Contributing to Leaflet

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

1. [Getting Involved](#getting-involved)

2. [Reporting Bugs](#reporting-bugs)

3. [Contributing Code](#contributing-code)

4. [Running the Tests](#running-the-tests)

6. [Improving Documentation](#improving-documentation)

7. [Code of Conduct](#code-of-conduct)

8. [Thank You](#thank-you)

## Getting Involved

Third-party patches are absolutely essential in our quest to create the best mapping library that will ever exist.

However, they're not the only way to get involved with Leaflet development.

You can help the project tremendously by discovering and [reporting bugs](#reporting-bugs);

[improving documentation](#improving-documentation);

helping others on [Stack Overflow](https://stackoverflow.com/questions/tagged/leaflet),

[GIS Stack Exchange](https://gis.stackexchange.com/questions/tagged/leaflet)

and [GitHub issues](https://github.com/Leaflet/Leaflet/issues);

tweeting to [@LeafletJS](http://twitter.com/LeafletJS);

and spreading the word about Leaflet among your colleagues and friends.

## Reporting Bugs

Before reporting a bug on the project's [issues page](https://github.com/Leaflet/Leaflet/issues),

first make sure that your issue is caused by Leaflet, not your application code

(e.g. passing incorrect arguments to methods, etc.).

Second, search the already reported issues for similar cases,

and if it's already reported, just add any additional details in the comments.

After you've made sure that you've found a new Leaflet bug,

here are some tips for creating a helpful report that will make fixing it much easier and quicker:

\* Write a \*\*descriptive, specific title\*\*. Bad: \*Problem with polylines\*. Good: \*Doing X in IE9 causes Z\*.

\* Include \*\*browser, OS and Leaflet version\*\* info in the description.

\* Create a \*\*simple test case\*\* that demonstrates the bug (e.g. using [Leaflet plunker](http://leafletjs.com/edit.html)).

\* Check whether the bug can be reproduced in \*\*other browsers\*\*.

\* Check if the bug occurs in the stable version, master, or both.

\* \*Bonus tip:\* if the bug only appears in the master version but the stable version is fine,

use `git bisect` to find the exact commit that introduced the bug.

If you just want some help with your project,

try asking on [Stack Overflow](https://stackoverflow.com/questions/tagged/leaflet)

or [GIS Stack Exchange](https://gis.stackexchange.com/questions/tagged/leaflet) instead.

## Contributing Code

### Considerations for Accepting Patches

While we happily accept patches, we're also committed to keeping Leaflet simple, lightweight and blazingly fast.

So bugfixes, performance optimizations and small improvements that don't add a lot of code

are much more likely to get accepted quickly.

Before sending a pull request with a new feature, check if it's been discussed before already

on [GitHub issues](https://github.com/Leaflet/Leaflet/issues)

and ask yourself two questions:

1. Are you sure that this new feature is important enough to justify its presence in the Leaflet core?

Or will it look better as a plugin in a separate repository?

2. Is it written in a simple, concise way that doesn't add bulk to the codebase?

If your feature or API improvement did get merged into master,

please consider submitting another pull request with the corresponding [documentation update](#improving-documentation).

### Setting up the Build System

The Leaflet build system uses [NodeJS](http://nodejs.org/).

To set up the Leaflet build system, install [NodeJS](https://nodejs.org/).

Then run the following commands in the project root to install dependencies:

```

npm install

```

or, if you prefer [`yarn`](https://yarnpkg.com/) over `npm`:

```

yarn install

```

### Making Changes to Leaflet Source

If you're not yet familiar with the way GitHub works (forking, pull requests, etc.),

be sure to check out the awesome [article about forking](https://help.github.com/articles/fork-a-repo)

on the GitHub Help website &mdash; it will get you started quickly.

You should always write each batch of changes (feature, bugfix, etc.) in \*\*its own topic branch\*\*.

Please do not commit to the `master` branch, or your unrelated changes will go into the same pull request.

You should also follow the code style and whitespace conventions of the original codebase.

In particular, use tabs for indentation and spaces for alignment.

Before committing your changes, run `npm run lint` to catch any JS errors in the code and fix them.

If you add any new files to the Leaflet source, make sure to also add them to `build/deps.js`

so that the build system knows about them.

Also, please make sure that you have [line endings configured properly](https://help.github.com/articles/dealing-with-line-endings) in Git! Otherwise the diff will show that all lines of a file were changed even if you touched only one.

Happy coding!

### Using RollupJS

The source JavaScript code for Leaflet is a few dozen files, in the `src/` directory.

But normally, Leaflet is loaded in a web browser as just one JavaScript file.

In order to create this file, run `npm run rollup` or `yarn run rollup`.

You'll find `dist/leaflet-src.js` and `dist/leaflet.js`. The difference is that

`dist/leaflet-src.js` has sourcemaps and it's not uglified, so it's better for

development. `dist/leaflet.js` is uglified and thus is smaller, so it's better

for deployment.

When developing (or bugfixing) core Leaflet functionalities, it's common to use

the webpages in the `debug/` directory, and run the unit tests (`spec/index.html`)

in a graphical browser. This requires regenerating the bundled files quickly.

In order to do so, run `npm run watch` or `yarn run watch`. This will keep

on rebuilding the bundles whenever any source file changes.

## Running the Tests

To run the tests from the command line,

install [PhantomJS](http://phantomjs.org/) (and make sure it's in your `PATH`),

then run:

```

npm test

```

To run all the tests in actual browsers at the same time, you can do:

```

npm test -- -- --browsers Firefox,Chrome,Safari,IE

```

(Note: the doubling of "`--`" [special option](https://docs.npmjs.com/cli/run-script#description) is [important](https://github.com/Leaflet/Leaflet/pull/6166#issuecomment-390959903))

To run the tests in a browser manually, open `spec/index.html`.

## Improving Documentation

The code of the live Leaflet website that contains all documentation and examples is located in the `docs/` directory of the `master` branch

and is automatically generated from a set of HTML and Markdown files by [Jekyll](http://jekyllrb.com/).

The easiest way to make little improvements such as fixing typos without even leaving the browser

is by editing one of the files with the online GitHub editor:

browse the [`docs/ directory`](https://github.com/Leaflet/Leaflet/tree/master/docs),

choose a certain file for editing (e.g. `plugins.md` for the list of Leaflet plugins),

click the Edit button, make changes and follow instructions from there.

Once it gets merged, the changes will immediately appear on the website.

If you need to make edits in a local repository to see how it looks in the process, do the following:

1. [Install Ruby](http://www.ruby-lang.org/en/) if you don't have it yet.

2. Run `gem install jekyll`.

3. Enter the directory where you cloned the Leaflet repository

4. Make sure you are in the `master` branch by running `git checkout master`

5. Enter the documentation subdirectory by running `cd docs`

6. Run `bundle install`

7. Run `jekyll serve --watch` (if you have a Gem::LoadError error run `bundle exec jekyll serve --watch` instead)

8. Open `localhost:4000` in your web browser.

Now any file changes will be updated when you reload pages automatically.

After committing the changes, just send a pull request.

### API documentation

Since Leaflet 1.0.0-rc1, the API documentation in `reference-1.0.0.html` is handled

via [Leafdoc](https://github.com/Leaflet/Leafdoc). This means that next to the

code for every method, option or property there is a special code comment documenting

that feature. In order to edit the API documentation, just edit these comments in the

source code.

In order to generate the documentation, make sure that the development dependencies

are installed (run either `npm install` or `yarn install`), then just run

```

npm run docs

```

and you'll find a `.html` file in the `dist/` directory.

On every release of a new Leaflet version, this file will be generated and copied

over to `docs/reference.html` - there is no need to send pull requests with changes to this file to update the API documentation.

## Code of Conduct

Everyone is invited to participate in the Leaflet community and related projects:

we want to create a welcoming and friendly environment.

Harassment of participants or other unethical and unprofessional behavior will not be tolerated in our spaces.

The [Contributor Covenant](http://contributor-covenant.org/version/1/3/0/)

applies to all projects under the Leaflet organization.

Report any issues to agafonkin@gmail.com.

## Thank You

Not only does your contribution to Leaflet and its community earn our gratitude, but it also makes you AWESOME.

Join [this approved list of awesome people](https://github.com/Leaflet/Leaflet/graphs/contributors)

and help us push the limits of what's possible with online maps!