

Figma Plugin Project Structure

TypeScript-based Figma plugin with custom UI, design manipulation, and proper sandbox architecture.

Updated 2025-12-22

#figma #plugin #design #typescript #ui

PNG

PDF

Copy

Prompt

Project Directory

my-plugin/

- manifest.json Plugin configur...
- src/ Source code
 - code.ts Main plugin cod...
 - ui.html Plugin UI entry
- ui/ UI components
 - App.tsx Main UI compone...
 - styles.css
 - components/
 - Button.tsx
 - Input.tsx
- utils/
 - helpers.ts
 - messages.ts UI ↔ code messa...
- types/
 - index.ts
- dist/ Build output
 - code.js
 - ui.html
- package.json
- tsconfig.json
- esbuild.config.mjs Build configura...
- figma.d.ts Figma API types
- .gitignore
- README.md

Why This Structure?

Figma plugins have two contexts: the main code runs in a sandbox with access to Figma's API, while the UI runs in an iframe. They communicate via `postMessage`. This structure separates these concerns with `code.ts` and `ui/` folders.

Key Directories

src/code.ts - Sandbox code with Figma API access

src/ui/ - React/HTML UI running in iframe

src/utils/messages.ts - Type-safe messaging between contexts

dist/ - Bundled output for Figma to load

Main Plugin Code

```
// src/code.ts
figma.showUI(__html__, { width: 300, height: 400 });

figma.ui.onmessage = (msg) => {
  if (msg.type === 'create-rectangle') {
    const rect = figma.createRectangle();
    rect.resize(msg.width, msg.height);
    figma.currentPage.appendChild(rect);
  }
  figma.closePlugin();
};
```

Getting Started

- Create plugin from Figma menu: Plugins → Development → New Plugin
- Clone or initialize your project structure
- `npm install` to install dependencies
- `npm run build` to compile TypeScript
- In Figma: Plugins → Development → Import plugin from manifest

Plugin Architecture

Sandbox (code.ts) - Full Figma API, no DOM access, limited JS APIs

UI (iframe) - Full DOM/browser APIs, no Figma API access

postMessage - Communication bridge between contexts

figma.ui.postMessage - Send data from sandbox to UI

Best Practices

- Keep sandbox code minimal—do heavy work in UI iframe
- Use typed messages for UI ↔ sandbox communication
- Batch node operations to avoid performance issues
- Call `figma.closePlugin()` when done
- Use `@figma/plugin-typings` for TypeScript support

Common APIs

figma.currentPage - Access current page and selection

figma.createX() - Create shapes, frames, text, etc.

node.exportAsync() - Export nodes as images

figma.loadFontAsync() - Required before editing text

When To Use This

- Automating repetitive design tasks
- Generating designs from data
- Exporting assets in custom formats
- Design system management
- Integrating with external services

Trade-offs

Sandbox Limits - No fetch, setTimeout capped, limited JS APIs

Two contexts - Messaging adds complexity vs single-context plugins

Build step - TypeScript requires bundling before running