---

options:

order: 0

tags:

- contributing

---

# Welcome Contributors

? First of all, we'd love for you to contribute to our source code. ?

## Code of Conduct

We believe in a welcoming and inclusive environment / community, that's why we enforce our Code of Conduct.

See the [Contributor Covenant Code of Conduct](CODE\_OF\_CONDUCT.md) for more information.

## Table of Contents

\* [Link collection](#short-collection-of-links-to-important-resources)

\* [Setup for contributors](#setup-for-contributors)

\* [Issues](#issues)

\* [Good first contribution](#good-first-issue)

\* [Label structure](#labels)

\* [Working on issues](#working-on-issues)

\* [Style Guide / Coding conventions](#style-guide--coding-conventions)

\* [Coding conventions & guidelines](#coding-conventions--guidelines)

## Short collection of Links to Important Resources

\* bugs / feature requests: https://github.com/meetalva/alva/issues

\* website: https://meetalva.io

\* sprint board: https://github.com/meetalva/alva/projects/6

\* roadmap: https://github.com/meetalva/alva/projects/7

## Setup for contributors

To add a contribution to Alva (improve its features, fix issues),

get the source of the application itself, and if you don't already have a Alva compatible styleguide, the kickstart styleguide (alva-design):

1. Clone the [alva](https://github.com/meetalva/alva) repository:

```shell

git clone git@github.com:meetalva/alva.git

```

2. Run in the main (Alva) repository:

```shell

yarn

yarn tsc -b -w

# In a second terminal window

yarn webpack -w

# In a third terminal window

yarn alva

```

3. Now open a compatible styleguide.

If you don't have a compatible styleguide, follow the next steps:

1. Clone the [Alva Design](https://github.com/meetalva/alva-design) repository:

```shell

git clone git@github.com:meetalva/alva-design.git

```

2. Run this command to build the Alva Design Library for Alva:

```shell

npm i && npm build

```

3. Now you should be able to open the Alva Design Library inside Alva

## Typedoc

To generate a Typedoc documentation of all classes, run

```

npm run docs

```

then open [build/docs/globals.html](build/docs/globals.html) in your browser.

## Issues

### Good first contribution

A good first contribution could be

\* reporting a bug

\* create a feature request

\* work on an issue with the label [good first issue](https://github.com/meetalva/alva/labels/good%20first%20issue)

### Labels

\* \_good first issue\_ - if you are new to \_Alva\_ or open source this could be a good first issue

\* \_help wanted\_ - issue where we need help

\* \_type: bug\_ - issue for an error/ failure of Alva

\* \_type: feature\_ - issue for feature request

\* \_type: design\_ - issue for a design todo

\* \_type: documentation\_ - issue for needed, missing or unclear documentation

\* \_type: feedback\_ - issue for feedback and task where we need to figure out if it's a bug or we want it as a feature (or something else)

\* \_type: question\_ - issue for open question (most of the time, they need to be answered by core team)

\* \_status: in progress\_ - issue is currently in progress

\* \_status: has PR\_ - issue has already an open pull request

\* \_priority: low\_ - issue which is a nice to have, like a backlog issue

\* \_priority: mid\_ - issue needed in the near future, but not super urgent

\* \_priority: high\_ - issue must be resolved as soon as possible

### Working on issues

Issues should be provided as branches starting with "feat/", as pull requests.

## Coding conventions & guidelines

### TypeScript & TSLint

We use strict tsconfig and TSLint rules, as well as Prettier. When it comes to naming things, we follow the [TypeScript guidelines](https://github.com/Microsoft/TypeScript/wiki/Coding-guidelines), and we do not abbreviate names (except for trivial names like HTML).

### We use .editorconfig

Editorconfig helps developers define and maintain consistent coding styles between different editors and IDEs.

There are a lot of plugins for editors/IDEs that support .editorconfig.

You'll find more information, extensions for editors and IDEs at the [editorconfig page](http://editorconfig.org/).

## Commit message conventions

For commit messages we use the [angular commit guidelines](https://github.com/angular/angular.js/blob/master/DEVELOPERS.md#commits) with one exception: Please put the GitHub issue reference into the header, e.g.

```

feat(store): patterns path configurable (fixes #214)

```

As tools, we use [commitlint](https://github.com/marionebl/commitlint) and [commitizen](https://github.com/commitizen/cz-cli).

## Source-code structure and architecture

Alva is a React application using MobX as state management and Electron to provide a stand-alone application.

Additionally, it ships with a living styleguide project, which consists of Patternplate React components.

All the sources of Alva are located in `src`, divided into the following folders:

\* \*\*components\*\*: All React components for the project, page, page-element, and property panes, as well as the design preview in the middle of the page. Components may be smart (they may contain their own state), but only as long as the state is nothing global, related to multiple components, or fundamental enough. In this case, the state is maintained by the store (see below).

\* \*\*electron\*\*: The bootstrap code to start the Electron App, including the container HTML, and the main menu.

\* \*\*lsg\*\*: The styled components Alva uses, as a living styleguide. Do not mixup this styleguide with the designkit (which also is a styleguide). The LSG contains the styled, logic-less components which are used by the Alva UI. The Alva Design Library contains the patterns the designer uses to create a basic design (wireframes).

\* \*\*resources\*\*: Resources are files related to the build, like the icons.

\* \*\*store\*\*: The store is the data-center and business logic of Alva. It is a collection of MobX observables and does not contain any UI elements like React components. Instead, all components bind their props to this store by decorating with `@MobX.observer`. The store contains the Alva projects and pages the user edits, as well as the styleguide and styleguide analyzers, the logic to interpret your frontend pattern components.

## Team

\* [Alex Peschel](http://github.com/Alexpeschel)

\* [Alexander Bokov](http://github.com/alxbok)

\* [Daniel Gooß](http://github.com/Dangoo)

\* [Felicitas Kugland](http://github.com/kotzendekrabbe)

\* [Frederik Reiss](http://github.com/frederikreiss)

\* [Gregor Adams](http://github.com/pixelass)

\* [Julian Cebulla](http://github.com/Jumace)

\* [Lasse Küchler](http://github.com/LKuechler)

\* [Luca Oelsner](http://github.com/lucoel)

\* [Mario Nebl](http://github.com/marionebl)

\* [Markus Ölhafen](http://github.com/markusoelhafen)

\* [Thomas Jacob](http://github.com/TheReincarnator)

\* [Tilman Frick](http://github.com/tilmx)

## Questions?

Just drop us an [email](hey@meetalva.io) or write a direct message on Twitter to [@meetalva](https://twitter.com/meetalva)