# Want to contribute?

If you would like to contribute, here are some notes and guidelines:

* All new development should be on feature/fix branches, which are then merged to the master branch once stable and approved; so the master branch is always the most up-to-date, working code
* If you are going to submit a pull request, please fork from master, and submit your pull request back as a fix/feature branch referencing the GitHub issue number
* Install (development) dependencies by running composer install inside your PhpSpreadsheet clone.
* The code must work with all PHP versions that we support.
  + You can call composer versions to test version compatibility.
* Code style should be maintained.
  + composer style will identify any issues with Coding Style.
  + composer fix will fix most issues with Coding Style.
* All code changes must be validated by composer check.
* Please include Unit Tests to verify that a bug exists, and that this PR fixes it.
* Please include Unit Tests to show that a new Feature works as expected.
* Please don't "bundle" several changes into a single PR; submit a PR for each discrete change/fix.
* Remember to update documentation if necessary.
* [Helpful article about forking](https://help.github.com/articles/fork-a-repo/)
* [Helpful article about pull requests](https://help.github.com/articles/using-pull-requests/)

## Unit Tests

When writing Unit Tests, please

* Always try to write Unit Tests for both the happy and unhappy paths.
* Put all assertions in the Test itself, not in an abstract class that the Test extends (even if this means code duplication between tests).
* Include any necessary setup() and tearDown() in the Test itself.
* If you change any global settings (such as system locale, or Compatibility Mode for Excel Function tests), make sure that you reset to the default in the tearDown().
* Use the ExcelError functions in assertions for Excel Error values in Excel Function implementations.

Not only does it reduce the risk of typos; but at some point in the future, ExcelError values will be an object rather than a string, and we won't then need to update all the tests.

* Don't over-complicate test code by testing happy and unhappy paths in the same test.

This makes it easier to see exactly what is being tested when reviewing the PR. I want to be able to see it in the PR, not have to hunt in other unchanged classes to see what the test is doing.

## How to release

1. Complete CHANGELOG.md and commit
2. Create an annotated tag
   1. git tag -a 1.2.3
   2. Tag subject must be the version number, eg: 1.2.3
   3. Tag body must be a copy-paste of the changelog entries.
3. Push the tag with git push --tags, GitHub Actions will create a GitHub release automatically, and the release details will automatically be sent to packagist.
4. By default, Github removes markdown headings in the Release Notes. You can either edit to restore these, or, probably preferably, change the default comment character on your system - git config core.commentChar ";".

**Note:** Tagged releases are made from the master branch. Only in an emergency should a tagged release be made from the release branch. (i.e. cherry-picked hot-fixes.) However, there are 4 branches which have been updated to apply security patches, and those may be tagged if future security updates are needed.

* release1291 (no further updates aside from security patches, including code changes needed for Php 8.5 compatibility)
* release210 (no further updates aside from security patches, including code changes needed for Php 8.5 compatibility)
* release222
* release390