Basic Matchers

> eq

`eq` is the most commonly-used matcher and asserts equality.

```
describe Array do
  it 'has two things' do
    expect(%w(thing1 thing2).length).to eq 2
  end
end
```

> match

`match` asserts with a regular expression match.

```
describe User, '.build_with_default_ssn' do
  it 'generates a valid SSN' do
    user_with_default_ssn = User.build_with_default_ssn
    expect(user_with_default_ssn.ssn).to match /^\d{3}-\d{2}-\d{4}$/
  end
end
```

Other Common Matchers

> include

`include` asserts that a value is present in the subject.

```
describe 'dynamics of include' do
  it 'works with arrays' do
    expect([1, 2, 3]).to include 1
  end

  it "works with strings" do
    expect('string').to include 'str'
  end

  it "works with hashes" do
    expect({ foo: :bar }).to include :foo
  end
end
```

> be_true

`be_true` asserts truthiness.

```
describe User, '#admin?' do
  it 'returns true when privileges include :admin' do
    user = User.new(privileges: [:admin])
    expect(user.admin?).to be_true
  end
end
```

> be_false

`be_false` asserts falsiness.

```
describe User, '#admin?' do
  it 'returns false when privileges are empty' do
    user = User.new(privileges: [])
    expect(user.admin?).to be_false
  end
end
```

Exceptions

> raise_error

In exceptional situations, you'll want to raise in Ruby. Testing can be done with `expect`, passing a block, and then calling `raise_error`. The class and error message are optional.

```
describe SocialNetworkNotifier, 'when the network is down' do
  it 'raises when contacting Twitter fails' do
    fake_network_outage

    expect {
      SocialNetworkNotifier.notify_twitter_with(MessageBuilder.generate)
    }.to raise_error(SocialNetworkNotifier::NetworkUnreachableError,
                     'Unable to contact Twitter.')
  end

  it 'raises when contacting Facebook fails' do
    fake_network_outage

    expect {
      SocialNetworkNotifier.notify_facebook_with(MessageBuilder.generate)
    }.to raise_error(SocialNetworkNotifier::NetworkUnreachableError, /Facebook/)
  end
end
```

`raise_error` also works with `not_to`.

```
describe SocialNetworkNotifier, 'when the network is working' do
  it 'does not raise when contacting Twitter' do
    fake_network_success

    expect {
      SocialNetworkNotifier.notify_twitter_with(MessageBuilder.generate)
    }.not_to raise_error
  end
end
```

Predicates

RSpec is kind enough to match predicate methods (methods ending with a ?) with appropriate matchers which read easily.

```
describe 'predicates' do
  it 'works with all Ruby objects' do
    expect(nil).to be_nil
    expect(%w(thing1 thing2)).not_to be_empty
  end
end
```

Predicates work on any Ruby object and any predicate method.

```
describe 'more predicates' do
  it 'works with all Ruby objects' do
    expect(User.new(privileges: [:admin])).to be_admin
    expect(User.new(privileges: [])).not_to be_admin
  end
end
```