dataclass type

Examples

```
>>> class q(DataClass): a: int
>>> q._schema.toDict()
{'type': 'object', 'properties': {'a': {'type': 'integer'}}, 'required': ['a']}
```

```
>>> q(a=10)
q(a=10)
```

```
>>> assert not isinstance({}, q)
```

classmethod `__init_subclass__` (*cls, **kwargs*)

4.2.5 wtypes.evented

Evented wtypes, the observable pattern

Module Contents

class `wtypes.evented.spec`

dlink (*this, source, that, target, callable*)

link (*this, source, that, target*)

`wtypes.evented.get_jawn` (*thing, key, object*)

`wtypes.evented.set_jawn` (*thing, key, object*)

class `wtypes.evented.spec_impl`

`__enter__` (*self*)

`__exit__` (*self, *e*)

class `wtypes.evented.wtypes_impl`

Bases: `wtypes.evented.spec_impl`

dlink (*this, source, that, target, callable*)

class `wtypes.evented.Link`

```
_registered_parents
_registered_links
_registered_id
_deferred_changed
_deferred_prior
_depth = 0
_display_id
__enter__(self)
__exit__(self, *e)
link (this, source, that, target='value')
dlink (self, source, that, target, callable=None)
```

Examples

```
>>> class d(Dict): a: int
>>> e, f = d(a=1), d(a=1)
>>> e.dlink('a', f, 'a', lambda x: 2*x)
{'a': 1}
>>> e['a'] = 7
>>> f
{'a': 14}
```

```
observe (self, source="", callable=None)
    The callable has to define a signature.
```

```
_propagate (self, *changed, **prior)
_update_display (self)
_ipython_display_ (self)
_repr_mimebundle_ (self, include=None, exclude=None)
```

```
class wtypes.evented._EventedObject
    Bases: wtypes.evented.Link
```

```
class wtypes.evented._EventedDict (*args, **kwargs)
    Bases: wtypes.evented.\_EventedObject
    _link_parent (self, object)
    __setitem__ (self, key, object)
    update (self, *args, **kwargs)
```

```
class wtypes.evented._EventedDataClass (*args, **kwargs)
    Bases: wtypes.evented.\_EventedObject
    _link_parent (self, object)
    __setattr__ (self, key, object)
```

```
class wtypes.evented._EventedList (*args, **kwargs)
    Bases: wtypes.evented.Link
```

```

    _link_parent (self, object)
    __exit__ (self, *e)
    __setitem__ (self, key, object)
    append (self, object)
    insert (self, index, object)
    extend (self, object)
    pop (self, index=-1)
    observe (self, callable=None)
        The callable has to define a signature.

```

```

class wtypes.evented.List (*args, **kwargs)
    Bases: wtypes.evented._EventedList, wtypes.wtypes.List

```

```

class wtypes.evented.Bunch (*args, **kwargs)
    Bases: wtypes.evented._EventedDict, wtypes.wtypes.Bunch
    An evented dictionary/bunch

```

Examples

```

>>> e, f = Bunch(), Bunch()
>>> e.link('a', f, 'b')
Bunch({})
>>> e['a'] = 1
>>> f.toDict()
{'b': 1}
>>> e.update(a=100)
>>> f.toDict()
{'b': 100}

```

```

>>> f['b'] = 2
>>> assert e['a'] == f['b']
>>> e = Bunch().observe('a', print)
>>> e['a'] = 2
{'new': 2, 'old': None, 'object': Bunch({'a': 2}), 'name': 'a'}

```

```

class wtypes.evented.Dict (*args, **kwargs)
    Bases: wtypes.evented._EventedDict, wtypes.wtypes.Dict

```

An evented dictionary/bunch

Examples

```

>>> e, f = Dict(), Dict()
>>> e.link('a', f, 'b')
{}
>>> e['a'] = 1
>>> f
{'b': 1}
>>> e.update(a=100)
>>> f
{'b': 100}

```

```
>>> f['b'] = 2
>>> assert e['a'] == f['b']
>>> e = Dict().observe('a', print)
>>> e['a'] = 2
{'new': 2, 'old': None, 'object': {'a': 2}, 'name': 'a'}
```

```
class wtypes.evented.DataClass(*args, **kwargs)
    Bases: wtypes.evented._EventedDataClass, wtypes.DataClass
```

```
class wtypes.evented.Namespace(*args, **kwargs)
    Bases: wtypes.evented.Dict
```

An event namespace to visualize track the annotated fields.

Examples

```
>>> # evented.Namespace.register()
```

```
_repr_mimebundle_(self, include=None, exclude=None)
```

```
classmethod register(cls)
```

```
classmethod unregister(cls)
```

4.2.6 wtypes.examples

Module Contents

wtypes.examples.generate_strategy_from_object(*object)

wtypes.examples.generate_strategy_from_type(type)

Generate a hypothesis strategy from a type.

Note: Requires hypothesis_jschema

wtypes.examples.generate_example(type)

wtypes.examples.example

4.2.7 wtypes.python_types

Module Contents

```
class wtypes.python_types._NoType
```

```
class wtypes.python_types._ForwardSchema
    Bases: wtypes.base._ContextMeta
```

A forward reference to an object, the object must exist in sys.modules.

Notes

Python types live on the `__annotations__` attribute.

`_type_args`

`_type_kwargs`

`__getitem__` (*cls*, *object*)

`validate` (*cls*, *object*)

`eval` (*cls*)

`__add__` (*cls*, *object*)

class `wtypes.python_types._ArgumentSchema`

Bases: `wtypes.python_types._ForwardSchema`

`__getitem__` (*cls*, *object*)

class `wtypes.python_types.Args`

Bases: `wtypes.python_types._NoType`, `wtypes.base._NoInit`, `wtypes.base._NoTitle`

class `wtypes.python_types.Kwargs`

Bases: `wtypes.python_types._NoType`, `wtypes.base._NoInit`, `wtypes.base._NoTitle`

class `wtypes.python_types.Forward`

Create type using objects or forward references.

Examples

```
>>> assert Forward['builtins.range']() is range
```

class `wtypes.python_types.Class`

Bases: `wtypes.python_types.Forward`

Create type using objects or forward references.

Examples

```
>>> assert isinstance(range, Class['builtins.range'])
```

classmethod `validate` (*cls*, *object*)

class `wtypes.python_types.Instance`

Bases: `wtypes.python_types.Forward`

Create an instance of a type using objects or forward references.

Examples

```
>>> assert (Instance[range] + Args[10, 20])() == range(10, 20)
>>> assert (Instance['builtins.range'] + Args[10, 20])() == range(10, 20)
>>> assert isinstance(range(10), Instance['builtins.range'])
```

Deferred references.

```
>>> assert not isinstance(1, Instance['pandas.DataFrame'])
>>> assert 'pandas' not in __import__('sys').modules
```

```
classmethod validate(cls, object)
```

4.2.8 wtypes.rdf

RDF properties from wtypes

Note: Requires rdflib

These are actions based on the `_context` of the types.

4.2.9 wtypes.spec

Module Contents

`wtypes.spec.specification`

`wtypes.spec.implementation`

`wtypes.spec.manager`

class `wtypes.spec.spec`

validate_type (*type*)
A hook to validate types.

validate_object (*object, schema*)
A hook to validate types.

4.2.10 wtypes.string_formats

Module Contents

class `wtypes.string_formats.Color`
Bases: `wtypes.String`

class `wtypes.string_formats.Datetime`
Bases: `wtypes.String`

class `wtypes.string_formats.Time`
Bases: `wtypes.String`

class `wtypes.string_formats.Date`
Bases: `wtypes.String`

class `wtypes.string_formats.Email`
Bases: `wtypes.String`

class `wtypes.string_formats.Idnemail`
Bases: `wtypes.String`

```

class wtypes.string_formats.Hostname
    Bases: wtypes.String

class wtypes.string_formats.Idnhostname
    Bases: wtypes.String

class wtypes.string_formats.Ipv4
    Bases: wtypes.String

class wtypes.string_formats.Ipv6
    Bases: wtypes.String

class wtypes.string_formats.Uri
    Bases: wtypes.String

    get
    post
    _httpx_method(self, method, *args, **kwargs)

class wtypes.string_formats.Urireference
    Bases: wtypes.String

class wtypes.string_formats.Iri
    Bases: wtypes.string_formats.Uri

class wtypes.string_formats.Ireference
    Bases: wtypes.String

class wtypes.string_formats.Uritemplate
    Bases: wtypes.String

    expand(self, **kwargs)
    URITemplate(self)

class wtypes.string_formats.Jsonpointer
    Bases: wtypes.String

    resolve_pointer(self, doc, default=None)

class wtypes.string_formats.Relativejsonpointer
    Bases: wtypes.String

class wtypes.string_formats.Regex
    Bases: wtypes.String

```

4.2.11 wtypes.utils

Module Contents

`wtypes.utils.istype(object, cls)`
 instance(*object, type*) and issubclass(*object, cls*)

Examples

```

>>> assert istype(int, int)
>>> assert not istype(10, int)

```

`wtypes.utils.validate_schema(object: object, schema: dict)` → None

```
wtypes.utils.validate_generic(object, cls)
```

4.3 Package Contents

```
wtypes.__version__ = 0.0.2
```

```
wtypes.specification
```

```
wtypes.implementation
```

```
wtypes.manager
```

```
class wtypes.spec
```

```
    validate_type(type)
```

```
        A hook to validate types.
```

```
    validate_object(object, schema)
```

```
        A hook to validate types.
```

```
wtypes.istype(object, cls)
```

```
    instance(object, type) and issubclass(object, cls)
```

Examples

```
>>> assert istype(int, int)
>>> assert not istype(10, int)
```

```
wtypes.validate_schema(object: object, schema: dict) → None
```

```
wtypes.validate_generic(object, cls)
```

```
wtypes.__version__ = 0.0.1
```

```
wtypes.ValidationError
```

```
class wtypes._Implementation
```

```
    An implementation of the pluggy wtypes spec.
```

Notes

The implementation needs to be registered with the plugin manager.

```
validate_type(type)
```

```
validate_object(object, schema)
```

```
wtypes.istype(object, cls)
```

```
instance(object, type) and issubclass(object, cls)
```

Examples

```
>>> assert istype(int, int)
>>> assert not istype(10, int)
```

```
class wtypes._NoTitle
    A subclass suppresses the class name when combining schema

class wtypes._NoInit
    A subclass to restrict initializing an object from the type.

class wtypes._ContextMeta
    Bases: abc.ABCMeta

    Meta operations for the context..

    _context
        The schema the object validates against.

        Type dict

    _type
        The schema the object validates against.

        Type type

    _schema
        Type dict
```

Notes

A context type cannot be verified as it only describes, although some descriptors like SHACL can validate

```
_schema
_context
_type
_merge_args (cls)
_merge_context (cls)
_merge_annotations (cls)
    Merge annotations from the module resolution order.
_merge_schema (cls)
    Merge schema from the module resolution order.
_merge_types (cls)
    Merge schema from the module resolution order.
__matmul__ (cls, object)
validate (cls, object)
    A context type does not validate.
__instancecheck__ (cls, object)
create (cls, name: str, **schema)
    Create a new schema type.

    Parameters
    • name (str) – The title of the new type/schema
    • **schema (dict) – Extra features to add include in the schema.

    Returns
```

Return type type

`__add__ (cls, object)`
`__neg__ (cls)`
The Not version of a type.
`__pos__ (cls)`
The type.
`__and__ (cls, object)`
AllOf the conditions
`__sub__ (cls, object)`
AnyOf the conditions
`__or__ (cls, object)`
OneOf the conditions

class wtypes._SchemaMeta

Bases: `wtypes.base._ContextMeta`

Meta operations for wtypes.

The `_SchemaMeta` ensures that a type's extended schema is validate. Types cannot be generated with invalid schema.

`__schema`
The schema the object validates against.

Type dict

`__neg__ (cls)`
The Not version of a type.
`__pos__ (cls)`
The type.
`__and__ (cls, object)`
AllOf the conditions
`__sub__ (cls, object)`
AnyOf the conditions
`__or__ (cls, object)`
OneOf the conditions

validate (cls, object)
Validate an object against type's schema.

Note: `isinstance` can used for validation, too.

Parameters `object` – An object to validate.

Raises `jsonschema.ValidationError` – The `jsonschema` module validation throws an exception on failure, otherwise the returns a `None` type.

class wtypes._ConstType

Bases: `wtypes.base._SchemaMeta`

`ConstType` permits bracketed syntax for defining complex types.

Note: The bracketed notebook should differentiate actions on types versus those on objects.

```

__getitem__ (cls, object)

class wtypes._ContainerType
    Bases: wtypes.base._ConstType

    ContainerType extras schema from bracketed arguments to define complex types.

    __getitem__ (cls, object)

wtypes._python_to_wtype (object)

wtypes._get_schema_from_typeish (object, key='anyOf')
    infer a schema from an object.

wtypes._lower_key (str)

wtypes._object_to_webtype (object)

wtypes._construct_title (cls)

class wtypes.Trait
    A trait is an object validated by a validate jsonschema.

    _schema

    _context

    classmethod _resolve_defaults (cls, *args, **kwargs)

wtypes.get_jawn (thing, key, object)

class wtypes.Description
    Bases: wtypes.base._NoInit, wtypes.base.Trait, wtypes.base._NoTitle

    An empty type with a description

```

Examples

```

>>> yo = Description['yo']
>>> yo._schema.toDict()
{'description': 'yo'}

```

```

class wtypes.Examples
    Bases: wtypes.base._NoInit, wtypes.base.Trait

class wtypes.Default
    Bases: wtypes.base._NoInit, wtypes.base.Trait

class wtypes.Title
    Bases: wtypes.base._NoInit, wtypes.base.Trait, wtypes.base._NoTitle

    An empty type with a title

```

Examples

```

>>> holla = Title['holla']
>>> holla._schema.toDict()
{'title': 'holla'}

```

```
class wtypes.Const
    Bases: wtypes.base._NoInit, wtypes.base.Trait
    A constant
```

Examples

```
>>> Const[10]._schema.toDict()
{'const': 10}
```

```
>>> assert isinstance('thing', Const['thing'])
>>> assert not isinstance('jawn', Const['thing']), "Because the compiler is from_
↳Philly."
```

```
class wtypes.Bool
    Bases: wtypes.base.Trait
    Boolean type
```

Examples

```
>>> Bool(), Bool(True), Bool(False)
(False, True, False)
>>> assert (Bool + Default[True])()
```

Note: It is not possible to base class `bool` so object creation is customized.

```
class wtypes.Null
    Bases: wtypes.base.Trait
    nil, none, null type
```

Examples

```
>>> Null(None)
>>> assert (Null + Default[None])() is None
```

```
class wtypes._NumericSchema
    Bases: wtypes.base._SchemaMeta
    Meta operations for numerical types
    __rgt__
    __rge__
    __rlt__
    __rle__
    __ge__(cls, object)
        Inclusive minimum
    __gt__(cls, object)
        Exclusive minimum
```

`__le__` (*cls, object*)
Inclusive maximum

`__lt__` (*cls, object*)
Exclusive maximum

`__truediv__` (*cls, object*)
multiple of a number

class `wtypes.Integer`
Bases: `wtypes.base.Trait`, `int`
integer type

Examples

```
>>> assert isinstance(10, Integer)
>>> assert not isinstance(10.1, Integer)
>>> (Integer+Default[9])(9)
9
```

```
>>> bounded = (10< Integer)< 100
>>> bounded._schema.toDict()
{'type': 'integer', 'exclusiveMinimum': 10, 'exclusiveMaximum': 100}
>>> assert isinstance(12, bounded)
>>> assert not isinstance(0, bounded)
>>> assert (Integer/3)(9) == 9
```

class `wtypes.Float`
Bases: `wtypes.base.Trait`, `float`
float type

```
>>> assert isinstance(10, Float)
>>> assert isinstance(10.1, Float)
```

Symbollic conditions.

```
>>> bounded = (10< Float)< 100
>>> bounded._schema.toDict()
{'type': 'number', 'exclusiveMinimum': 10, 'exclusiveMaximum': 100}
```

```
>>> assert isinstance(12.1, bounded)
>>> assert not isinstance(0.1, bounded)
```

Multiples

```
>>> assert (Float+MultipleOf[3])(9) == 9
```

class `wtypes.MultipleOf`
Bases: `wtypes.base.NoInit`, `wtypes.base.Trait`
A multipleof constraint for numeric types.

class `wtypes.Minimum`
Bases: `wtypes.base.NoInit`, `wtypes.base.Trait`
A minimum constraint for numeric types.

```

class wtypes.ExclusiveMinimum
    Bases: wtypes.base._NoInit, wtypes.base.Trait

    A exclusive minimum constraint for numeric types.

class wtypes.Maximum
    Bases: wtypes.base._NoInit, wtypes.base.Trait

    A exclusive maximum constraint for numeric types.

class wtypes.ExclusiveMaximum
    Bases: wtypes.base._NoInit, wtypes.base.Trait

    A exclusive maximum constraint for numeric types.

class wtypes.Properties
    Bases: wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle

    Object properties.

class wtypes._ObjectSchema
    Bases: wtypes.base._SchemaMeta

    Meta operations for the object schema.

    __getitem__ (cls, object)

```

Examples

```

>>> Dict[wtypes.Forward[range], int].__annotations__
{'': typing.Union[abc.Forward, int]}
>>> Dict[wtypes.Forward[range], int]._schema.toDict()
{'type': 'object', 'properties': {}, 'additionalProperties': {'anyOf': [{'type':
↪ ': 'integer'}]}}

```

```

class wtypes._Object
    Base class for validating object types.

    classmethod __init_subclass__ (cls, **schema)

    classmethod from_config_file (cls, *object)

class wtypes.Dict
    Bases: wtypes.base.Trait, dict, wtypes.base._Object

    dict type

```

Examples

```

>>> assert isinstance(Dict, __import__('collections').abc.MutableMapping)
>>> assert (Dict + Default[{'b': 'foo'}])() == {'b': 'foo'}
>>> assert (Dict + Default[{'b': 'foo'}])({'a': 'bar'}) == {'b': 'foo', 'a': 'bar'
↪ '}

```

```

>>> assert isinstance({}, Dict)
>>> assert not isinstance([], Dict)

```

```

>>> assert isinstance({'a': 1}, Dict + Required['a',])
>>> assert not isinstance({}, Dict + Required['a',])

```

```
>>> assert not isinstance({'a': 'b'}, Dict[Integer, Float])
>>> assert Dict[Integer]({'a': 1}) == {'a': 1}
```

```
>>> Dict[{'a': int}]._schema.toDict()
{'type': 'object', 'properties': {'a': {'type': 'integer'}}}
>>> Dict[{'a': int}]({'a': 1})
{'a': 1}
```

__setitem__ (*self*, *key*, *object*)

Only test the key being set to avoid invalid state.

update (*self*, **args*, ***kwargs*)

class wtypes.**Bunch**

Bases: *wtypes.base.Dict*, *munch.Munch*

Bunch type

Examples

```
>>> Bunch[{'a': int}]._schema.toDict()
{'type': 'object', 'properties': {'a': {'type': 'integer'}}}
>>> Bunch[{'a': int}]({'a': 1}).toDict()
{'a': 1}
```

class wtypes.**AdditionalProperties**

Bases: *wtypes.base.Trait*, *wtypes.base._NoInit*, *wtypes.base._NoTitle*

Additional object properties.

class wtypes.**Required**

Bases: *wtypes.base.Trait*, *wtypes.base._NoInit*, *wtypes.base._NoTitle*

Required properties.

class wtypes.**minProperties**

Bases: *wtypes.base.Trait*, *wtypes.base._NoInit*, *wtypes.base._NoTitle*

Minimum number of properties.

class wtypes.**maxProperties**

Bases: *wtypes.base.Trait*, *wtypes.base._NoInit*, *wtypes.base._NoTitle*

Maximum number of properties.

class wtypes.**PropertyNames**

Bases: *wtypes.base.Trait*, *wtypes.base._NoInit*, *wtypes.base._NoTitle*

Property name constraints.

class wtypes.**Dependencies**

Bases: *wtypes.base.Trait*, *wtypes.base._NoInit*, *wtypes.base._NoTitle*

Properties dependencies.

class wtypes.**PatternProperties**

Bases: *wtypes.base.Trait*, *wtypes.base._NoInit*, *wtypes.base._NoTitle*

Pattern properties names.

```
class wtypes._StringSchema
    Bases: wtypes.base._SchemaMeta

    Meta operations for strings types.

    __mod__ (cls, object)
        A pattern string type.

    __gt__ (cls, object)
        Minumum string length

    __lt__ (cls, object)
        Maximum string length
```

```
class wtypes.String
    Bases: wtypes.base.Trait, str

    string type.
```

Examples

```
>>> assert isinstance('abc', String)
>>> assert (String+Default['abc'])() == 'abc'
```

String patterns

```
>>> assert isinstance('abc', String%"^a")
>>> assert not isinstance('abc', String%"^b")
```

String constraints

```
>>> assert isinstance('abc', (2<String)<10)
>>> assert not isinstance('a', (2<String)<10)
>>> assert not isinstance('a'*100, (2<String)<10)
```

```
class wtypes.MinLength
    Bases: wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle

    Minimum length of a string type.
```

```
class wtypes.MaxLength
    Bases: wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle

    Maximum length of a string type.
```

```
class wtypes.ContentMediaType
    Bases: wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle

    Content type of a string.
```

```
class wtypes.Pattern
    Bases: wtypes.base.Trait, wtypes.base._NoInit

    A regular expression pattern.
```

```
class wtypes.Format
    Bases: wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle
```

```
class wtypes._ListSchema
    Bases: wtypes.base._SchemaMeta

    Meta operations for list types.
```

`__getitem__` (*cls, object*)
List meta operations for bracketed type notation.

Examples

```
>>> List[wtypes.Forward[range], int]._type
typing.List[typing.Union[abc.Forward, int]]
```

```
>>> Tuple[wtypes.Forward[range], int]._type
typing.Tuple[abc.Forward, int]
```

`__gt__` (*cls, object*)
Minimum array length

`__lt__` (*cls, object*)
Maximum array length

class wtypes.**List**
Bases: *wtypes.base.Trait*, list
List type

Examples

List

```
>>> assert isinstance([], List)
>>> assert not isinstance({}, List)
```

Typed list

```
>>> assert List[Integer]([1, 2, 3])
>>> assert isinstance([1], List[Integer])
>>> assert not isinstance([1.1], List[Integer])
```

```
>>> List[Integer, String]._schema.toDict()
{'type': 'array', 'items': {'anyOf': [{'type': 'integer'}, {'type': 'string'}]}}
```

Tuple

```
>>> assert List[Integer, String]([1, 'abc', 2])
>>> assert isinstance([1, '1'], List[Integer, String])
>>> assert not isinstance([1, {}], List[Integer, String])
```

`__verify_item` (*self, object, id=None*)
Elemental verification for interactive type checking.

`__setitem__` (*self, id, object*)

`append` (*self, object*)

`insert` (*self, id, object*)

`extend` (*self, object*)

`pop` (*self, index=-1*)

```
class wtypes.Unique
    Bases: wtypes.base.List
    Unique list type
```

Examples

```
>>> assert Unique(list('abc'))
>>> assert isinstance([1,2], Unique)
>>> assert not isinstance([1,1], Unique)
```

```
class wtypes.Tuple
    Bases: wtypes.base.List
    tuple type
```

Note: There are no tuples in json, they are typed lists.

```
>>> assert Tuple._schema == List._schema
```

Examples

```
>>> assert isinstance([1,2], Tuple)
>>> assert Tuple[Integer, String]([1, 'abc'])
>>> Tuple[Integer, String]._schema.toDict()
{'type': 'array', 'items': [{'type': 'integer'}, {'type': 'string'}]}
```

```
>>> assert isinstance([1,'1'], Tuple[Integer, String])
>>> assert not isinstance([1,1], Tuple[Integer, String])
```

```
class wtypes.UniqueItems
    Bases: wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle
    Schema for unique items in a list.
```

```
class wtypes.Contains
    Bases: wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle
```

```
class wtypes.Items
    Bases: wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle
```

```
class wtypes.AdditionalItems
    Bases: wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle
```

```
class wtypes.MinItems
    Bases: wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle
    Minimum length of an array.
```

```
class wtypes.MaxItems
    Bases: wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle
    Maximum length of an array.
```

```
class wtypes.Enum
    Bases: wtypes.base.Trait

    An enumerate type that is restricted to its inputs.
```

Examples

```
>>> assert Enum['cat', 'dog']('cat')
>>> assert isinstance('cat', Enum['cat', 'dog'])
>>> assert not isinstance('', Enum['cat', 'dog'])
```

```
class wtypes.ContentEncoding
    Bases: Enum['7bit 8bit binary quoted-printable base64'.split()], wtypes.base._NoInit, wtypes.base._NoTitle
```

Content encodings for a string.

```
class wtypes.If
    Bases: wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle

    if condition type
```

```
class wtypes.Then
    Bases: wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle

    then condition type
```

```
class wtypes.Else
    Bases: wtypes.base.Trait, wtypes.base._NoInit, wtypes.base._NoTitle

    else condition type
```

```
class wtypes._NotType
    Bases: wtypes.base._ConstType

    validate (cls, object)

    __getitem__ (cls, object)
```

```
class wtypes.Not
    Bases: wtypes.base.Trait

    not schema.
```

Examples

```
>>> assert Not[wtypes.String](100) == 100
>>> assert not isinstance('abc', Not[wtypes.String])
```

Note: See the `__neg__` method for symbolic not composition.

```
class wtypes._AnyOfType
    Bases: wtypes.base._ConstType

    validate (cls, object)

    __getitem__ (cls, object)
```

```
class wtypes.AnyOf
    Bases: wtypes.base.Trait

    anyOf combined schema.
```

Examples

```
>>> assert AnyOf[wtypes.Integer, wtypes.String]('abc')
>>> assert isinstance(10, AnyOf[wtypes.Integer, wtypes.String])
>>> assert not isinstance([], AnyOf[wtypes.Integer, wtypes.String])
```

```
class wtypes._AllOfType
    Bases: wtypes.base._ConstType

    validate (cls, object)

    __getitem__ (cls, object)
```

```
class wtypes.AllOf
    Bases: wtypes.base.Trait

    allOf combined schema.
```

Examples

```
>>> assert AllOf[wtypes.Float>0, wtypes.Integer/3](9)
>>> assert isinstance(9, AllOf[wtypes.Float>0, wtypes.Integer/3])
>>> assert not isinstance(-9, AllOf[wtypes.Float>0, wtypes.Integer/3])
```

```
class wtypes._OneOfType
    Bases: wtypes.base._ConstType

    validate (cls, object)

    __getitem__ (cls, object)
```

```
class wtypes.OneOf
    Bases: wtypes.base.Trait

    oneOf combined schema.
```

Examples

```
>>> assert OneOf[wtypes.Float>0, wtypes.Integer/3](-9)
>>> assert isinstance(-9, OneOf[wtypes.Float>0, wtypes.Integer/3])
>>> assert not isinstance(9, OneOf[wtypes.Float>0, wtypes.Integer/3])
```

```
class wtypes.Setter

    __setattr__ (self, key, object)
        Only test the attribute being set to avoid invalid state.
```

```
class wtypes.DataClass
    Bases: wtypes.dataclass.Setter, wtypes.Trait, wtypes.base._Object

    Validating dataclass type
```

Examples

```
>>> class q(DataClass): a: int
>>> q._schema.toDict()
{'type': 'object', 'properties': {'a': {'type': 'integer'}}, 'required': ['a']}
```

```
>>> q(a=10)
q(a=10)
```

```
>>> assert not isinstance({}, q)
```

```
classmethod __init_subclass__(cls, **kwargs)
```

```
class wtypes._NoType
```

```
class wtypes._ForwardSchema
```

Bases: *wtypes.base._ContextMeta*

A forward reference to an object, the object must exist in sys.modules.

Notes

Python types live on the `__annotations__` attribute.

`_type_args`

`_type_kwargs`

`__getitem__` (*cls, object*)

`validate` (*cls, object*)

`eval` (*cls*)

`__add__` (*cls, object*)

```
class wtypes._ArgumentSchema
```

Bases: *wtypes.python_types._ForwardSchema*

`__getitem__` (*cls, object*)

```
class wtypes.Args
```

Bases: *wtypes.python_types._NoType, wtypes.base._NoInit, wtypes.base._NoTitle*

```
class wtypes.Kwargs
```

Bases: *wtypes.python_types._NoType, wtypes.base._NoInit, wtypes.base._NoTitle*

```
class wtypes.Forward
```

Create type using objects or forward references.

Examples

```
>>> assert Forward['builtins.range']() is range
```

```
class wtypes.Class
```

Bases: *wtypes.python_types.Forward*

Create type using objects or forward references.

Examples

```
>>> assert isinstance(range, Class['builtins.range'])
```

classmethod validate (*cls, object*)

class wtypes.Instance

Bases: *wtypes.python_types.Forward*

Create an instance of a type using objects or forward references.

Examples

```
>>> assert (Instance[range] + Args[10, 20])() == range(10, 20)
>>> assert (Instance['builtins.range'] + Args[10, 20])() == range(10, 20)
>>> assert isinstance(range(10), Instance['builtins.range'])
```

Deferred references.

```
>>> assert not isinstance(1, Instance['pandas.DataFrame'])
>>> assert 'pandas' not in __import__('sys').modules
```

classmethod validate (*cls, object*)

class wtypes.Color

Bases: *wtypes.String*

class wtypes.Datetime

Bases: *wtypes.String*

class wtypes.Time

Bases: *wtypes.String*

class wtypes.Date

Bases: *wtypes.String*

class wtypes.Email

Bases: *wtypes.String*

class wtypes.Idnemail

Bases: *wtypes.String*

class wtypes.Hostname

Bases: *wtypes.String*

class wtypes.Idnhostname

Bases: *wtypes.String*

class wtypes.Ipv4

Bases: *wtypes.String*

class wtypes.Ipv6

Bases: *wtypes.String*

class wtypes.Uri

Bases: *wtypes.String*

get

post

```
    _httpx_method (self, method, *args, **kwargs)

class wtypes.Urireference
    Bases: wtypes.String

class wtypes.Iri
    Bases: wtypes.string_formats.Uri

class wtypes.Irireference
    Bases: wtypes.String

class wtypes.Uritemplate
    Bases: wtypes.String

    expand (self, **kwargs)

    URITemplate (self)

class wtypes.Jsonpointer
    Bases: wtypes.String

    resolve_pointer (self, doc, default=None)

class wtypes.Relativejsonpointer
    Bases: wtypes.String

class wtypes.Regex
    Bases: wtypes.String
```


C

`conf`, 3
`conftest`, 7

S

`setup`, 5

W

`wtypes`, 9
`wtypes.base`, 9
`wtypes.combining_types`, 21
`wtypes.content_types`, 22
`wtypes.dataclass`, 22
`wtypes.evented`, 23
`wtypes.examples`, 26
`wtypes.python_types`, 26
`wtypes.rdf`, 28
`wtypes.spec`, 28
`wtypes.string_formats`, 28
`wtypes.utils`, 29
`wtypes.widgets`, 9
`wtypes.widgets.ipynbwidgets`, 9

Symbols

- `_AllOfType` (class in `wtypes`), 42
- `_AllOfType` (class in `wtypes.combining_types`), 21
- `_AnyOfType` (class in `wtypes`), 41
- `_AnyOfType` (class in `wtypes.combining_types`), 21
- `_ArgumentSchema` (class in `wtypes`), 43
- `_ArgumentSchema` (class in `wtypes.python_types`), 27
- `_ConstType` (class in `wtypes`), 32
- `_ConstType` (class in `wtypes.base`), 12
- `_ContainerType` (class in `wtypes`), 33
- `_ContainerType` (class in `wtypes.base`), 12
- `_Content` (class in `wtypes.content_types`), 22
- `_ContextMeta` (class in `wtypes`), 31
- `_ContextMeta` (class in `wtypes.base`), 10
- `_EventedDataClass` (class in `wtypes.evented`), 24
- `_EventedDict` (class in `wtypes.evented`), 24
- `_EventedList` (class in `wtypes.evented`), 24
- `_EventedObject` (class in `wtypes.evented`), 24
- `_ForwardSchema` (class in `wtypes`), 43
- `_ForwardSchema` (class in `wtypes.python_types`), 26
- `_Implementation` (class in `wtypes`), 30
- `_Implementation` (class in `wtypes.base`), 10
- `_ListSchema` (class in `wtypes`), 38
- `_ListSchema` (class in `wtypes.base`), 18
- `_NoInit` (class in `wtypes`), 31
- `_NoInit` (class in `wtypes.base`), 10
- `_NoTitle` (class in `wtypes`), 30
- `_NoTitle` (class in `wtypes.base`), 10
- `_NoType` (class in `wtypes`), 43
- `_NoType` (class in `wtypes.python_types`), 26
- `_NotType` (class in `wtypes`), 41
- `_NotType` (class in `wtypes.combining_types`), 21
- `_NumericSchema` (class in `wtypes`), 34
- `_NumericSchema` (class in `wtypes.base`), 14
- `_Object` (class in `wtypes`), 36
- `_Object` (class in `wtypes.base`), 16
- `_ObjectSchema` (class in `wtypes`), 36
- `_ObjectSchema` (class in `wtypes.base`), 15
- `_OneOfType` (class in `wtypes`), 42
- `_OneOfType` (class in `wtypes.combining_types`), 22
- `_SchemaMeta` (class in `wtypes`), 32
- `_SchemaMeta` (class in `wtypes.base`), 11
- `_StringSchema` (class in `wtypes`), 37
- `_StringSchema` (class in `wtypes.base`), 17
- `__add__` () (`wtypes._ContextMeta` method), 32
- `__add__` () (`wtypes._ForwardSchema` method), 43
- `__add__` () (`wtypes.base._ContextMeta` method), 11
- `__add__` () (`wtypes.python_types._ForwardSchema` method), 27
- `__and__` () (`wtypes._ContextMeta` method), 32
- `__and__` () (`wtypes._SchemaMeta` method), 32
- `__and__` () (`wtypes.base._ContextMeta` method), 11
- `__and__` () (`wtypes.base._SchemaMeta` method), 12
- `__enter__` () (`wtypes.evented.Link` method), 24
- `__enter__` () (`wtypes.evented.spec_impl` method), 23
- `__exit__` () (`wtypes.evented.Link` method), 24
- `__exit__` () (`wtypes.evented._EventedList` method), 25
- `__exit__` () (`wtypes.evented.spec_impl` method), 23
- `__ge__` () (`wtypes._NumericSchema` method), 34
- `__ge__` () (`wtypes.base._NumericSchema` method), 14
- `__getitem__` () (`wtypes._AllOfType` method), 42
- `__getitem__` () (`wtypes._AnyOfType` method), 41
- `__getitem__` () (`wtypes._ArgumentSchema` method), 43
- `__getitem__` () (`wtypes._ConstType` method), 33
- `__getitem__` () (`wtypes._ContainerType` method), 33
- `__getitem__` () (`wtypes._ForwardSchema` method), 43
- `__getitem__` () (`wtypes._ListSchema` method), 38
- `__getitem__` () (`wtypes._NotType` method), 41
- `__getitem__` () (`wtypes._ObjectSchema` method), 36
- `__getitem__` () (`wtypes._OneOfType` method), 42
- `__getitem__` () (`wtypes.base._ConstType` method), 12
- `__getitem__` () (`wtypes.base._ContainerType` method), 12
- `__getitem__` () (`wtypes.base._ListSchema` method), 18

```

__getitem__() (wtypes.base._ObjectSchema method), 15
__getitem__() (wtypes.combining_types._AllOfType method), 21
__getitem__() (wtypes.combining_types._AnyOfType method), 21
__getitem__() (wtypes.combining_types._NotType method), 21
__getitem__() (wtypes.combining_types._OneOfType method), 22
__getitem__() (wtypes.python_types._ArgumentSchema method), 27
__getitem__() (wtypes.python_types._ForwardSchema method), 27
__gt__() (wtypes._ListSchema method), 39
__gt__() (wtypes._NumericSchema method), 34
__gt__() (wtypes._StringSchema method), 38
__gt__() (wtypes.base._ListSchema method), 18
__gt__() (wtypes.base._NumericSchema method), 14
__gt__() (wtypes.base._StringSchema method), 17
__init_subclass__() (wtypes.DataClass class method), 43
__init_subclass__() (wtypes._Object class method), 36
__init_subclass__() (wtypes.base._Object class method), 16
__init_subclass__() (wtypes.dataclass.DataClass class method), 23
__instancecheck__() (wtypes._ContextMeta method), 31
__instancecheck__() (wtypes.base._ContextMeta method), 11
__le__() (wtypes._NumericSchema method), 34
__le__() (wtypes.base._NumericSchema method), 14
__lt__() (wtypes._ListSchema method), 39
__lt__() (wtypes._NumericSchema method), 35
__lt__() (wtypes._StringSchema method), 38
__lt__() (wtypes.base._ListSchema method), 18
__lt__() (wtypes.base._NumericSchema method), 14
__lt__() (wtypes.base._StringSchema method), 17
__matmul__() (wtypes._ContextMeta method), 31
__matmul__() (wtypes.base._ContextMeta method), 11
__mod__() (wtypes._StringSchema method), 38
__mod__() (wtypes.base._StringSchema method), 17
__neg__() (wtypes._ContextMeta method), 32
__neg__() (wtypes._SchemaMeta method), 32
__neg__() (wtypes.base._ContextMeta method), 11
__neg__() (wtypes.base._SchemaMeta method), 11
__or__() (wtypes._ContextMeta method), 32
__or__() (wtypes._SchemaMeta method), 32
__or__() (wtypes.base._ContextMeta method), 11
__or__() (wtypes.base._SchemaMeta method), 12
__pos__() (wtypes._ContextMeta method), 32
__pos__() (wtypes._SchemaMeta method), 32
__pos__() (wtypes.base._ContextMeta method), 11
__pos__() (wtypes.base._SchemaMeta method), 11
__rge__() (wtypes._NumericSchema attribute), 34
__rge__() (wtypes.base._NumericSchema attribute), 14
__rgt__() (wtypes._NumericSchema attribute), 34
__rgt__() (wtypes.base._NumericSchema attribute), 14
__rle__() (wtypes._NumericSchema attribute), 34
__rle__() (wtypes.base._NumericSchema attribute), 14
__rlt__() (wtypes._NumericSchema attribute), 34
__rlt__() (wtypes.base._NumericSchema attribute), 14
__setattr__() (wtypes.Setter method), 42
__setattr__() (wtypes.dataclass.Setter method), 23
__setattr__() (wtypes.evented._EventedDataClass method), 24
__setitem__() (wtypes.Dict method), 37
__setitem__() (wtypes.List method), 39
__setitem__() (wtypes.base.Dict method), 16
__setitem__() (wtypes.base.List method), 19
__setitem__() (wtypes.evented._EventedDict method), 24
__setitem__() (wtypes.evented._EventedList method), 25
__sub__() (wtypes._ContextMeta method), 32
__sub__() (wtypes._SchemaMeta method), 32
__sub__() (wtypes.base._ContextMeta method), 11
__sub__() (wtypes.base._SchemaMeta method), 12
__truediv__() (wtypes._NumericSchema method), 35
__truediv__() (wtypes.base._NumericSchema method), 14
__version__() (in module wtypes), 30
__version__() (in module wtypes.base), 10
_construct_title() (in module wtypes), 33
_construct_title() (in module wtypes.base), 12
_context (wtypes.Trait attribute), 33
_context (wtypes._ContextMeta attribute), 31
_context (wtypes.base.Trait attribute), 12
_context (wtypes.base._ContextMeta attribute), 10
_deferred_changed (wtypes.evented.Link attribute), 24
_deferred_prior (wtypes.evented.Link attribute), 24
_depth (wtypes.evented.Link attribute), 24
_display_id (wtypes.evented.Link attribute), 24
_get_schema_from_typeish() (in module wtypes), 33
_get_schema_from_typeish() (in module wtypes.base), 12
_httpx_method() (wtypes.Uri method), 44
_httpx_method() (wtypes.string_formats.Uri method), 29
_ipython_display() (wtypes.evented.Link method), 24

```

`_link_parent()` (*wtypes.evented._EventedDataClass method*), 24
`_link_parent()` (*wtypes.evented._EventedDict method*), 24
`_link_parent()` (*wtypes.evented._EventedList method*), 24
`_lower_key()` (*in module wtypes*), 33
`_lower_key()` (*in module wtypes.base*), 12
`_merge_annotations()` (*wtypes._ContextMeta method*), 31
`_merge_annotations()` (*wtypes.base._ContextMeta method*), 11
`_merge_args()` (*wtypes._ContextMeta method*), 31
`_merge_args()` (*wtypes.base._ContextMeta method*), 11
`_merge_context()` (*wtypes._ContextMeta method*), 31
`_merge_context()` (*wtypes.base._ContextMeta method*), 11
`_merge_schema()` (*wtypes._ContextMeta method*), 31
`_merge_schema()` (*wtypes.base._ContextMeta method*), 11
`_merge_types()` (*wtypes._ContextMeta method*), 31
`_merge_types()` (*wtypes.base._ContextMeta method*), 11
`_object_to_webtype()` (*in module wtypes*), 33
`_object_to_webtype()` (*in module wtypes.base*), 12
`_propagate()` (*wtypes.evented.Link method*), 24
`_python_to_wtype()` (*in module wtypes*), 33
`_python_to_wtype()` (*in module wtypes.base*), 12
`_registered_id` (*wtypes.evented.Link attribute*), 24
`_registered_links` (*wtypes.evented.Link attribute*), 24
`_registered_parents` (*wtypes.evented.Link attribute*), 23
`_repr_data_()` (*wtypes.content_types._Content method*), 22
`_repr_metadata_()` (*wtypes.content_types._Content method*), 22
`_repr_mimebundle_()` (*wtypes.content_types._Content method*), 22
`_repr_mimebundle_()` (*wtypes.evented.Link method*), 24
`_repr_mimebundle_()` (*wtypes.evented.Namespace method*), 26
`_resolve_defaults()` (*wtypes.Trait class method*), 33
`_resolve_defaults()` (*wtypes.base.Trait class method*), 12
`_schema` (*wtypes.Trait attribute*), 33
`_schema` (*wtypes._ContextMeta attribute*), 31
`_schema` (*wtypes._SchemaMeta attribute*), 32
`_schema` (*wtypes.base.Trait attribute*), 12
`_schema` (*wtypes.base._ContextMeta attribute*), 10
`_schema` (*wtypes.base._SchemaMeta attribute*), 11
`_type` (*wtypes._ContextMeta attribute*), 31
`_type` (*wtypes.base._ContextMeta attribute*), 10
`_type_args` (*wtypes._ForwardSchema attribute*), 43
`_type_args` (*wtypes.python_types._ForwardSchema attribute*), 27
`_type_kwargs` (*wtypes._ForwardSchema attribute*), 43
`_type_kwargs` (*wtypes.python_types._ForwardSchema attribute*), 27
`_update_display()` (*wtypes.evented.Link method*), 24
`_verify_item()` (*wtypes.List method*), 39
`_verify_item()` (*wtypes.base.List method*), 19

A

AdditionalItems (*class in wtypes*), 40
AdditionalItems (*class in wtypes.base*), 20
AdditionalProperties (*class in wtypes*), 37
AdditionalProperties (*class in wtypes.base*), 17
AllOf (*class in wtypes*), 42
AllOf (*class in wtypes.combining_types*), 22
AnyOf (*class in wtypes*), 41
AnyOf (*class in wtypes.combining_types*), 21
append() (*wtypes.base.List method*), 19
append() (*wtypes.evented._EventedList method*), 25
append() (*wtypes.List method*), 39
Args (*class in wtypes*), 43
Args (*class in wtypes.python_types*), 27
autoapi_dirs (*in module conf*), 3
autoapi_type (*in module conf*), 3

B

Bool (*class in wtypes*), 34
Bool (*class in wtypes.base*), 13
Bunch (*class in wtypes*), 37
Bunch (*class in wtypes.base*), 16
Bunch (*class in wtypes.evented*), 25

C

chinese (*in module conf*), 3
Class (*class in wtypes*), 43
Class (*class in wtypes.python_types*), 27
Color (*class in wtypes*), 44
Color (*class in wtypes.string_formats*), 28
conf (*module*), 3
conftest (*module*), 7
Const (*class in wtypes*), 33
Const (*class in wtypes.base*), 13
Contains (*class in wtypes*), 40

Contains (class in *wtypes.base*), 20
 ContentEncoding (class in *wtypes*), 41
 ContentEncoding (class in *wtypes.base*), 20
 ContentMediaType (class in *wtypes*), 38
 ContentMediaType (class in *wtypes.base*), 18
 context (in module *conf*), 3
 create() (*wtypes._ContextMeta* method), 31
 create() (*wtypes.base._ContextMeta* method), 11

D

DataClass (class in *wtypes*), 42
 DataClass (class in *wtypes.dataclass*), 23
 DataClass (class in *wtypes.evented*), 26
 Date (class in *wtypes*), 44
 Date (class in *wtypes.string_formats*), 28
 Datetime (class in *wtypes*), 44
 Datetime (class in *wtypes.string_formats*), 28
 Default (class in *wtypes*), 33
 Default (class in *wtypes.base*), 13
 Dependencies (class in *wtypes*), 37
 Dependencies (class in *wtypes.base*), 17
 Description (class in *wtypes*), 33
 Description (class in *wtypes.base*), 12
 Dict (class in *wtypes*), 36
 Dict (class in *wtypes.base*), 16
 Dict (class in *wtypes.evented*), 25
 dlink() (*wtypes.evented.Link* method), 24
 dlink() (*wtypes.evented.spec* method), 23
 dlink() (*wtypes.evented.wtypes_impl* method), 23

E

Else (class in *wtypes*), 41
 Else (class in *wtypes.base*), 21
 Email (class in *wtypes*), 44
 Email (class in *wtypes.string_formats*), 28
 Enum (class in *wtypes*), 40
 Enum (class in *wtypes.base*), 20
 eval() (*wtypes._ForwardSchema* method), 43
 eval() (*wtypes.python_types._ForwardSchema* method), 27
 example (in module *wtypes.examples*), 26
 Examples (class in *wtypes*), 33
 Examples (class in *wtypes.base*), 13
 ExclusiveMaximum (class in *wtypes*), 36
 ExclusiveMaximum (class in *wtypes.base*), 15
 ExclusiveMinimum (class in *wtypes*), 35
 ExclusiveMinimum (class in *wtypes.base*), 15
 expand() (*wtypes.string_formats.Uritemplate* method), 29
 expand() (*wtypes.Uritemplate* method), 45
 extend() (*wtypes.base.List* method), 19
 extend() (*wtypes.evented._EventedList* method), 25
 extend() (*wtypes.List* method), 39
 extensions (in module *conf*), 3

F

flit_data (in module *setup*), 5
 Float (class in *wtypes*), 35
 Float (class in *wtypes.base*), 15
 Format (class in *wtypes*), 38
 Format (class in *wtypes.base*), 18
 Forward (class in *wtypes*), 43
 Forward (class in *wtypes.python_types*), 27
 from_config_file() (*wtypes._Object* class method), 36
 from_config_file() (*wtypes.base._Object* class method), 16

G

generate_example() (in module *wtypes.examples*), 26
 generate_strategy_from_object() (in module *wtypes.examples*), 26
 generate_strategy_from_type() (in module *wtypes.examples*), 26
 get (*wtypes.string_formats.Uri* attribute), 29
 get (*wtypes.Uri* attribute), 44
 get_jawn() (in module *wtypes*), 33
 get_jawn() (in module *wtypes.base*), 12
 get_jawn() (in module *wtypes.evented*), 23

H

Hostname (class in *wtypes*), 44
 Hostname (class in *wtypes.string_formats*), 28
 html_context (in module *conf*), 3
 html_static_path (in module *conf*), 3

I

Idnemail (class in *wtypes*), 44
 Idnemail (class in *wtypes.string_formats*), 28
 Idnhostname (class in *wtypes*), 44
 Idnhostname (class in *wtypes.string_formats*), 29
 If (class in *wtypes*), 41
 If (class in *wtypes.base*), 20
 implementation (in module *wtypes*), 30
 implementation (in module *wtypes.spec*), 28
 insert() (*wtypes.base.List* method), 19
 insert() (*wtypes.evented._EventedList* method), 25
 insert() (*wtypes.List* method), 39
 Instance (class in *wtypes*), 44
 Instance (class in *wtypes.python_types*), 27
 Integer (class in *wtypes*), 35
 Integer (class in *wtypes.base*), 14
 Ipv4 (class in *wtypes*), 44
 Ipv4 (class in *wtypes.string_formats*), 29
 Ipv6 (class in *wtypes*), 44
 Ipv6 (class in *wtypes.string_formats*), 29
 Iri (class in *wtypes*), 45

Iri (class in *wtypes.string_formats*), 29
 Irireference (class in *wtypes*), 45
 Irireference (class in *wtypes.string_formats*), 29
 istype() (in module *wtypes*), 30
 istype() (in module *wtypes.base*), 10
 istype() (in module *wtypes.utils*), 29
 Items (class in *wtypes*), 40
 Items (class in *wtypes.base*), 20

J

japanese (in module *conf*), 3
 Jsonpointer (class in *wtypes*), 45
 Jsonpointer (class in *wtypes.string_formats*), 29

K

Kwargs (class in *wtypes*), 43
 Kwargs (class in *wtypes.python_types*), 27

L

language_user (in module *conf*), 3
 latex_elements_user (in module *conf*), 3
 latex_engine (in module *conf*), 3
 latex_engine_user (in module *conf*), 3
 latex_use_xindy (in module *conf*), 3
 Link (class in *wtypes.evented*), 23
 link() (*wtypes.evented.Link* method), 24
 link() (*wtypes.evented.spec* method), 23
 List (class in *wtypes*), 39
 List (class in *wtypes.base*), 19
 List (class in *wtypes.evented*), 25

M

manager (in module *wtypes*), 30
 manager (in module *wtypes.spec*), 28
 master_doc (in module *conf*), 3
 Maximum (class in *wtypes*), 36
 Maximum (class in *wtypes.base*), 15
 MaxItems (class in *wtypes*), 40
 MaxItems (class in *wtypes.base*), 20
 MaxLength (class in *wtypes*), 38
 MaxLength (class in *wtypes.base*), 18
 maxProperties (class in *wtypes*), 37
 maxProperties (class in *wtypes.base*), 17
 metadata (in module *setup*), 5
 Minimum (class in *wtypes*), 35
 Minimum (class in *wtypes.base*), 15
 MinItems (class in *wtypes*), 40
 MinItems (class in *wtypes.base*), 20
 MinLength (class in *wtypes*), 38
 MinLength (class in *wtypes.base*), 18
 minProperties (class in *wtypes*), 37
 minProperties (class in *wtypes.base*), 17
 MultipleOf (class in *wtypes*), 35

MultipleOf (class in *wtypes.base*), 15

N

Namespace (class in *wtypes.evented*), 26
 Not (class in *wtypes*), 41
 Not (class in *wtypes.combining_types*), 21
 Null (class in *wtypes*), 34
 Null (class in *wtypes.base*), 14

O

observe() (*wtypes.evented._EventedList* method), 25
 observe() (*wtypes.evented.Link* method), 24
 OneOf (class in *wtypes*), 42
 OneOf (class in *wtypes.combining_types*), 22

P

Pattern (class in *wtypes*), 38
 Pattern (class in *wtypes.base*), 18
 PatternProperties (class in *wtypes*), 37
 PatternProperties (class in *wtypes.base*), 17
 pop() (*wtypes.base.List* method), 19
 pop() (*wtypes.evented._EventedList* method), 25
 pop() (*wtypes.List* method), 39
 post (*wtypes.string_formats.Uri* attribute), 29
 post (*wtypes.Uri* attribute), 44
 project_language (in module *conf*), 3
 Properties (class in *wtypes*), 36
 Properties (class in *wtypes.base*), 15
 PropertyNames (class in *wtypes*), 37
 PropertyNames (class in *wtypes.base*), 17

R

Regex (class in *wtypes*), 45
 Regex (class in *wtypes.string_formats*), 29
 register() (*wtypes.evented.Namespace* class method), 26
 Relativejsonpointer (class in *wtypes*), 45
 Relativejsonpointer (class in *wtypes.string_formats*), 29
 Required (class in *wtypes*), 37
 Required (class in *wtypes.base*), 17
 resolve_pointer() (*wtypes.Jsonpointer* method), 45
 resolve_pointer() (*wtypes.string_formats.Jsonpointer* method), 29

S

set_jawn() (in module *wtypes.evented*), 23
 Setter (class in *wtypes*), 42
 Setter (class in *wtypes.dataclass*), 23
 setup (module), 5
 simpleTypes (in module *wtypes*), 9

spec (class in wtypes), 30
 spec (class in wtypes.evented), 23
 spec (class in wtypes.spec), 28
 spec_impl (class in wtypes.evented), 23
 specification (in module wtypes), 30
 specification (in module wtypes.spec), 28
 String (class in wtypes), 38
 String (class in wtypes.base), 17
 SUFFIX (in module conf), 3

T

TextHtml (class in wtypes.content_types), 22
 TextMarkdown (class in wtypes.content_types), 22
 TextPlain (class in wtypes.content_types), 22
 theme (in module conf), 3
 Then (class in wtypes), 41
 Then (class in wtypes.base), 20
 Time (class in wtypes), 44
 Time (class in wtypes.string_formats), 28
 Title (class in wtypes), 33
 Title (class in wtypes.base), 13
 Trait (class in wtypes), 33
 Trait (class in wtypes.base), 12
 Tuple (class in wtypes), 40
 Tuple (class in wtypes.base), 19

U

Unique (class in wtypes), 39
 Unique (class in wtypes.base), 19
 UniqueItems (class in wtypes), 40
 UniqueItems (class in wtypes.base), 20
 unregister() (wtypes.evented.Namespace class method), 26
 update() (wtypes.base.Dict method), 16
 update() (wtypes.Dict method), 37
 update() (wtypes.evented._EventDict method), 24
 Uri (class in wtypes), 44
 Uri (class in wtypes.string_formats), 29
 Urireference (class in wtypes), 45
 Urireference (class in wtypes.string_formats), 29
 Uritemplate (class in wtypes), 45
 Uritemplate (class in wtypes.string_formats), 29
 URITemplate() (wtypes.string_formats.Uritemplate method), 29
 URITemplate() (wtypes.Uritemplate method), 45
 using_rtd_theme (in module conf), 3

V

validate() (wtypes._AllOfType method), 42
 validate() (wtypes._AnyOfType method), 41
 validate() (wtypes._ContextMeta method), 31
 validate() (wtypes._ForwardSchema method), 43
 validate() (wtypes._NotType method), 41
 validate() (wtypes._OneOfType method), 42

validate() (wtypes._SchemaMeta method), 32
 validate() (wtypes.base._ContextMeta method), 11
 validate() (wtypes.base._SchemaMeta method), 12
 validate() (wtypes.Class class method), 44
 validate() (wtypes.combining_types._AllOfType method), 21
 validate() (wtypes.combining_types._AnyOfType method), 21
 validate() (wtypes.combining_types._NotType method), 21
 validate() (wtypes.combining_types._OneOfType method), 22
 validate() (wtypes.Instance class method), 44
 validate() (wtypes.python_types._ForwardSchema method), 27
 validate() (wtypes.python_types.Class class method), 27
 validate() (wtypes.python_types.Instance class method), 28
 validate_generic() (in module wtypes), 30
 validate_generic() (in module wtypes.utils), 30
 validate_object() (wtypes._Implementation method), 30
 validate_object() (wtypes.base._Implementation method), 10
 validate_object() (wtypes.spec method), 30
 validate_object() (wtypes.spec.spec method), 28
 validate_schema() (in module wtypes), 30
 validate_schema() (in module wtypes.utils), 29
 validate_type() (wtypes._Implementation method), 30
 validate_type() (wtypes.base._Implementation method), 10
 validate_type() (wtypes.spec method), 30
 validate_type() (wtypes.spec.spec method), 28
 ValidationError (in module wtypes), 30
 ValidationError (in module wtypes.base), 10

W

websupport2_base_url (in module conf), 3
 wtypes (module), 9
 wtypes.base (module), 9
 wtypes.combining_types (module), 21
 wtypes.content_types (module), 22
 wtypes.dataclass (module), 22
 wtypes.evented (module), 23
 wtypes.examples (module), 26
 wtypes.python_types (module), 26
 wtypes.rdf (module), 28
 wtypes.spec (module), 28
 wtypes.string_formats (module), 28
 wtypes.utils (module), 29
 wtypes.widgets (module), 9
 wtypes.widgets.ipynbwidgets (module), 9

`wtypes_impl` (*class in `wtypes.evented`*), [23](#)