Equivalence

```ruby
expect(actual).to eq(expected)  # actual == expected
expect(actual).to eql(expected) # actual.eql?(expected)
expect(actual).not_to eql(not_expected)
```

Identity

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

Comparisons

```ruby
expect(actual).to be >  expected
expect(actual).to be >= expected
expect(actual).to be <= expected
expect(actual).to be <  expected
expect(actual).to be_within(delta).of(expected)
```

Regular expressions

```ruby
expect(actual).to match(/expression/)
```

Truthiness

```ruby
expect(actual).to be_truthy   # actual not nil or false
expect(actual).to be true     # actual == true
expect(actual).to be_falsy    # actual is falsy (nil or false)
expect(actual).to be false    # actual == false
expect(actual).to be_nil      # actual is nil
expect(actual).to_not be_nil  # actual is not nil
```

Types/classes

```ruby
expect(actual).to be_an_instance_of(expected) # actual.class == expected
expect(actual).to be_a(expected)              # actual.kind_of?(expected)
expect(actual).to be_an(expected)             # an alias for be_a
```

Yielding

```ruby
expect { |b| 5.tap(&b) }.to yield_control # passes regardless of yielded args
expect { |b| yield_if_true(true, &b) }.to yield_with_no_args # no args yielded
expect { |b| 5.tap(&b) }.to yield_with_args(5)
expect { |b| 5.tap(&b) }.to yield_with_args(Fixnum)
expect { |b| "a string".tap(&b) }.to yield_with_args(/str/)
expect { |b| [1, 2, 3].each(&b) }.to yield_successive_args(1, 2, 3)
expect { |b| {:a=>1}.each(&b) }.to yield_successive_args([:a, 1])
```

Collection membership

```ruby
expect(actual).to include(expected)
expect(actual).to start_with(expected)
expect(actual).to end_with(expected)
expect(actual).to match_array(expected_array)
```

Ranges (Ruby >= 1.9 only)

```ruby
expect(1..10).to cover(3)
```

Expecting errors

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

Expecting throws

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

Predicate matchers

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

Examples

```ruby
expect([1, 2, 3]).to include(1, 2)
expect([1, 2, 3]).to start_with(1, 2)
expect([1, 2, 3]).to end_with(2, 3)
expect({:a => 'b'}).to include(:a => 'b')
expect("this string").to include("is str")
expect("this string").to start_with("this")
expect("this string").to end_with("ring")
expect("s").to start_with("a").and end_with("b")
expect([1, 2, 3]).to match_array([3, 2, 1])
expect { k += 1 }.to change { k }.by(1)
expect { s = "barn" }.to change { s }
  .from( a_string_matching(/foo/) )
  .to( a_string_matching(/bar/) )
```

RSpec Expectations, v3.5, July 1, '16 (00c895)
https://github.com/rspec/rspec-expectations