Contributing to Pico

====================

Pico aims to be a high quality Content Management System (CMS) but at the same time wants to give contributors freedom when submitting fixes or improvements.

By contributing to Pico, you accept and agree to the \*Developer Certificate of Origin\* for your present and future contributions submitted to Pico. Please refer to the \*Developer Certificate of Origin\* section below.

Aside from this, we want to \*encourage\*, but not obligate you, the contributor, to follow the following guidelines. The only exception to this are the guidelines elucidated in the \*Prevent `merge-hell`\* section. Having said that: we really appreciate it when you apply the guidelines in part or wholly as that will save us time which, in turn, we can spend on bugfixes and new features.

Issues

------

If you want to report an \*issue\* with Pico's core, please create a new [Issue](https://github.com/picocms/Pico/issues) on GitHub. Concerning problems with plugins or themes, please refer to the website of the developer of this plugin or theme.

Before creating a [new Issue on GitHub](https://github.com/picocms/Pico/issues/new), please make sure the problem wasn't reported yet using [GitHubs search engine](https://github.com/picocms/Pico/search?type=Issues).

Please describe your issue as clear as possible and always include the \*Pico version\* you're using. Provided that you're using \*plugins\*, include a list of them too. We need information about the \*actual and expected behavior\*, the \*steps to reproduce\* the problem, and what steps you have taken to resolve the problem by yourself (i.e. \*your own troubleshooting\*).

Contributing

------------

Once you decide you want to contribute to \*Pico's core\* (which we really appreciate!) you can fork the project from https://github.com/picocms/Pico. If you're interested in developing a \*plugin\* or \*theme\* for Pico, please refer to the [development section](http://picocms.org/development/) of our website.

### Developer Certificate of Origin

By contributing to Pico, you accept and agree to the following terms and conditions for your present and future contributions submitted to Pico. Except for the license granted herein to Pico and recipients of software distributed by Pico, you reserve all right, title, and interest in and to your contributions. All contributions are subject to the following DCO + license terms.

```

Developer Certificate of Origin

Version 1.1

Copyright (C) 2004, 2006 The Linux Foundation and its contributors.

1 Letterman Drive

Suite D4700

San Francisco, CA, 94129

Everyone is permitted to copy and distribute verbatim copies of this

license document, but changing it is not allowed.

Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I

have the right to submit it under the open source license

indicated in the file; or

(b) The contribution is based upon previous work that, to the best

of my knowledge, is covered under an appropriate open source

license and I have the right under that license to submit that

work with modifications, whether created in whole or in part

by me, under the same open source license (unless I am

permitted to submit under a different license), as indicated

in the file; or

(c) The contribution was provided directly to me by some other

person who certified (a), (b) or (c) and I have not modified

it.

(d) I understand and agree that this project and the contribution

are public and that a record of the contribution (including all

personal information I submit with it, including my sign-off) is

maintained indefinitely and may be redistributed consistent with

this project or the open source license(s) involved.

```

All contributions to this project are licensed under the following MIT License:

```

Copyright (c) <YEAR> <COPYRIGHT HOLDER>

Permission is hereby granted, free of charge, to any person

obtaining a copy of this software and associated documentation

files (the "Software"), to deal in the Software without

restriction, including without limitation the rights to use,

copy, modify, merge, publish, distribute, sublicense, and/or sell

copies of the Software, and to permit persons to whom the

Software is furnished to do so, subject to the following

conditions:

The above copyright notice and this permission notice shall be

included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,

EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES

OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND

NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT

HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,

WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING

FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR

OTHER DEALINGS IN THE SOFTWARE.

```

Please note that developing a \*plugin\* or \*theme\* for Pico is \*not\* assumed to be a contribution to Pico itself. By developing a plugin or theme you rather create a 3rd-party project that just uses Pico. Following the spirit of open source, we want to \*encourage\* you to release your plugin or theme under the terms of a [OSI-approved open source license](https://opensource.org/licenses). After all, Pico is open source, too!

### Prevent `merge-hell`

Please do \*not\* develop your contribution on the `master` branch of your fork, but create a separate feature branch, that is based off the `master` branch, for each feature that you want to contribute.

> Not doing so means that if you decide to work on two separate features and place a pull request for one of them, that the changes of the other issue that you are working on is also submitted. Even if it is not completely finished.

To get more information about the usage of Git, please refer to the [Pro Git book](https://git-scm.com/book) written by Scott Chacon and/or [this help page of GitHub](https://help.github.com/articles/using-pull-requests).

### Pull Requests

Please keep in mind that pull requests should be small (i.e. one feature per request), stick to existing coding conventions and documentation should be updated if required. It's encouraged to make commits of logical units and check for unnecessary whitespace before committing (try `git diff --check`). Please reference issue numbers in your commit messages where appropriate.

### Coding Standards

Pico uses the [PSR-2 Coding Standard](http://www.php-fig.org/psr/psr-2/) as defined by the [PHP Framework Interoperability Group (PHP-FIG)](http://www.php-fig.org/).

For historical reasons we don't use formal namespaces. Markdown files in the `content-sample` folder (the inline documentation) must follow a hard limit of 80 characters line length.

It is recommended to check your code using [PHP\_CodeSniffer](https://github.com/squizlabs/PHP\_CodeSniffer) using Pico's `.phpcs.xml` standard. Use the following command:

$ ./vendor/bin/phpcs --standard=.phpcs.xml [file]...

With this command you can specify a file or folder to limit which files it will check or omit that argument altogether, in which case the current working directory is checked.

### Keep documentation in sync

Pico accepts the problems of having redundant documentation on different places (concretely Pico's inline user docs, the `README.md` and the website) for the sake of a better user experience. When updating the docs, please make sure to keep them in sync.

If you update the [`README.md`](https://github.com/picocms/Pico/blob/master/README.md) or [`content-sample/index.md`](https://github.com/picocms/Pico/blob/master/content-sample/index.md), please make sure to update the corresponding files in the [`\_docs`](https://github.com/picocms/picocms.github.io/tree/master/\_docs/) folder of the `picocms.github.io` repo (i.e. [Pico's website](http://picocms.org/docs/)) and vice versa. Unfortunately this involves three (!) different markdown parsers. If you're experiencing problems, use Pico's [`erusev/parsedown-extra`](https://github.com/erusev/parsedown-extra) as a reference. You can try to make the contents compatible to [Kramdown](http://kramdown.gettalong.org/) (Pico's website) and [CommonMarker](https://github.com/gjtorikian/commonmarker) (`README.md`) by yourself, otherwise please address the issues in your pull request message and we'll take care of it.

Versioning

----------

Pico follows [Semantic Versioning 2.0](http://semver.org) and uses version numbers like `MAJOR`.`MINOR`.`PATCH`. We will increment the:

- `MAJOR` version when we make incompatible API changes,

- `MINOR` version when we add functionality in a backwards-compatible manner, and

- `PATCH` version when we make backwards-compatible bug fixes.

For more information please refer to the http://semver.org website.

Branching

---------

The `master` branch contains the current development version of Pico. It is likely \*unstable\* and \*not ready for production use\*.

However, the `master` branch always consists of a deployable (but not necessarily deployed) version of Pico. As soon as development of a new `MAJOR` or `MINOR` release starts, a separate branch (e.g. `pico-1.1`) is created and a [pull request](https://github.com/picocms/Pico/pulls) is opened to receive the desired feedback.

Pico's actual development happens in separate development branches. Development branches are prefixed by:

- `feature/` for bigger features,

- `enhancement/` for smaller improvements, and

- `bugfix/` for non-trivial bug fixes.

As soon as development reaches a point where feedback is appreciated, a pull request is opened. After some time (very soon for bug fixes, and other improvements should have a reasonable feedback phase) the pull request is merged and the development branch will be deleted. Trivial bug fixes that will be part of the next `PATCH` version will be merged directly into `master`.

Build & Release process

-----------------------

We're using [Travis CI](https://travis-ci.com) to automate the build & release process of Pico. It generates and deploys a [PHP class documentation](http://picocms.org/phpDoc/) (powered by [phpDoc](http://phpdoc.org)) for new releases and on every commit to the `master` branch. Travis also prepares new releases by generating Pico's pre-built release packages, a version badge, code statistics (powered by [cloc](https://github.com/AlDanial/cloc)) and updates our website (the [`picocms.github.io` repo](https://github.com/picocms/picocms.github.io)). Please refer to our [`.travis.yml`](https://github.com/picocms/Pico/blob/master/.travis.yml), the [`picocms/ci-tools` repo](https://github.com/picocms/ci-tools) and the [`.build` directory](https://github.com/picocms/Pico/tree/master/.build) for details.

As insinuated above, it is important that each commit to `master` is deployable. Once development of a new Pico release is finished, trigger Pico's build & release process by pushing a new Git tag. This tag should reference a (usually empty) commit on `master`, which message should adhere to the following template:

```

Version 1.0.0

\* [Security] ...

\* [New] ...

\* [Changed] ...

\* [Fixed] ...

\* [Removed] ...

```

Before pushing a new Git tag, make sure to update the `Pico::VERSION` and `Pico::VERSION\_ID` constants. The versions of Pico's official [default theme](https://github.com/picocms/pico-theme) and the [`PicoDeprecated` plugin](https://github.com/picocms/pico-deprecated) both strictly follow Pico's version. Since Pico's pre-built release package contains them, you must always create a new release of them (even though nothing has changed) before creating a new Pico release.

If you're pushing a new major or minor release of Pico, you should also update Pico's `composer.json` to require the latest minor releases of Pico's dependencies. Besides, don't forget to update the `@version` tags in the PHP class docs.

Travis CI will draft the new [release on GitHub](https://github.com/picocms/Pico/releases) automatically, but will require you to manually amend the descriptions formatting. The latest Pico version is always available at https://github.com/picocms/Pico/releases/latest, so please make sure to publish this URL rather than version-specific URLs. [Packagist](http://packagist.org/packages/picocms/pico) will be updated automatically.

Labeling of Issues & Pull Requests

----------------------------------

Pico makes use of GitHub's label and milestone features, to aide developers in quickly identifying and prioritizing which issues need to be worked on. The starting point for labeling issues and pull requests is the `type` label, which is explained in greater detail below. The `type` label might get combined with a `pri` label, describing the issue's priority, and a `status` label, describing the current status of the issue.

Issues and pull requests labeled with `info: Feedback Needed` indicate that feedback from others is highly appreciated. We always appreciate feedback at any time and from anyone, but when this label is present, we explicitly \*ask\* you to give feedback. It would be great if you leave a comment!

- The `type: Bug` label is assigned to issues or pull requests, which have been identified as bugs or security issues in Pico's core. It might get combined with the `pri: High` label, when the problem was identified as security issue, or as a so-called "show stopper" bug. In contrast, uncritical problems might get labeled with `pri: Low`. `type: Bug` issues and pull requests are usually labeled with one of the following `status` labels when being closed:

- `status: Resolved` is used when the issue has been resolved.

- `status: Conflict` indicates a conflict with another issue or behavior of Pico, making it impossible to resolve the problem at the moment.

- `status: Won't Fix` means, that there is indeed a problem, but for some reason we made the decision that resolving it isn't reasonable, making it intended behavior.

- `status: Rejected` is used when the issue was rejected for another reason.

- The `type: Enhancement` and `type: Feature` labels are used to tag pull requests, which introduce either a comparatively small enhancement, or a "big" new feature. As with the `type: Bug` label, they might get combined with the `pri: High` or `pri: Low` labels to indicate the pull request's priority. If a pull request isn't mergeable at the moment, it is labeled with `status: Work In Progress` until development of the pull request is finished. After merging or closing the pull request, it is labeled with one of the `status` labels as described above for the `type: Bug` label.

- The `type: Idea` label is similar to the `type: Enhancement` and `type: Feature` labels, but is used for issues or incomplete and abandoned pull requests. It is otherwise used in the exact same way as `type: Enhancement` and `type: Feature`.

- The `type: Release` label is used in the exact same way as `type: Feature` and indicates the primary pull request of a new Pico release (please refer to the \*Branching\* and \*Build & Release process\* sections above).

- The `type: Notice`, `type: Question` and `type: Discussion` labels are used to indicate "fyi" issues, issues opened by users or developers asking questions, and issues with disucssions about arbitrary topics related to Pico. They are neither combined with `pri` labels, nor with `status` labels.

- The `type: Duplicate` label is used when there is already another issue or pull request related to this problem or feature request. Issues labeled with `type: Duplicate` are immediately closed.

- The `type: Invalid` label is used for everything else, e.g. issues or pull requests not related to Pico, or invalid bug reports. This includes supposed bug reports that concern actually intended behavior.

The `status: Deferred` label might get added to any open issue or pull request to indicate that it is still unresolved and will be resolved later. This is also true for the `info: Pinned` label: It indicates a important issue or pull request that remains open on purpose.

After resolving a issue, we usually keep it open for about a week to give users some more time for feedback and further questions. This is especially true for issues with the `type: Notice`, `type: Question`, `type: Discussion` and `type: Invalid` labels. After 7 days with no interaction, [Probot](https://probot.github.io/)'s [Stale](https://github.com/apps/stale) bot adds the `info: Stale` label to the issue to ask the participants whether the issue has been resolved. If no more activity occurs, the issue will be automatically closed by Stale bot 2 days later.

Issues and pull requests labeled with `info: Information Needed` indicate that we have asked one of the participants for further information and didn't receive any feedback yet. It is usually added after Stale bot adds the `info: Stale` label to give the participants some more days to give the necessary information.

Issues and pull requests, which are rather related to upstream projects (i.e. projects Pico depends on, like Twig), are additionally labeled with `info: Upstream`.

When a issue or pull request isn't directly related to Pico's core, but the project as a whole, it is labeled with `info: Meta`. Issues labeled with `info: Website` are related to [Pico's website](http://picocms.org), however, in this case it is usually expedient to move the issue to the [`picocms.github.io` repo](https://github.com/picocms/picocms.github.io) instead. The same applies to the `info: Pico CMS for Nextcloud` label; these issues are related to [Pico CMS for Nextcloud](https://apps.nextcloud.com/apps/cms\_pico).