Guide contents

Find the current script environment

Register one plugin

Choose local or remote deliberately

Sources / References

Documentation: In progress **Quick start: In progress**

Offline copy from ModDocs. Source links remain clickable.

OpenRCT2 plugin development

OpenRCT2 runs **JavaScript plugins** to add behavior, UI and multiplayer functionality on top of parks and scenarios. This is different from creating ride/park assets. The project's own scripting guide describes the plugin folder, `registerPlugin`, API types and multiplayer restrictions.

> **Quick start:** create one `.js` file in your OpenRCT2 user directory's `plugin` folder. Register a plugin that logs a message when a park opens, launch a test park, and confirm the log. Only then add UI or game-state changes.

Find the current script environment

On the title screen, use the red toolbox menu's **Open custom content folder**, then find the `plugin` directory. The official guide lists typical OS paths and notes that every `.js` below this directory is loaded recursively. Moving or renaming a `.js` file out of that directory disables it. Use this fact to test both the installed and removed states.

Current OpenRCT2 uses QuickJS-NG with modern JavaScript support, but older versions used Duktape. The official guide also notes that `import/export` module syntax is not loaded directly; bundle multi-file projects if needed. Inspect your installed OpenRCT2 version and the matching guide revision before copying a new syntax feature or API version number.

Register one plugin

The official starter uses a `main` function passed to `registerPlugin`. The shape is:

```javascript
function main() {
    console.log("ModDocs test plugin started");
}

registerPlugin({
    name: "ModDocs Test",
    version: "1.0",
    authors: ["Your Name"],
    type: "local",
    licence: "MIT",
    main: main
});
```

The exact supported `targetApiVersion`/`minApiVersion` fields and values should be chosen from the current guide for the release you target. Put this in a uniquely named `.js`, open a park and watch the console. On Windows, the guide says `openrct2.com` provides an interactive console, whereas `openrct2.exe` may not. If nothing logs, check file extension, folder and current API requirements before adding more code. Leave the park and reopen it after editing unless hot reload is enabled.

Choose local or remote deliberately

The scripting guide explains three plugin types: `local`, `remote` and `intransient`. In single-player their restrictions are broad; in multiplayer they differ sharply. A local plugin runs on a client and should change park state through supported game actions. A remote plugin installed on a server can be sent to clients. Direct, non-deterministic park mutations can desynchronize players. Do not test multiplayer compatibility by only starting a solo park.

For a UI feature, guard access to `ui` because headless servers do not expose it. For a park-state feature, use the appropriate built-in or custom game action and validate both query and execute behavior. The repository's type definitions provide signatures; the guide's sample scripts show working structures. Keep state changes deterministic across players.

Package the `.js` and any original resources, then test in a clean user directory. Document the minimum OpenRCT2/API version and whether the plugin is local, remote or intransient. This guide is **unfinished** beyond the first loading test: it does not yet include a fully tested custom window, park action or multiplayer scenario.

Sources / References

External project and documentation links cited in this guide. Check the upstream release and version before following version-sensitive steps.

- scripting guide
- type definitions