Contributing to RealWorld

We would love for you to contribute to RealWorld and help make it even better than it is today! As a contributor, here are the guidelines we would like you to follow:

- Code of Conduct
- Question or Problem?
- Issues and Bugs
- Feature Requests
- Submission Guidelines
- Coding Rules
- Commit Message Guidelines

Code of Conduct

Help us keep RealWorld open and inclusive. Please read and follow our Code of Conduct.

Got a Question or Problem?

Do not open issues for general support questions as we want to keep GitHub issues for bug reports and feature requests.
For open discussions, we encourage you to use the Github Discussions channels.

Interested in creating Conduit for your framework?

To create an official implementation of Conduit, check out our Github Discussions and see if anyone else has requested and/or is already working on your framework. If not, feel free to start working on one!

Start here!

Found a Bug?

If you find a bug in the project, you can help us by submitting an issue to our GitHub Repository. Even better, you can submit a Pull Request with a fix.

Missing a Feature?

You can *request* a new feature by submitting an issue to our GitHub repository.

If you would like to *implement* a new feature, please submit an issue with a proposal for your work **FIRST**, to be sure that we can use it. Please consider what kind of change it is:

- For a **Major Feature**, first open an issue and outline your proposal so that it can be discussed. This will also allow us to better coordinate our

efforts, prevent duplication of work, and help you to craft the change so that it is successfully accepted into the project.
- **Small Features** can be crafted and directly submitted as a Pull Request.

Submission Guidelines

Submitting an Issue

Before you submit an issue, please search the issue tracker, maybe an issue for your problem already exists and the discussion might inform you of workarounds readily available.

You can file new issues by selecting from our new issue templates and filling out the issue template.

Submitting a Pull Request (PR)

Before you submit your Pull Request (PR) consider the following guidelines:

1. Search GitHub for an open or closed PR that relates to your submission. You don't want to duplicate effort.
2. Be sure that an issue describes the problem you're fixing, or documents the design for the feature you'd like to add. Discussing the design up front helps to ensure that we're ready to accept your work.
3. Fork the gothinkster/realworld repo.
4. Make your changes in a new git branch:

   ```
   git checkout -b my-fix-branch master
   ```

5. Create your patch.
6. Commit your changes using a descriptive commit message that follows our commit message conventions.
7. Push your branch to GitHub:

   ```
   git push origin my-fix-branch
   ```

8. In GitHub, send a pull request to `realworld:master`.

- If we suggest changes then:
  - Make the required updates.
  - Rebase your branch and force push to your GitHub repository (this will update your Pull Request):

    ```
    git rebase master -i
    git push -f
    ```

That's it! Thank you for your contribution!

After your pull request is merged After your pull request is merged, you can safely delete your branch and pull the changes from the master (upstream) repository:

- Delete the remote branch on GitHub either through the GitHub web UI or your local shell as follows:

  ```
  git push origin --delete my-fix-branch
  ```

- Check out the master branch:

  ```
  git checkout master -f
  ```

- Delete the local branch:

  ```
  git branch -D my-fix-branch
  ```

- Update your master with the latest upstream version:

  ```
  git pull --ff upstream master
  ```

Commit Message Guidelines

These guidelines have been added to the project starting from

We have very precise rules over how our git commit messages can be formatted. This leads to **more readable messages** that are easy to follow when looking through the **project history**.

Commit Message Format

Each commit message consists of a **header**, a **body** and a **footer**. The header has a special format that includes a **type**, a **scope** and a **subject**:

```
<type>(<scope>): <subject>
<BLANK LINE>
<body>
<BLANK LINE>
<footer>
```

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

Any line of the commit message cannot be longer 100 characters! This allows the message to be easier to read on GitHub as well as in various git tools.

The footer should contain a closing reference to an issue if any.

Samples:

```
docs(changelog): update changelog to beta.5
```

```
fix(release): need to depend on latest ng-lib

The version in our package.json gets copied to the one we publish, and users need the latest
```

Type

Must be one of the following: - **docs**: Documentation only changes - **feat**: A new feature - **fix**: A bug fix

Scope

The scope should be the name of the npm package affected (as perceived by the person reading the changelog generated from commit messages).

The following is the list of supported scopes:

- **specs**
- **project**

Subject

The subject contains a succinct description of the change:

- use the imperative, present tense: “change” not “changed” nor “changes”
- don’t capitalize the first letter
- no dot (.) at the end

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.

Samples :

```
Close #394
```

```
BREAKING CHANGE:
change login route to /users/login
```