# Contributing

Thanks for your interest in contributing to the Ionic Framework! :tada:

- [Contributing Etiquette](#contributing-etiquette)

- [Creating an Issue](#creating-an-issue)

\* [Creating a Good Code Reproduction](#creating-a-good-code-reproduction)

- [Creating a Pull Request](#creating-a-pull-request)

\* [Setup](#setup)

\* [Core](#core)

+ [Modifying Components](#modifying-components)

+ [Preview Changes](#preview-changes)

+ [Lint Changes](#lint-changes)

+ [Modifying Documentation](#modifying-documentation)

+ [Modifying Tests](#modifying-tests)

- [Screenshot Tests](#screenshot-tests)

+ [Building Changes](#building-changes)

\* [Submit Pull Request](#submit-pull-request)

- [Commit Message Guidelines](#commit-message-guidelines)

\* [Commit Message Format](#commit-message-format)

\* [Revert](#revert)

\* [Type](#type)

\* [Scope](#scope)

\* [Subject](#subject)

\* [Body](#body)

\* [Footer](#footer)

\* [Examples](#examples)

- [License](#license)

## Contributing Etiquette

Please see our [Contributor Code of Conduct](https://github.com/ionic-team/ionic/blob/master/CODE\_OF\_CONDUCT.md) for information on our rules of conduct.

## Creating an Issue

\* If you have a question about using the framework, please ask on the [Ionic Forum](http://forum.ionicframework.com/) or in the [Ionic Worldwide Slack](http://ionicworldwide.herokuapp.com/) group.

\* It is required that you clearly describe the steps necessary to reproduce the issue you are running into. Although we would love to help our users as much as possible, diagnosing issues without clear reproduction steps is extremely time-consuming and simply not sustainable.

\* The issue list of this repository is exclusively for bug reports and feature requests. Non-conforming issues will be closed immediately.

\* Issues with no clear steps to reproduce will not be triaged. If an issue is labeled with "needs: reply" and receives no further replies from the author of the issue for more than 14 days, it will be closed.

\* If you think you have found a bug, or have a new feature idea, please start by making sure it hasn't already been [reported](https://github.com/ionic-team/ionic/issues?utf8=%E2%9C%93&q=is%3Aissue). You can search through existing issues to see if there is a similar one reported. Include closed issues as it may have been closed with a solution.

\* Next, [create a new issue](https://github.com/ionic-team/ionic/issues/new/choose) that thoroughly explains the problem. Please fill out the populated issue form before submitting the issue.

## Creating a Good Code Reproduction

### What is a Code Reproduction?

A code reproduction is a small application that is built to demonstrate a particular issue. The code reproduction should contain the minimum amount of code needed to recreate the issue and should focus on a single issue.

### Why Should You Create a Reproduction?

A code reproduction of the issue you are experiencing helps us better isolate the cause of the problem. This is an important first step to getting any bug fixed!

Without a reliable code reproduction, it is unlikely we will be able to resolve the issue, leading to it being closed. In other words, creating a code reproduction of the issue helps us help you.

### How to Create a Reproduction

\* Create a new Ionic application using one of our starter templates. The `blank` starter application is a great choice for this. You can create one using the following Ionic CLI command: `ionic start myApp blank`

\* Add the minimum amount of code needed to recreate the issue you are experiencing. Do not include anything that is not required to reproduce the issue. This includes any 3rd party plugins you have installed.

\* Publish the application on GitHub and include a link to it when [creating an issue](#creating-an-issue).

\* Be sure to include steps to reproduce the issue. These steps should be clear and easy to follow.

### Benefits of Creating a Reproduction

\* \*\*Uses the latest version of Ionic:\*\* By creating a new Ionic application, you are ensuring that you are testing against the latest version of the framework. Sometimes the issues you are experiencing have already been resolved in a newer version of the framework!

\* \*\*Minimal surface area:\*\* By removing code that is not needed in order to reproduce the issue, it makes it easier to identify the cause of the issue.

\* \*\*No secret code needed:\*\* Creating a minimal reproduction of the issue prevents you from having to publish any proprietary code used in your project.

\* \*\*Get help fixing the issue:\*\* If we can reliably reproduce an issue, there is a good chance we will be able to address it.

## Creating a Pull Request

\* We appreciate you taking the time to contribute! Before submitting a pull request, we ask that you please [create an issue](#creating-an-issue) that explains the bug or feature request and let us know that you plan on creating a pull request for it. If an issue already exists, please comment on that issue letting us know you would like to submit a pull request for it. This helps us to keep track of the pull request and make sure there isn't duplicated effort.

\* Looking for an issue to fix? Make sure to look through our issues with the [help wanted](https://github.com/ionic-team/ionic/issues?q=is%3Aopen+is%3Aissue+label%3A%22help+wanted%22) label!

### Setup

1. [Download the installer](https://nodejs.org/) for the LTS version of Node.js. This is the best way to also [install npm](https://blog.npmjs.org/post/85484771375/how-to-install-npm#\_=\_).

2. Fork this repository.

3. Clone your fork.

4. Create a new branch from master for your change.

5. Navigate into the directory of the package you wish to modify (core, angular, etc.).

6. Run `npm install` to install dependencies for this package.

7. Follow the steps for the specific package below.

### Core

#### Modifying Components

1. Locate the component(s) to modify inside `/core/src/components/`.

2. Take a look at the [Stencil Documentation](https://stenciljs.com/docs/introduction/) and other components to understand the implementation of these components.

3. Make your changes to the component. If the change is overly complex or out of the ordinary, add comments so we can understand the changes.

4. [Preview your changes](#preview-changes) locally.

5. [Modify the documentation](#modifying-documentation) if needed.

6. [Run lint](#lint-changes) on the directory and make sure there are no errors.

7. [Build the project](#building-changes).

8. After the build is finished, commit the changes. Please follow the [commit message format](#commit-message-format) for every commit.

9. [Submit a Pull Request](#submit-pull-request) of your changes.

#### Preview Changes

1. Run `npm start` from within the `core` directory.

2. A browser should open at `http://localhost:3333/`.

3. From here, navigate to one of the component's tests to preview your changes.

4. If a test showing your change doesn't exist, [add a new test or update an existing one](#modifying-tests).

5. To test in RTL mode, once you are in the desired component's test, add `?rtl=true` at the end of the url; for example: `http://localhost:3333/src/components/alert/test/basic?rtl=true`.

#### Lint Changes

1. Run `npm run lint` to lint the TypeScript and Sass.

2. If there are lint errors, run `npm run lint.fix` to automatically fix any errors. Repeat step 1 to ensure the errors have been fixed, and manually fix them if not.

3. To lint and fix only TypeScript errors, run `npm run lint.ts` and `npm run lint.ts.fix`, respectively.

4. To lint and fix only Sass errors, run `npm run lint.sass` and `npm run lint.sass.fix`, respectively.

#### Modifying Documentation

1. Locate the `readme.md` file in the component's directory.

2. Modify the documentation \*\*above\*\* the line that says `<!-- Auto Generated Below -->` in this file.

3. To update any of the auto generated documentation below that line, make the relevant changes in the following places:

- `Usage`: update the component's usage examples in the component's `usage/` directory

- `Properties`, `Events`, or `Methods`: update the component's TypeScript file (`\*.tsx`)

- `CSS Custom Properties`: update the component's main Sass file (`\*.scss`)

#### Modifying Tests

1. Locate the test to modify inside the `test/` folder in the component's directory.

2. If a test exists, modify the test by adding an example to reproduce the problem fixed or feature added.

3. If a new test is needed, the easiest way is to copy the `basic/` directory from the component's `test/` directory, rename it, and edit the content in both the `index.html` and `e2e.ts` file (see [Screenshot Tests](#screenshot-tests) for more information on this file).

4. The `preview/` directory is used in the documentation as a demo. Only update this test if there is a bug in the test or if the API has a change that hasn't been updated in the test.

##### Screenshot Tests

1. If the test exists in screenshot, there will be a file named `e2e.ts` in the directory of the test.

2. A screenshot test can be added by including this file and adding one or more `test()` calls that include a call to `page.compareScreenshot()`. See [Stencil end-to-end testing](https://stenciljs.com/docs/end-to-end-testing) and existing tests in `core/` for examples.

3. \*\*Important:\*\* each `test()` should have only one screenshot (`page.compareScreenshot()`) call \*\*or\*\* it should check the expect at the end of each test. If there is a mismatch it will fail the test which will prevent the rest of the test from running, i.e. if the first screenshot fails the remaining screenshot calls would not be called \_unless\_ they are in a separate test or all of the expects are called at the end.

4. To run screenshot locally, use the following command: `npm run test.screenshot`.

- To run screenshot for a specific test, pass the path to the test or a string to search for.

- For example, running all `alert` tests: `npm run test.screenshot alert`.

- Or, running the basic `alert` tests: `npm run test.screenshot src/components/alert/test/basic/e2e.ts`.

#### Building Changes

1. Once all changes have been made and the documentation has been updated, run `npm run build` inside of the `core` directory. This will add your changes to any auto-generated files, if necessary.

2. Review the changes and, if everything looks correct, [commit](#commit-message-format) the changes.

3. Make sure the build has finished before committing. If you made changes to the documentation, properties, methods, or anything else that requires an update to a generate file, this needs to be committed.

4. After the changes have been pushed, publish the branch and [create a pull request](#creating-a-pull-request).

### Submit Pull Request

1. [Create a new pull request](https://github.com/ionic-team/ionic/compare) with the `master` branch as the `base`. You may need to click on `compare across forks` to find your changes.

2. See the [Creating a pull request from a fork](https://help.github.com/articles/creating-a-pull-request-from-a-fork/) GitHub help article for more information.

3. Please fill out the provided Pull Request template to the best of your ability and include any issues that are related.

## Commit Message Guidelines

We have very precise rules over how our git commit messages should be formatted. This leads to readable messages that are easy to follow when looking through the project history. We also use the git commit messages to generate our [changelog](https://github.com/ionic-team/ionic/blob/master/CHANGELOG.md). Our format closely resembles Angular's [commit message guidelines](https://github.com/angular/angular/blob/master/CONTRIBUTING.md#commit).

### Commit Message Format

We follow the [Conventional Commits specification](https://www.conventionalcommits.org/). A commit message consists of a \*\*header\*\*, \*\*body\*\* and \*\*footer\*\*. The header has a \*\*type\*\*, \*\*scope\*\* and \*\*subject\*\*:

```

<type>(<scope>): <subject>

<BLANK LINE>

<body>

<BLANK LINE>

<footer>

```

The \*\*header\*\* is mandatory and the \*\*scope\*\* of the header is optional.

### Revert

If the commit reverts a previous commit, it should begin with `revert: `, followed by the header of the reverted commit. In the body it should say: `This reverts commit <hash>.`, where the hash is the SHA of the commit being reverted.

### Type

If the prefix is `feat`, `fix` or `perf`, it will appear in the changelog. However if there is any [BREAKING CHANGE](#footer), the commit will always appear in the changelog.

Must be one of the following:

\* \*\*feat\*\*: A new feature

\* \*\*fix\*\*: A bug fix

\* \*\*docs\*\*: Documentation only changes

\* \*\*style\*\*: Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc)

\* \*\*refactor\*\*: A code change that neither fixes a bug nor adds a feature

\* \*\*perf\*\*: A code change that improves performance

\* \*\*test\*\*: Adding missing tests

\* \*\*chore\*\*: Changes to the build process or auxiliary tools and libraries such as documentation generation

### Scope

The scope can be anything specifying place of the commit change. Usually it will refer to a component but it can also refer to a utility. For example `action-sheet`, `button`, `css`, `menu`, `nav`, etc. If you make multiple commits for the same component, please keep the naming of this component consistent. For example, if you make a change to navigation and the first commit is `fix(nav)`, you should continue to use `nav` for any more commits related to navigation. As a general rule, if you're modifying a component use the name of the folder.

### Subject

The subject contains succinct description of the change:

\* use the imperative, present tense: "change" not "changed" nor "changes"

\* do not capitalize first letter

\* do not place a period `.` at the end

\* entire length of the commit message must not go over 50 characters

\* describe what the commit does, not what issue it relates to or fixes

\* \*\*be brief, yet descriptive\*\* - we should have a good understanding of what the commit does by reading the subject

### Body

Just as in the \*\*subject\*\*, use the imperative, present tense: "change" not "changed" nor "changes".

The body should include the motivation for the change and contrast this with previous behavior.

### Footer

The footer should contain any information about \*\*Breaking Changes\*\* and is also the place to

reference GitHub issues that this commit \*\*Closes\*\*.

\*\*Breaking Changes\*\* should start with the word `BREAKING CHANGE:` with a space or two newlines. The rest of the commit message is then used for this.

### Examples

Does not appear in the generated changelog:

```

docs(changelog): update steps to update

```

Appears under "Features" header, toast subheader:

```

feat(toast): add 'buttons' property

```

Appears under "Bug Fixes" header, skeleton-text subheader, with a link to issue #28:

```

fix(skeleton-text): use proper color when animated

closes #28

```

Appears under "Performance Improvements" header, and under "Breaking Changes" with the breaking change explanation:

```

perf(css): remove all css utility attributes

BREAKING CHANGE: The CSS utility attributes have been removed. Use CSS classes instead.

```

Appears under "Breaking Changes" with the breaking change explanation:

```

refactor(animations): update to new animation system

BREAKING CHANGE:

Removes the old animation system to use the new Ionic animations.

```

The following commit and commit `667ecc1` do not appear in the changelog if they are under the same release. If not, the revert commit appears under the "Reverts" header.

```

revert: feat(skeleton-text): add animated property

This reverts commit 667ecc1654a317a13331b17617d973392f415f02.

```

## License

By contributing your code to the ionic-team/ionic GitHub Repository, you agree to license your contribution under the MIT license.