

Blockchain.com Wallet

Be Your Own Bank at login.blockchain.com. Please contact support if you have any issues using the wallet.

About

This repo contains the three codebases/packages listed below.

Packages

- **blockchain-info-components** The shared UI components library.
- **blockchain-wallet-v4** The functional library for handling wallets.
- **blockchain-wallet-v4-frontend** The frontend application built with React/Redux.

Local Development

1. Ensure Node version ≥ 14.16 is installed. Using NVM is recommended.
2. From the project root, run the following command to install dependencies: `./setup.sh`.
3. Start the application in development mode: `yarn start`
4. The frontend application will now be accessible via browser at `localhost:8080`

If you require the application to run locally over HTTPS, follow the instructions here. You can disable SSL by setting the `DISABLE_SSL` env param to `true` with any start command. (e.g. `DISABLE_SSL=true yarn start:staging`)

Windows Support

To ensure proper support for Windows, please take the following actions before running the above setup instructions.

1. Open a Powershell window with rights elevated to an Administrator.
2. Run `npm install -g windows-build-tools`. This will install Python 2.7 and Visual C++ Build Tools which are required to compile some native Node modules.

-
3. Ensure Python has been added to your environment variables by opening a cmd prompt and typing `python`. If you get a `CommandNotFoundException` message, add the folder `%USERPROFILE%\windows-build-tools\python27` to your environment variables.

Tips & Useful Commands

1. To completely remove all dependencies and artifacts run `yarn clean`
2. To add/remove an NPM package run `yarn add` or `yarn remove` in the package folder. After installing or uninstalling a NPM package, run `yarn` in the root folder to re-init the project
3. All development specific dependencies should be installed as a `dev-dependency` in the top level `package.json` via `yarn i --save-dev [package-name]`
4. All application specific dependencies should be installed in the specific packages `package.json` via `yarn i --save [package-name]`

Running Environments Locally

The frontend application can be ran locally with different build configurations found in `config/env`. The following commands are available:

- `yarn start` Runs the application with the `development.js` configuration file
- `yarn start:dev` Runs the application with the `development.js` configuration file
- `yarn start:staging` Runs the application with the `staging.js` configuration file
- `yarn start:prod` Runs the application with the `production.js` configuration file
- `yarn run:prod` Runs the application mimicking the production environment entirely (i.e. code is bundled and minified, HMR is disabled, Express server is used (`./server.js`) and the `production.js` configuration file is loaded)

Notes:

- Developers will need to manually create the `development.js` and `staging.js` files
- Custom application runtimes are possible by modifying the corresponding environment files found in the `config/env` folder

Useful Chrome Extensions

- React Developer Tools Inspect the React component tree
- Redux DevTools View/debug Redux state changes

Release Process

Prerequisites To be able to create a release follow these steps starting with “Obtain a personal access token...”: <https://github.com/release-it/release-it#github-releases>

`GITHUB_TOKEN` should be saved as `RELEASE_IT_TOKEN` instead in your `bash_profile` or wherever you keep env variables

You’ll need git changelog to generate the history since the last release:

Release Steps

1. From the tip of the `development` branch, run `yarn release`
2. Answer the questions prompted via CLI, accepting the defaults for each
3. Once completed, this will create a new tag which will trigger a builds
4. Once builds have finished, deploy the images to desired environments
5. Test and verify the latest changes in desired environments
6. Create PR to merge the HEAD of `development` into `master`
7. Merge PR to `master` so that `master` always reflects what is currently in production

Code Quality

- `yarn vet` Runs Prettier, lint JS, lint CSS and finally all unit tests

Linting

We follow the rules outlined by the Javascript Standard Style as well as a few React specific rules.

Code linting is handled by ESLint. The following commands are available:

- `yarn lint` Lints all packages
- `yarn lint:components` Lints only blockchain-info-components
- `yarn lint:core` Lints only blockchain-wallet-v4
- `yarn lint:frontend` Lints only blockchain-wallet-v4-frontend
- `yarn lint:fix` Automatically resolves fixable issues via ESLint

These IDE plugins/packages assist with complying with these lint rules while developing:

- Atom
- VS Code
- WebStorm

Prettier

We follow all standard rules that are provided by Prettier. The following commands are available:

- `yarn prettier` Runs Prettier against all packages
- `yarn prettier:components` Runs Prettier against only blockchain-info-components
- `yarn prettier:core` Runs Prettier against only blockchain-wallet-v4
- `yarn prettier:frontend` Runs Prettier against only blockchain-wallet-v4-frontend

It is recommended to setup a Prettier plugin for your IDE plugins/packages that will automatically prettify your files on save.

- Atom
- VS Code
- WebStorm

When installing the plugin for VS Code make sure you are on v3.7.0 or lower

Unit Tests

Testing is done via Jest and Enzyme.

Running Tests

- `yarn test` Runs unit tests for all packages
- `yarn test:components` Runs unit tests for only blockchain-info-components
- `yarn test:core` Runs unit tests for only blockchain-wallet-v4
- `yarn test:frontend` Runs unit tests for only blockchain-wallet-v4-frontend

Note: if you see errors that Jest cannot resolve package imports, you may need to run `yarn test` before testing specific packages (eg, `yarn test:frontend`)

Running Tests via Watch

- `yarn test:watch` Watches and then runs desired tests
- `yarn test:components:watch` Watches and then runs desired tests for only blockchain-info-components
- `yarn test:core:watch` Watches and then runs desired tests for only blockchain-wallet-v4
- `yarn test:frontend:watch` Watches and then runs desired tests for only blockchain-wallet-v4-frontend

Debugging Tests To enable debugging for unit tests via the Chrome browser, run the following commands:

- `yarn test:components:debug` Debugs unit tests for only blockchain-info-components
- `yarn test:core:debug` Debugs unit tests for only blockchain-wallet-v4
- `yarn test:frontend:debug` Debugs unit tests for only blockchain-wallet-v4-frontend

After running one of the above commands, Node will wait for a debugger to attach before starting the tests. To attach, simply open your browser and go to `chrome://inspect` and click on “Open Dedicated DevTools for Node”, which will give you a list of available node instances you can connect to. Click on the address displayed in the terminal (usually localhost:9229) and you will be able to debug tests using Chrome’s DevTools.

Updating Snapshot Tests We are snapshot testing UI some components. Here are the commands to update them when necessary:

- `yarn test:components:update` Updates component snapshots for only blockchain-info-components
- `yarn test:frontend:update` Updates component snapshots for only blockchain-wallet-v4-frontend

Code Coverage

To generate code coverage reports via Istanbul, the following commands are available:

- `yarn coverage` Generates a coverage report for all packages
- `yarn coverage:components` Generates coverage report for only blockchain-info-components
- `yarn coverage:core` Generates coverage report for only blockchain-wallet-v4
- `yarn coverage:frontend` Generates coverage report for only blockchain-wallet-v4-frontend

Depending upon which coverage report was ran, the results can be found in the following directories:

- `coverage/index.html`
- `coverage/blockchain-info-components/index.html`
- `coverage/blockchain-wallet-v4/index.html`
- `coverage/blockchain-wallet-v4-frontend/index.html` Simply open the `index.html` file in your browser to view.

TypeScript

TypeScript is supported and should be used when adding new code. It's also recommended to replace legacy JS with TS when time allows.

TS Coverage

We are using Codechecks and Typecov for coverage reporting. Coverage is automatically analyzed for PRs and the following command is available.

- `yarn codechecks`

Code Bundle Analysis/Reports

To visualize and interact with the tree of the production code bundles files:

- `yarn analyze` Once completed, a browser will automatically open with the results.

Storybook

Storybook is used by the blockchain-info-components and blockchain-wallet-v4-frontend packages to interactively view, develop and test components. The following commands are available:

- `storybook:build-wallet`: Builds the static storybook assets for wallet specific components (if base components is running locally, storybook will put wallet and base components into the same storybook UI)
- `storybook:build-base`: Builds the static storybook assets for base shared components
- `storybook:serve-wallet` Builds storybook assets and then serves them locally at `localhost:6006`
- `storybook:serve-base` Builds storybook assets and then serves them locally at `localhost:6007`
- `storybook:deploy-wallet` Builds storybook assets and then serves them to github pages. **You will probably need to run `cd ./packages/blockchain-info-components && git remote add origin git@github.com:blockchain/blockchain-wallet-v4-frontend.git` first.**
- `storybook:deploy-base` Builds storybook assets and then serves them to github pages. **You will probably need to run `cd ./packages/blockchain-info-components && git remote add origin git@github.com:blockchain/blockchain-wallet-v4-frontend.git` first.**

If the deploy begins to fail, deleting the static build file before redeploy will likely help.

Contribute

Please review to the Wiki

Security

Security issues can be reported to us in the following venues:

- Email: security@blockchain.info
- Bug Bounty: <https://hackerone.com/blockchain>