# Contributing to TestCafe

TestCafe would not be possible without active support from the community. We appreciate and encourage your contributions, no matter how big or small.

Review our contribution guidelines:

\* [Code of Conduct](#code-of-conduct)

\* [General Discussion](#general-discussion)

\* [Reporting a Problem](#reporting-a-problem)

\* [Code Contribution](#code-contribution)

\* [Contribute to Documentation](#contribute-to-documentation)

\* [Documentation for the Current Functionality](#documentation-for-the-current-functionality)

\* [Documentation for New Features](#documentation-for-new-features)

## Code of Conduct

TestCafe abides by the [Contributor Code of Conduct](CODE\_OF\_CONDUCT.md).

## General Discussion

Join the TestCafe community on Stack Overflow: ask and answer [questions with the TestCafe tag](https://stackoverflow.com/questions/tagged/testcafe).

## Reporting a Problem

If you encounter a bug when using TestCafe, please file an issue in our [GitHub repository](https://github.com/DevExpress/testcafe/issues).

We recommend searching through the existing issues to see if the problem has already been reported or addressed.

When you create a new issue, the template text is automatically added to its body. You should complete all the sections in this template to help us understand the issue you are describing. Missing information could delay the processing time.

## Code Contribution

Follow the steps below when submitting your code.

1. Search the [list of issues](https://github.com/DevExpress/testcafe/issues) to see if there is an issue for the bug or feature you are going to work on or create a new one.

2. If you are going to address an existing issue, check the comment thread to make sure that nobody is working on it at the moment.

3. Leave a comment saying that you are willing to fix this issue, and if possible, provide details on how you are going to do this.

4. Core team members may need to discuss the details of the proposed fix with you. As soon as you get the green light from them,

leave a comment saying that you are currently working on this issue.

5. Fork TestCafe and create a branch in your fork. Name this branch with an issue number, for example `gh852`, `gh853`.

> If you are going to update the documentation follow the steps described in [Contribute to Documentation](#contribute-to-documentation).

6. Commit your changes into the branch.

7. Add regression tests to the appropriate sections if you are fixing a bug. You can find these sections by searching for `Regression` in the code.

Add unit and/or functional tests if you are developing a new functionality.

8. Fetch upstream changes and rebase your branch onto `master`.

9. Run tests before submitting a pull request to ensure that everything works properly.

```sh

gulp test-server

gulp test-functional-local

gulp test-client-local

```

10. Push changes to your fork.

11. Submit a pull request. If you are also updating the documentation, submit a separate pull request as described in [Contribute to Documentation](#contribute-to-documentation).

The pull request name should describe what has been done and contain

the [closes](https://github.com/blog/1506-closing-issues-via-pull-requests) directive

with an appropriate issue number.

## Contribute to Documentation

### Documentation for the Current Functionality

If you want to fix a bug in the current documentation or make an enhancement that relates to the existing functionality, follow the instructions below.

1. Fork TestCafe and create a branch in your fork. Name this branch with an issue number followed by the `docs` postfix, e.g. `gh852-docs`.

2. Commit your changes into the branch.

> Note that links in the documentation should point to `.md` files in the repository, so that the documentation is browsable on GitHub. When the website is built, all links are automatically modified to HTML links.

3. Fetch upstream changes and rebase your branch onto `master`.

4. Run tests before submitting a pull request to ensure that Markdown is styled properly and there is no broken links.

```sh

gulp test-website

```

5. Push changes to your fork.

6. Submit a pull request. Documentation pull requests should have the `[docs]` prefix in their title. This ensures that documentation tests are triggered against these pull requests.

### Documentation for New Features

Documentation for new features is written in a separate branch `new-docs`.

Do the following to fetch this branch and commit to it.

1. Fork TestCafe.

2. Download the `new-docs` branch to your local repository and perform check-out.

```sh

git fetch upstream new-docs:new-docs

git checkout new-docs

```

3. Set the local branch to track the remote `new-docs` branch.

```sh

git branch -u upstream/new-docs

```

4. Create a new branch at the top of `new-docs`.

```sh

git checkout -b new-branch new-docs

```

5. Commit your changes into this branch.

> Note that links in the documentation should point to `.md` files in the repository, so that the documentation is browsable on GitHub. When the website is built, all links are automatically modified to HTML links.

6. Run tests before submitting a pull request to ensure that Markdown is styled properly and there is no broken links.

```sh

gulp test-website

```

7. Push changes to your fork.

8. Open a pull request against `DevExpress:new-docs`. Documentation pull requests should have the `[docs]` prefix in their title. This ensures that documentation tests are triggered against these pull requests.