LeanDog
AN AGILE SOFTWARE STUDIO

RSpec Quick Reference

Source: https://relishapp.com/rspec/rspec-expectations/docs/built-in-matchers

Object identity

```
expect(actual).to be(expected) # passes if actual.equal?(expected)
```

Object equivalence

```
expect(actual).to eq(expected) # passes if actual == expected
```

Optional APIs for identity/equivalence

```
expect(actual).to eql(expected)    # passes if actual.eql?(expected)
expect(actual).to equal(expected) # passes if actual.equal?(expected)
```

Comparisons

```
expect(actual).to be >  expected
expect(actual).to be >= expected
expect(actual).to be <= expected
expect(actual).to be <  expected
expect(actual).to be_between(minimum, maximum).inclusive
expect(actual).to be_between(minimum, maximum).exclusive
expect(actual).to match(/expression/)
expect(actual).to be_within(delta).of(expected)
expect(actual).to start_with expected
expect(actual).to end_with expected
```

Types/classes/response

```
expect(actual).to be_instance_of(expected)
expect(actual).to be_kind_of(expected)
expect(actual).to respond_to(expected)
```

Truthiness and existentialism

```
expect(actual).to be_truthy      # passes if actual is truthy (not nil or false)
expect(actual).to be true        # passes if actual == true
expect(actual).to be_falsey      # passes if actual is falsy (nil or false)
expect(actual).to be false       # passes if actual == false
expect(actual).to be_nil         # passes if actual is nil
expect(actual).to exist          # passes if actual.exist? and/or actual.exists? are truthy
expect(actual).to exist(*args)   # passes if actual.exist?(*args) and/or actual.exists?(*args) are truthy
```

Expecting errors

```
expect { ... }.to raise_error
expect { ... }.to raise_error(ErrorClass)
expect { ... }.to raise_error("message")
expect { ... }.to raise_error(ErrorClass, "message")
```

Expecting throws

```
expect { ... }.to throw_symbol
expect { ... }.to throw_symbol(:symbol)
expect { ... }.to throw_symbol(:symbol, 'value')
```

Predicate matchers

```
expect(actual).to be_xxx             # passes if actual.xxx?
expect(actual).to have_xxx(:arg)  # passes if actual.has_xxx?(:arg)
```

Examples

```
expect([]).to be_empty
expect(:a => 1).to have_key(:a)
```

Collection membership

```
expect(actual).to      include(expected)
expect(array).to       match_array(expected_array)
# ...which is the same as:
expect(array).to       contain_exactly(individual, elements)
```

Examples

```
expect([1, 2, 3]).to            include(1)
expect([1, 2, 3]).to            include(1, 2)
expect(:a => 'b').to            include(:a => 'b')
expect("this string").to        include("is str")
expect([1, 2, 3]).to            contain_exactly(2, 1, 3)
expect([1, 2, 3]).to            match_array([3, 2, 1])
```