RSpec Quick Reference

Core API

Arbitrary Block
```
object.should_satisfy {|arg| ...}
object.should_not_satisfy {|arg| ...}
```

Equality
```
object.should_equal <value>
object.should_not_equal <value>
object.should == <value>
```

Floating Point Comparison
```
object.should_be_close <val>, <delta>
object.should_not_be_close <val>, <delta>
```

Identity
```
object.should_be <value>
object.should_not_be <value>
```

Arbitrary Predicate
```
object.should_predicate [args]
object.should_be_predicate [args]
object.should_not_predicate [args]
object.should_not_be_predicate [args]
object.should_be > <value>
object.should_be < <value>
```

Pattern Matching
```
object.should_match <regex>
object.should_not_match <regex>
object.should =~ <regex>
```

Ancestor Class
```
object.should_be_an_instance_of <class>
object.should_not_be_an_instance_of <class>
object.should_be_a_kind_of <class>
object.should_not_be_a_kind_of <class>
```

Type
```
object.should_respond_to <symbol>
object.should_not_respond_to <symbol>
```

Raising
```
proc.should_raise <exception>
proc.should_not_raise <exception>
proc.should_raise
proc.should_not_raise
```

Throwing
```
proc.should_throw <symbol>
proc.should_not_throw <symbol>
proc.should_not_throw
```

Containment
```
object.should_include <object>
object.should_not_include <object
```

Collection Size
```
object.should_have(<n>).things
object.should_have_at_least(<n>).things
object.should_have_at_most(<n>).things
```

Mock API

Creating a mock
```
mock = mock(<name>)
mock = mock(<name>, <options>)

mock = mock("blah", :null_object => true)
```

Expecting Messages
```
mock.should_receive(:<message>)
mock.should_not_receive(:<message>)
```

Arbitrary Message Handling
```
mock.should_receive(:msg) do | a1 a2 ... |
  ...
end
```

Expecting Arguments
```
.with(<args>)
.with(:no_args)
.with(:any_args)
```

Argument Constraints
```
:anything, :numeric, :boolean, :string,
duck_type(message(s))
```
accepts anything that responds to **all** the message(s):
```
#accepts a Fixnum for the second arg
.with(a, duck_type(:abs, :div))
```

Receive Counts
```
.any_number_of_times
.once
.twice
.exactly(n).times
.at_least(:once)
.at_least(:twice)
.at_least(n).times
```

Return Values
```
.and_return(v)
.and_return(v1, v2, ..., vn)
```
implies consequtive returns & `.at_least(n).times`
```
.and_return {...}
```

Raising and Throwing
```
.and_raise(<exception>)
.and_throw(<symbol>)
```

Ordering
```
mock.should_receive(:flip).once.ordered
mock.should_receive(:flop).once.ordered
```

Command Line Interface

Basic Usage

```
spec file
```
runs all specs in the given file

```
spec directory
```
runs all specs in all ruby files in the given directory

```
spec file/dir-1 file/dir-2 … file/dir-n
```

Running Specific Specifications

```
-s CONTEXT_AND_OR_SPEC_NAME
--spec CONTEXT_AND_OR_SPEC_NAME
```

General Options

```
-v, --version
-h, --help
-d, --dry-run
-r file, --require file
```

Output Formating

```
$spec -s "An empty stack" examples/stack_spec.rb

...

Finished in 0.001903 seconds

1 context, 3 specifications, 0 failures
```

```
-f s
-f specdoc
--format s
--format specdoc

$spec -f s -s "An empty stack"
examples/stack_spec.rb

An empty stack
- should accept an item when sent push
- should complain when sent top
- should complain when sent pop

Finished in 0.003552 seconds

1 context, 3 specifications, 0 failures
```

```
-f r
-f rdoc
--format r
--format rdoc

spec -f r -s "An empty stack"
examples/stack_spec.rb
# An empty stack
# * should accept an item when sent push
# * should complain when sent top
# * should complain when sent pop
```

```
-f <custom formatter class>
--format <custom formatter class>
```

```
-c, --color, --colour
```

Colour the final status line green or red

When There Are Failures

```
$spec failing_examples/mocking_spec.rb

.F

1)
MockExpectationError in 'Mocker should fail when expected message not received'
Mock 'poke me' expected 'poke' once, but received it 0 times
./failing_examples/mocking_spec.rb:13:in `'

Finished in 0.00242 seconds

2 specifications, 1 failures
```

```
spec -f s failing_examples/mocking_spec.rb

Mocker
- should be able to call mock()
- should fail when expected message not received (FAILED - 1)

1)
MockExpectationError in 'Mocker should fail when expected message not received'
Mock 'poke me' expected 'poke' once, but received it 0 times
./failing_examples/mocking_spec.rb:13:in `'
Finished in 0.004135 seconds

2 specifications, 1 failures
```