

Stub/Spy Equivalents

<code>t.mock.returns.stub(Foo, "bar")</code>	<code>const barSpy = t.sandbox.stub(Foo, "bar")</code>
<code>t.mock.returns.stub(Foo, "bar")</code>	<code>const barStub = t.sandbox.stub(Foo, "bar")</code>

Behavior Equivalents

<code>.andReturn(foo)</code>	<code>.returns(foo)</code>
------------------------------	----------------------------

If different behavior for different inputs (rare):

<code>.invoke(foo).andReturn(bar)</code>	<code>.withArgs(foo).returns(bar)</code>
<code>.andThrow(foo)</code>	<code>.throws(foo)</code>
<code>.andStub(fooFn)</code>	<code>.callsFake(fooFn)</code>

Stub a property (not a method) (<code>runWithReplacedGlobals</code>)	<code>.value(foo)</code>
--	--------------------------

Verification Equivalents (do you need it?)

t.mocker ...	t.sandbox.assert ...
<code>.on().invoke()</code>	No verification necessary!
<code>.on().invoke(arg1, arg2)</code>	<code>.alwaysCalledWithExactly(fooStub, arg1, arg2)</code>
<code>.expect().invoke()</code>	<code>.calledOnce(fooStub)</code>
<code>.expect().invoke(arg1, arg2)</code>	<code>.calledOnceWithExactly(fooStub, arg1, arg2)</code>
<code>.never()</code>	<code>.notCalled(fooSpy)</code>
<code>.once()</code>	Gives <code>.on()</code> the behavior of <code>.expect()</code>

Matcher Equivalents

<code>Matcher.hasKeyValues({ foo: bar })</code>	<code>Sinon.match({ foo: bar })</code>
---	--

Necessary for most StateObjects (may surface as "Maximum call stack size exceeded"):

<code>Matcher.hasKeyValues({ task: expected_task })</code>	<code>Sinon.match({ task: Sinon.match.same(expected_task) })</code>
--	---

You can also consider matching on e.g. `task.global_id`

<code>Matcher.hasExactKeyValueValues({ foo: bar })</code>	<code>{ foo: bar }</code>
<code>Matcher.any</code>	<code>Sinon.match.any</code>
<code>-</code>	<code>Sinon.match.any</code>
<code>Matcher.hasSubstring("foo")</code>	<code>Sinon.match(/foo/)</code>
<code>Matcher.isInstanceOf(Foo)</code>	<code>Sinon.match.instanceOf(Foo)</code>
<code>Matcher.matchesRegex(/foo/)</code>	<code>Sinon.match(/foo/)</code>
<code>Matcher.isDefined()</code>	<code>Sinon.match.defined</code>
<code>Matcher.isEqual(foo, bar)</code>	<code>Sinon.match.same(foo, bar)</code>
<code>Matcher.isString()</code>	<code>Sinon.match.string</code>
<code>Matcher.isArray()</code>	<code>Sinon.match.array</code>



