# Contributing to microsoft/react-native-windows

If this is your first time, below are some more detailed steps for getting your branch set up and submitting a PR. You can either do this with the command prompt or with a combination of the command prompt and SourceTree.

### Setting up your branch for changes

For each bug or task you complete, it is recommended that you start with a fresh branch. If you have any lingering changes in your current branch that you want to save, go ahead and commit them. If you are just beginning, then you are good to go. On github, navigate to your repository which should be forked from microsoft/react-native-windows. If you haven't already created a fork, follow the steps in \*\*[Git branch Setup](branch-setup.md)\*\*. Above the list of files is a dropdown that should say master. Use the dropdown to create a new branch and name is according to what you will be working on. (I.e. DropdownHighlight, CleanUpExamples, etc). Now you have created a new branch.

\*\*SourceTree:\*\*

If you are using SourceTree you will want your branch to show up in SourceTree so you can commit changes to your branch. It takes time for it to show up automatically, so you can make it show by running `git remote update origin –prune` in your command prompt from the root. Once you see your new branch in SourceTree under Remotes on the left navigation pane, double click on your branch to check it out locally. A dialog will come up and the default settings should be fine, click Ok.

\*\*Git Command Line\*\*

If you are using the command line, you will want to make sure you have your branch locally. It takes time for it to show up automatically, so you can make it show by running `git remote update origin –prune` in your command prompt from the root. Run `git branch -a` to see if your new branch shows up. Now you will want to check out your branch locally. You can do this with `git checkout -b branch-name`. Confirm you are now working out of the branch with `git branch`.

### Make the fix

Now that your branch is set up and ready for commits, go ahead and fix the bug you are working on or make some small change that you want to check in.

### Verify your changes

Manually test your fix, and ensure that all tests are in a good state

### Commit your changes

\*\*SourceTree:\*\*

In SourceTree, click on commit in the top left. This won't actually do anything to your files, it will just change to show the commit UI. In the bottom container, stage all of the files you want to submit by selecting them and clicking "Stage". Add a short message in the textbox at the bottom on what is included in your change. This will not show as your entire submission text, just for this commit.

\*\*Git Command Line\*\*

To stage files using the command line, you need to run `git add MyFileOne.tsx` for each file. You can also look up how to add all files with changes under a directory. Next you will want to commit changes with `git commit –m "This change updates the padding in the dropdown"`

You can commit multiple times until you are ready to make a pull request. You should keep the message short since it will not be used in the bug notes and is just for keeping track of the multiple commits in one pull request.

### Run auto-format

Run `yarn format` in the repository's root directory if you're making C++ changes.

Run `yarn lint:fix` if making JS/TS changes.

### Run yarn api if you made API changes

If you made API changes, run `yarn api` to update the auto-generated API docs.

### Provide changelog information

Run `yarn change` in the root of the repo.

### Creating a Pull Request

\*\*SourceTree:\*\*

In SourceTree click Push.

\*\*Git Command Line\*\*

Run `git push`.

This will push any staged files you have in your branch.

Now go back to your fork on github. You should see a yellow bar at the top with your change and a button that says "Compare & Pull Request". Click that button.

Click "Create Pull Request".

A bunch of tests will automatically kick off to verify your PR. The tests marked as `required` must pass before a PR can be merged. Currently the Azure Dev Ops tests will not kick off automatically. -- These have to be kicked off by a team member.

Someone will also have to review your change before the change is allowed to be merged in. They may ask questions for more information or ask you to change things. Be sure to respond to their comments and push additional changes to the branch if they ask you to modify things before they sign off.

Now you are done! Celebrate!