## Pull requests

XCPretty is tested with [Cucumber](http://cukes.info) and [RSpec](http://rspec.info).

If you're planning to contribute, please do write tests.

Here's an example workflow for a contribution:

#### 0. Fork the project and create a branch

- Make separate branches for unrelated changes, like `my-awesome-feature` or `fix-terrible-bug`

#### 1. Write a failing feature

- These are a full-stack end to end tests

- You can find features in `features/`. You'll need to write a `feature` and implement it's `steps`.

- Try to reuse as many matchers as possible

- This tests are slower because they're executing `xcpretty` command for each test

Here's an example feature for adding output without UTF8:

``` gherkin

Scenario: Running tests without UTF-8 support

Given I have a passing test in my suite

And I pipe to xcpretty with "--no-utf"

Then I should see a non-utf prefixed output

```

And the steps:

- `Given I have a passing test in my suite`

``` ruby

Given(/^I have a passing test in my suite$/) do

add\_run\_input SAMPLE\_OCUNIT\_TEST

end

```

- `And I pipe to xcpretty with "--no-utf"`

``` ruby

When(/^I pipe to xcpretty with "(.\*?)"$/) do |flags|

run\_xcpretty(flags)

end

```

- `Then I should see a non-utf prefixed output`

``` ruby

Then(/^I should see a non-utf prefixed output$/) do

run\_output.should start\_with(".")

end

```

#### 2. Write a failing BDD test

- These are unit tests, and they're very fast (below 200ms for the entire suite)

- You should be running them continuously with `kicker`, or your awesome Vim binding

#### 3. Implement your awesome contribution

- This should fix unit tests one-by-one, and finally your `feature` will be passing