Vue.js Contributing Guide

Hi! I'm really excited that you are interested in contributing to Vue.js. Before submitting your contribution, please make sure to take a moment and read through the following guidelines:

- Code of Conduct
- Issue Reporting Guidelines
- Pull Request Guidelines
- Development Setup
- Project Structure

Issue Reporting Guidelines

- Always use https://new-issue.vuejs.org/ to create new issues.

Pull Request Guidelines

- The `master` branch is just a snapshot of the latest stable release. All development should be done in dedicated branches. **Do not submit PRs against the `master` branch.**
- Checkout a topic branch from the relevant branch, e.g. `dev`, and merge back against that branch.
- Work in the `src` folder and **DO NOT** checkin `dist` in the commits.
- It's OK to have multiple small commits as you work on the PR - GitHub will automatically squash it before merging.
- Make sure `npm test` passes. (see development setup)
- If adding a new feature:
  - Add accompanying test case.
  - Provide a convincing reason to add this feature. Ideally, you should open a suggestion issue first and have it approved before working on it.
- If fixing bug:
  - If you are resolving a special issue, add `(fix #xxxx[,#xxxx])` (#xxxx is the issue id) in your PR title for a better release log, e.g. `update entities encoding/decoding (fix #3899)`.
  - Provide a detailed description of the bug in the PR. Live demo preferred.
  - Add appropriate test coverage if applicable.

Development Setup

You will need Node.js **version 8+**, Java Runtime Environment (for running Selenium server during e2e tests) and yarn.

After cloning the repo, run:

```
$ yarn # install the dependencies of the project
```

Committing Changes

Commit messages should follow the commit message convention so that changelogs can be automatically generated. Commit messages will be automatically validated upon commit. If you are not familiar with the commit message convention, you can use `npm run commit` instead of `git commit`, which provides an interactive CLI for generating proper commit messages.

Commonly used NPM scripts

```
# watch and auto re-build dist/vue.js
$ npm run dev

# watch and auto re-run unit tests in Chrome
$ npm run dev:test

# build all dist files, including npm packages
$ npm run build

# run the full test suite, including linting/type checking
$ npm test
```

There are some other scripts available in the `scripts` section of the `package.json` file.

The default test script will do the following: lint with ESLint -> type check with Flow -> unit tests with coverage -> e2e tests. **Please make sure to have this pass successfully before submitting a PR.** Although the same tests will be run against your PR on the CI server, it is better to have it working locally.

Project Structure

- `scripts`: contains build-related scripts and configuration files. Usually, you don't need to touch them. However, it would be helpful to familiarize yourself with the following files:
  - `scripts/alias.js`: module import aliases used across all source code and tests.

    - `scripts/config.js`: contains the build configurations for all files found in `dist/`. Check this file if you want to find out the entry source file for a dist file.

- **`dist`**: contains built files for distribution. Note this directory is only updated when a release happens; they do not reflect the latest changes in development branches.

  See dist/README.md for more details on dist files.

- **`flow`**: contains type declarations for Flow. These declarations are loaded **globally** and you will see them used in type annotations in normal source code.

- **`packages`**: contains `vue-server-renderer` and `vue-template-compiler`, which are distributed as separate NPM packages. They are automatically generated from the source code and always have the same version with the main `vue` package.

- **`test`**: contains all tests. The unit tests are written with Jasmine and run with Karma. The e2e tests are written for and run with Nightwatch.js.

- **`src`**: contains the source code. The codebase is written in ES2015 with Flow type annotations.

    - **`compiler`**: contains code for the template-to-render-function compiler.

      The compiler consists of a parser (converts template strings to element ASTs), an optimizer (detects static trees for vdom render optimization), and a code generator (generate render function code from element ASTs). Note that codegen directly generates code strings from the element AST - it's done this way for smaller code size because the compiler is shipped to the browser in the standalone build.

    - **`core`**: contains universal, platform-agnostic runtime code.

      The Vue 2.0 core is platform-agnostic. That is, the code inside `core` is able to be run in any JavaScript environment, be it the browser, Node.js, or an embedded JavaScript runtime in native applications.

        * **`observer`**: contains code related to the reactivity system.

        * **`vdom`**: contains code related to vdom element creation and patching.

        * **`instance`**: contains Vue instance constructor and prototype methods.

        * **`global-api`**: contains Vue global api.

        * **`components`**: contains universal abstract components.

    - **`server`**: contains code related to server-side rendering.

  - `platforms`: contains platform-specific code.

    Entry files for dist builds are located in their respective platform directory.

    Each platform module contains three parts: `compiler`, `runtime` and `server`, corresponding to the three directories above. Each part contains platform-specific modules/utilities which are imported and injected to the core counterparts in platform-specific entry files. For example, the code implementing the logic behind `v-bind:class` is in `platforms/web/runtime/modules/class.js` - which is imported in `entries/web-runtime.js` and used to create the browser-specific vdom patching function.

  - `sfc`: contains single-file component (`*.vue` files) parsing logic. This is used in the `vue-template-compiler` package.
  - `shared`: contains utilities shared across the entire codebase.
  - `types`: contains TypeScript type definitions
    - `test`: contains type definitions tests

Financial Contribution

As a pure community-driven project without major corporate backing, we also welcome financial contributions via GitHub Sponsors and OpenCollective. Please consult the Sponsor Page for more details.

Credits

Thank you to all the people who have already contributed to Vue.js!