# Contributing to Web Template Studio

The foundation of \_Web Template Studio\_ is to get developers up and running with a new

web app that incorporates cloud services on Azure as fast as possible.

Using Web Template Studio, a developer can select their preferred web

frameworks, pages and Azure cloud services to generate boilerplate code for their web app.

The generated code not only follows best practices and design guidelines, but also

includes comments to guide the developer through the logic and adding functionality to

the code.

Pull Requests must be done against the \*\*[dev branch](https://github.com/Microsoft/WebTemplateStudio/tree/dev)\*\*.

You should also rebase off dev before the PR as well.

## Legal

This project welcomes contributions and suggestions. Most contributions require you to agree to a

Contributor License Agreement (CLA) declaring that you have the right to, and actually do, grant us

the rights to use your contribution. For details, visit https://cla.microsoft.com.

When you submit a pull request, a CLA-bot will automatically determine whether you need to provide

a CLA and decorate the PR appropriately (e.g., label, comment). Simply follow the instructions

provided by the bot. You will only need to do this once across all repos using our CLA.

This project has adopted the [Microsoft Open Source Code of Conduct](https://opensource.microsoft.com/codeofconduct/).

For more information see the [Code of Conduct FAQ](https://opensource.microsoft.com/codeofconduct/faq/) or

contact [opencode@microsoft.com](mailto:opencode@microsoft.com) with any additional questions or comments.

## Before you begin

While we're grateful for any and all contributions, we don't want you to waste anyone's time. Please consider the following points before you start working on any contributions.

- Please comment on an issue to let us know you're interested in working on something before you start the work. Not only does this avoid multiple people unexpectedly working on the same thing at the same time but it enables us to make sure everyone is clear on what should be done to implement any new functionality. It's less work for everyone, in the long run, to establish this up front.

- The code that is outputted in the generated projects may end up in thousands of apps so it must be of the highest quality. Expect it to be reviewed very thoroughly. It must meet our standards for style, structure, and format. There are details below and automated tests to verify their use.

- Get familiar with the automated tests that are part of the project. With so many possible combinations of output, it's impossible to verify everything manually. You will need to make sure they all pass.

- When adding anything new it should be created to work with all supported frameworks. If this is going to be a problem, discuss it before beginning work.

- New features and services shouldn't break or hide old features and services. For ex: Adding

SQL server support shouldn't make Cosmos DB harder to use or inaccessible. New features

should also not break functionality on one or more of the supported operating systems.

## A good pull request

Every contribution has to come with:

- Before starting coding, \*\*you must open an issue\*\* and start discussing with the community to see if the idea/feature is interesting enough.

- A documentation page in the [documentation folder](https://github.com/Microsoft/WebTemplateStudio/tree/release/docs) if applicable.

- Unit tests (If applicable, or an explanation why they're not)

- If you've changed the UI:

- Be sure you are including screenshots to show the changes.

- Be sure you have reviewed the [web accessibility standard](https://www.w3.org/WAI/standards-guidelines/wcag/).

- If you've included a new template, make sure it meets the [web accessibility standard](https://www.w3.org/WAI/standards-guidelines/wcag/) and is compatible with other templates.

- You've run all existing tests to make sure you've not broken anything. Manual testing

must be done against all PRs following the testing plan while unit tests are not in place.

- PR has to target dev branch.

PR has to be validated by at least two core members before being merged. Once merged, it will be in the pre-release package. To find out more, head to [Installing / Using the extension](docs/getting-started-developers.md).

## Quality insurance for pull requests for updating the Wizard

We encourage developers to consider the following guidelines when submitting pull requests for the wizard:

- The Wizard must be usable and efficient with keyboard only.

- Tab order must be logical.

- Focused controls must be visible.

- Action must be triggered when hitting Enter key.

- Do not use custom colors but instead rely on VSCode theme colors. High contrast themes

should be usable with your controls.

## Naming and styling conventions

All code that gets merged in must be formatted with [Prettier](https://prettier.io/). Also,

we use [TSLint](https://palantir.github.io/tslint/) to lint our TypeScript code and [ESLint](https://eslint.org/)

for JavaScript.

## Documentation

- DO provide great documentation with all new features and code.

- DO use readable and self-documenting identifier names.

- DO use consistent naming and terminology.

- DO provide strongly typed APIs.

- DO use verbose identifier names.

## Files and folders

- DO associate no more than one class per file.

- DO use folders to group classes based on features.