[//]: # (This is partially derived from https://github.com/bitcoin/bitcoin/blob/6579d80572d2d33aceabbd3db45a6a9f809aa5e3/CONTRIBUTING.md)

# Contributing to bitcoinjs-lib

Firstly in terms of structure, there is no particular concept of "bitcoinjs developers" in a sense of privileged people.

Open source revolves around a meritocracy where contributors who help gain trust from the community.

For practical purpose, there are repository "maintainers" who are responsible for merging pull requests.

We are always accepting of pull requests, but we do adhere to specific standards in regards to coding style, test driven development and commit messages.

## Communication Channels

GitHub is the preferred method of communication between members.

Otherwise, in order of preference:

\* bitcoinjs.slack.com

\* #bitcoinjs-dev on Freenode IRC

## Workflow

The codebase is maintained using the "contributor workflow" where everyone without exception contributes patch proposals using "pull requests".

This facilitates social contribution, easy testing and peer review.

To contribute a patch, the workflow is as follows:

1. Fork repository

1. Create topic branch

1. Commit patches

1. Push changes to your fork

1. Submit a pull request to https://github.com/bitcoinjs/bitcoinjs-lib

[Commits should be atomic](https://en.wikipedia.org/wiki/Atomic\_commit#Atomic\_commit\_convention) and diffs easy to read.

If your pull request is accepted for merging, you may be asked by a maintainer to squash and or [rebase](https://git-scm.com/docs/git-rebase) your commits before it is merged.

Please refrain from creating several pull requests for the same change.

Patchsets should be focused:

\* Adding a feature, or

\* Fixing a bug, or

\* Refactoring code.

If you combine these, the PR may be rejected or asked to be split up.

The length of time required for peer review is unpredictable and will vary from pull request to pull request.

Refer to the [Git manual](https://git-scm.com/doc) for any information about `git`.

## Regarding TypeScript

This library is written in TypeScript with tslint, prettier, and the tsc transpiler. These tools will help during testing to notice improper logic before committing and sending a pull request.

Some rules regarding TypeScript:

\* Modify the typescript source code in an IDE that will give you warnings for transpile/lint errors.

\* Once you are done with the modifications, run `npm run format` then `npm test`

\* Running the tests will transpile the ts files into js and d.ts files.

\* Use `git diff` or other tools to verify that the ts and js are changing the same parts.

\* Commit all changes to ts, js, and d.ts files.

\* Add tests where necessary.

\* Submit your pull request.

Using TypeScript is for preventing bugs while writing code, as well as automatically generating type definitions. However, the JS file diffs must be verified, and any unverified JS will not be published to npm.

## We adhere to Bitcoin-Core policy

Bitcoin script payment/script templates are based on community consensus, but typically adhere to bitcoin-core node policy by default.

- `bitcoinjs.script.decompile` is consensus bound only, it does not reject based on policy.

- `bitcoinjs.script.compile` will try to adhere to bitcoin-core `IsStandard` policies rules. (eg. minimalpush in https://github.com/bitcoinjs/bitcoinjs-lib/pull/638)

Any elliptic curve `sign` operations should adhere to `IsStandard` policies, like `LOW\_S`, but `verify` should not reject them [by default].

If you need non-standard rejecting `decoding`, you should use an external module, not this library.

#### TLDR

Where "standards compliant" refers to the default policies of bitcoin-core, we adhere to the following:

- Any "creation" event must create standards-compliant data (standards bound)

- Any "validation" event must allow for non-standards compliant data (consensus bound)

For stricter validation, use an external module which we [may have] provided.