
Rabby Wallet

Rabby Wallet is an open source browser plugin for the DeFi ecosystem, providing users with a better-to-use and more secure multi-chain experience.

Install

You can Download the latest Rabby here.

Guideline for integrating Rabby Wallet

To help dapp developers support and integrate Rabby Wallet more easily, we recommend you use our integration solution that has almost NO development cost and does not introduce any uncertainty:

Problem

When a dapp connects to an extension wallet, it usually works in this way:

1. The extension wallet will integrate an “Ethereum” object into the dapp page while it’s loading;
2. The dapp will look for this “Ethereum” object to determine if an extension wallet is installed;
3. If the “Ethereum” object is detected, all following interactions between the dapp and the extension wallet are realized by this “Ethereum” object.
4. If the “Ethereum” object is not detected, the dapp will ask users to download a new extension wallet.

The problem is that many dapps will wrongly display this detected “Ethereum” object as “MetaMask” and displays a “connect to MetaMask” button by default which brings a lot of confusion to the users as any Web3 wallet can inject this “Ethereum” object.

Solution:

We recommend you to solve above problem with simple modifications as follows:

1. On your connection page, display both connection buttons for “MetaMask” & “Rabby Wallet” when the “Ethereum” object is detected: these two buttons basically have the same function. Users can click any of them to interact with the “Ethereum” object and perform the connection operation. These two buttons are only used to display both brands’ logos to help users understand their operation path.

-
2. If the “Ethereum” object is not detected, then suggest the users go download the extension wallet and provide download links for both “MetaMask” and “Rabby Wallet”.

This solution does not involve any change to your actual business logic and is just simple UI adjustments. It does not introduce any uncertainty and is with rather low cost.

You can refer to “<https://debank.com>” for final display effect.

Potential issues:

According to the above solution, if a user is using the “Rabby Wallet” and clicks the “connect to MetaMask” button, he will still interact with the “Rabby Wallet” and vice versa which might be a little bit weird.

However, above issue is a very rare scenario and very unlikely to happen because users are not likely to click and interact with an extension wallet that he hasn’t installed. Even it happens, it’s not a real problem from the user’s perspective.

Please don’t hesitate to reach us if you have any doubt.

Contribution

Install dependency

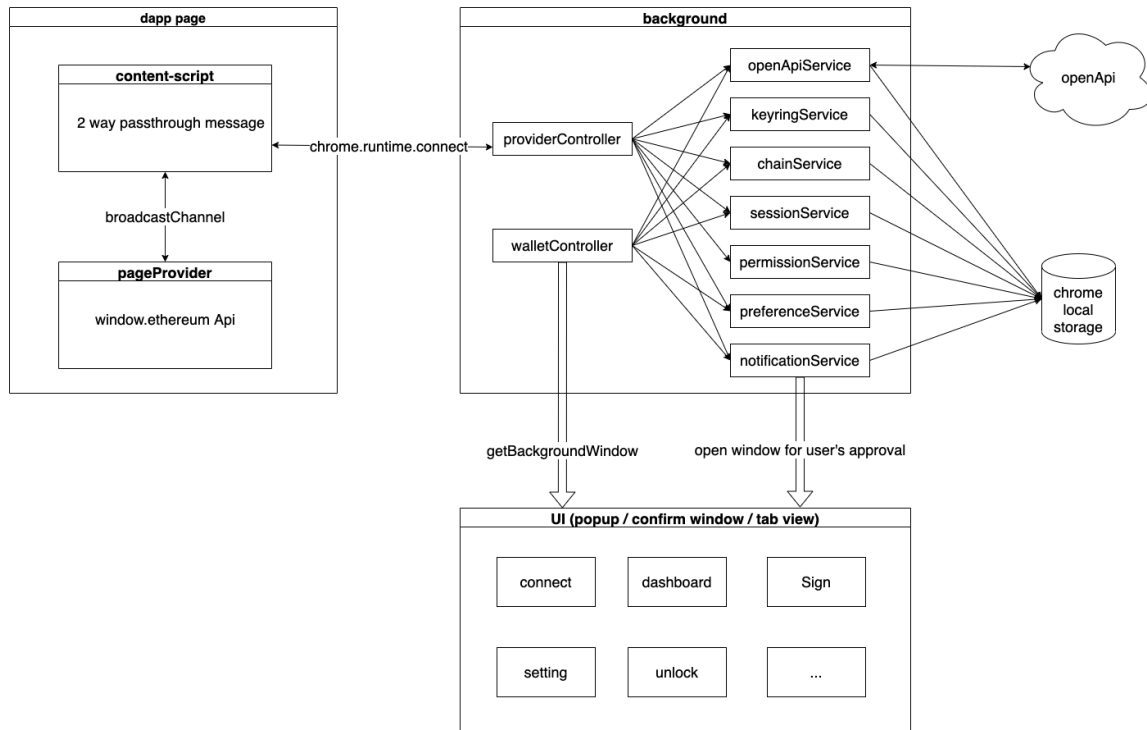
1. Install Node.js version 14 or later
2. Install Yarn `npm install -g yarn`
3. Run `yarn` to install dependency

Development

Run `yarn build:dev` to develop with file watching and development log(you can see request sent by dapp in website console in this mode and notification will not close when focus lost)

Run `yarn build:pro` to build a production package, it’s in dist folder

Architecture



Extension's scripts

below 4 scripts all live in different context!

- background.js

for all async request and encrypt things.

user's keyrings, password and wallet personal preference data all stored in chrome local storage.

it has 2 main controllers:

1. **walletController**

it expose methods to background window, so other scripts can access these methods with `runtime.getBackgroundPage`, e.g. `ui.js`.

2. **providerController**

it handles request from pages(dapp request).

- content-script

injected at `document_start`, share the same dom with dapp, use `broadcastChannel` to tap `pageProvider`.

the main purpose is inject `pageProvider.js` and pass messages between `pageProvider.js` and `background.js`.

- pageProvider.js

this script is injected into dapp's context through `content-script`. it mounts `ethereum` to `window`.

when dapp use `window.ethereum` to request, it will send message to `content-script` with `broadcastChannel` and wait for it's response.

then the `content-script` will send message to `background` with `runtime.connect`.

after `background` receive the message, it will use `providerController` to handle the request. and keep the message channel in `sessionService` for later communicate.

- ui

it's used by 3 pages which share the same js code, but the template html is different for respective purpose.

1. `notification.html`

triggered by dapp to request user's permission.

2. `index.html`

opened in browser tab for better user interaction experience.

3. `popup.html`

user click the extension icon on the right of address bar, the popup will show.

Thanks

Thanks for contributions from MetaMask team to browser extension wallet community, Rabby uses (or forks) them to make Rabby better.

Other Docs

- How to add a new translation to Rabby