Remix Plugin Directory Documentation

Remix Team

Dec 03, 2020

Contents:

Using the Remix Plugin system is an easy way to extend the functionality of the Remix IDE.

CHAPTER 1

Getting started

1.1 Basic concepts of a simple Remix plugin

A remix plugin in its simplest form is a webapp running in an iFrame on the Remix IDE.

To interact with Remix you will need to use the Remix plugin code. This will provide:

- making calls to Remix, for example to load a file
- listen to events from Remix, for example when a user has compiled a sol file
- loading the currently active CSS theme in your plugin
- interacting with other plugins
- helper functions like loading content

You can use any framework you like, React, Webpack, Angular . . .

1.2 Loading the library

The library you have to use for an iFrame plugin is called plugin-webview.

Install the library with your package manager as described in plugin-webview.

Using the library you create an instance of the plugin *client*, you can have your own class extend the plugin or just instanciate a plugin client.

The client can listen to events and call methods.

1.3 The onload event

The onload event is the signal in your app communication is setup with Remix. In case you don't see that event happening something is wrong. You should check the console to find out.

```
client.onload(async () => {
})
```

1.4 Bootstrap CSS

When the client is loaded Remix will provide your iFrame with a theme, the current theme choosen by the user.

If that onload event doesn't happen, the theme won't be loaded and neither will the client be ready to interact with Remix.

You should not load your own bootstrap css! It is not necessary. When using bootstrap JS functions however, you should load them in your codebase.

Your styling should always directed towards compliance with the look & feel of Remix. Use the standard buttons, like primary, secondary, success, danger and so on. They will be styled by the Remix CSS that is loaded on your iFrame. You should follow the general guidelines outlined in this manual.

1.5 The plugin API

1.5.1 Files

By simply calling the fileManager method on the client you can interact with the Remix filesystem.

Read about those file methods and events here https://github.com/ethereum/remix-plugin/blob/master/packages/api/doc/file-system.md.

1.5.2 Importing content

You can have Remix handle importing content for you. It can load data from various sources and also write it to file: https://github.com/ethereum/remix-plugin/blob/master/packages/api/doc/content-import.md

1.5.3 The solidity compiler

You can interact with the solidity compiler. Follow these methods https://github.com/ethereum/remix-plugin/blob/master/packages/api/doc/solidity.md

1.5.4 Exposing methods

You can expose methods in your app to other plugins or call methods on other plugins. It is described in plugin-webview.

You should include those methods in your profile.json which you create to publish your plugin.

1.5.5 Calling your plugin & methods directly from a URL

Read the documentation here.

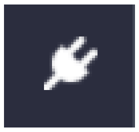
1.5.6 More API

Check out these docs to find out what more you can do with the plugin. These include networking, unit testing and much more.

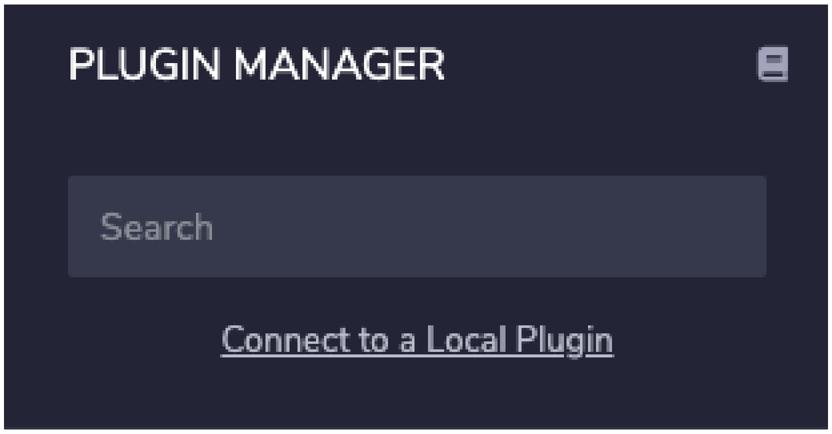
1.6 Installing the plugin locally

When you've developed your app you can run locally and install it in Remix.

1. Click on the plugin icon

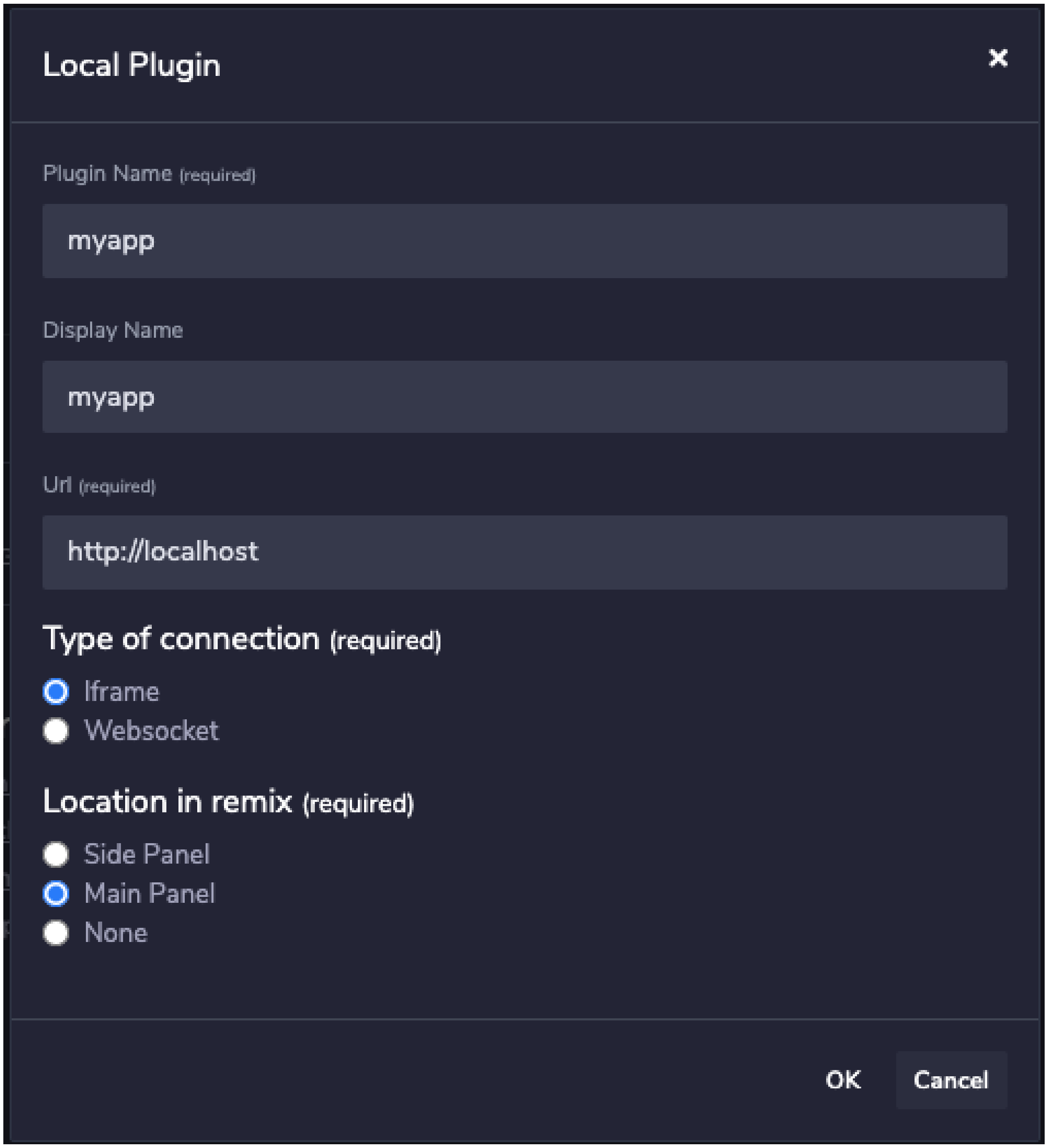


1. Click on add local plugin



You can load your plugin either in the main panel, or on the side.

1. Specify your plugin URL



Local Plugin

Plugin Name (required)

myapp

Display Name

myapp

Url (required)

http://localhost

Type of connection (required)

[✓ Iframe] Websocket

Location in remix (required)

Side Panel [✓ Main Panel] None

CHAPTER 2

Guidelines for documentation and styling

2.1 UI guidelines

A great user experience needs a good interface. Here are some requirements for a successfull plugin.

Boostrap CSS

As mentioned in the getting started part of these docs, Remix loads a Bootstrap CSS theme for you in your plugin. You want to try **not to write CSS yourself** and just **use the bootstrap components and layout**.

- Use the bootstrap sizing, padding, margin and styling classes.
- Use buttons for buttons and form-control elements for forms and so on.
- Look at the Remix interface to get a feel for the components used.

Clear navigation

Really guide your user through all the steps. Provide dummy or mockup data if possible to generate a useful result. There should not be an empty state. Use tooltips or links to the documentation.

Feedback

Every step should have feedback concerning errors, successes or requirements. If a step requires the user to do something in another part of the Remix environment this should be clearly indicated. If files are not loading correctly or the Remix IDE returns errors on calls this should be made clear. Your user should never be stuck. Point your user to the results generated, tell them what to do if the results fail.

Dependencies

If there are dependencies on other services or software clearly tell the user where to look or what to do. If certain service providers or functionalities in Remix have not been actived your plugin should test for them and provide feedback.

Tooltips and documentation

Always refer to any documentation you have on all input fields and steps. Use tooltips whenever possible. Just a form with some input fields is not a good user experience. Don't assume your user knows what to do.

2.2 Documentation guidelines

Great documentation is crucial to the success of your plugin. We encourage you to publish your docs on Read the docs.

We have some great tips on how to make your documentation effective.

Clearly state the purpose of your plugin

Describe in detail what the plugin does. Do not assume any knowledge on the part of your user, introduce all the concepts and technologies. Use external links to anything related to your plugin.

Who is your plugin for?

Define who your user is, what they might need and what they might get out of using your plugin.

Requirements

What is needed to successfully run the plugin? Some plugins might need other software or services to run. Provide information, links and step by step guides to setting those up.

In some cases certain functionalities or settings in the Remix environment are needed. Clearly state the knowledge and steps needed. If possible provide links to sections in the Remix documentation that explain what the user needs to know.

Installation and configuration

Tell your user where to find your plugin the Remix directory and how to set it up. Your plugin will have an icon in the sidebar of Remix, show your user what it is.

Set by step demonstration

The best way to explain your plugin is by demonstration. Provide a detailed step by step tutorial on how to use it. Dummy data that leads to results is a great way to help your user. Try to avoid an empty state.

Use screenshots

A picture is worth a thousand words.

Be available for troubleshooting

Always assume something can go wrong. Provide a place where users can go with their issues. This can be done on github, gitter, any chat channel or e-mail.

Introduce yourself

Link to any websites, companies, repositories you are involved with. This can greatly help the user to engage with your plugin and maybe encourage them to assist in development and feedback.

Always be up to date

Be accurate in every detail, update the documentation when your plugin has an update.

CHAPTER 3

Publishing your plugin

The repo ethereum/plugins-directory holds a directory called plugins. Each plugin has its own directory.

3.1 Profile.json

To publish a plugin you need to create a pull request there that publishes a JSON profile.json file describing your plugin.

Mandatory fields are:

- name
- displayName
- version
- URL
- icon
- location
- documentation

```
{
  "name": "<name of the plugin>",
  "displayName": "<display name of the plugin>",
  "methods": [], // list of methods that the plugin is making available to other␣
↪plugins
  "version": "1.0.0-alpha", // semver version
  "url": "<URL>", // can be an https or ipfs URL - ipfs://<ipfs_hash>
  "description": "<description>",
  "icon": "<link to image>", // https link to image or BASE64 value
  "location": "<panel where the plugin should be rendered>", // can be sidePanel,␣
↪mainPanel or hidden
```

(continues on next page)

(continued from previous page)

```
  "documentation":"<URL>"
}
```

When the plugin is approved it will be published to the plugin directory in Remix.

3.2 Profile tools & Hosting

You can host your plugin anywhere you want. We provide tools to upload your plugin to our own IPFS gateway.

This tool will guide you through the process. You can have it create the profile file directly into the plugins directory so you can push your PR to us.

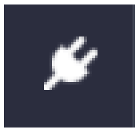
```
git clone https://github.com/ethereum/remix-plugins-directory
cd remix-plugins-directory/tools
npm i
node ipfs-upload/bin/upload-remix-plugin
```

You will get an IPFS link to your app you can copy/paste to Remix to test it.

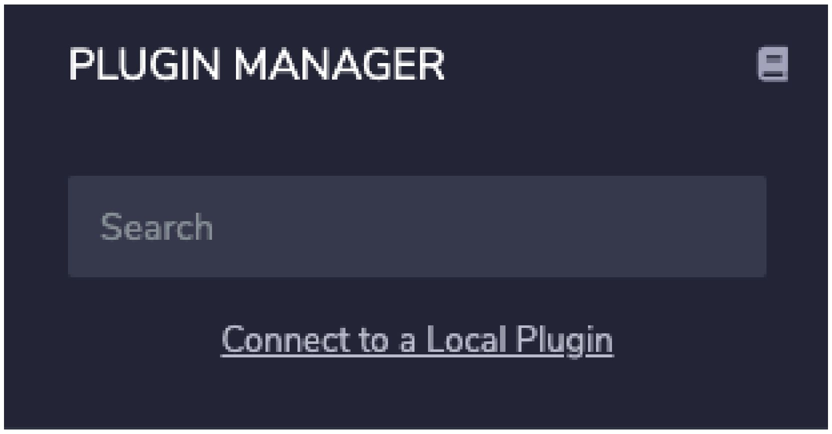
3.3 Testing your app in IPFS

Before you push your PR you should always test your build. Running locally or on IPFS can be different, depending how you setup your app.

1. Click on the plugin icon

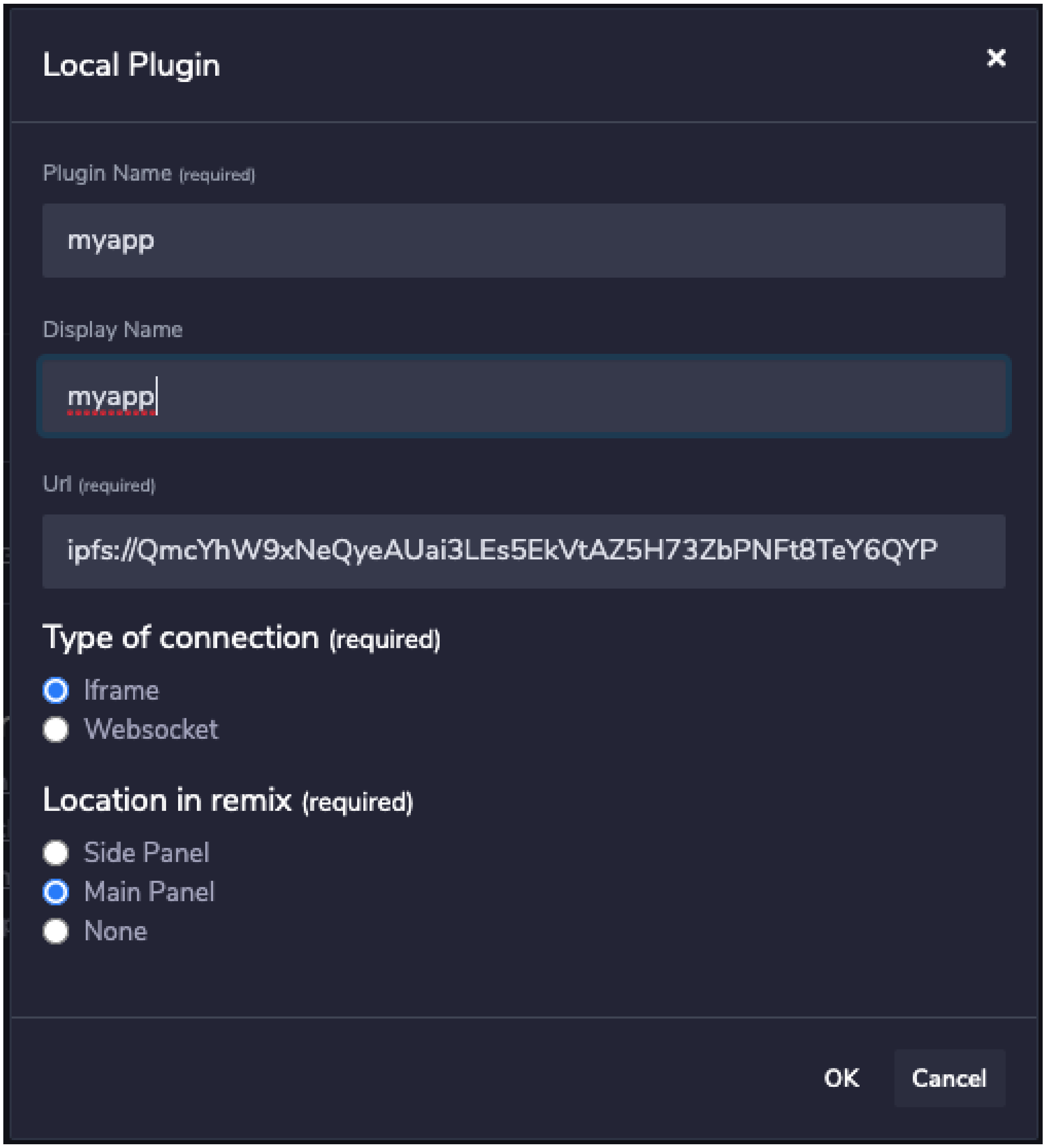


1. Click on add local plugin



You can load your plugin either in the main panel, or on the side.

1. Specify your plugin URL, for example ipfs://xxxxx


Local Plugin

Plugin Name (required)

myapp

Display Name

myapp

Url (required)

ipfs://QmcYhW9xNeQyeAUai3LEs5EkVtAZ5H73ZbPNFt8TeY6QYP

Type of connection (required)

[✓ Iframe]
Websocket

Location in remix (required)

Side Panel
[✓ Main Panel]
None

CHAPTER 4

Getting help

Feel free to join our gitter chat if you have any question

- Remix Dev - Remix IDE development
- Remix Plugin Dev - Plugins development