2 種打包方式的差異 (final 分支跟 dq2apilib 分支)

下面幫你把「**Jimmy 法**」跟「我們剛剛封裝法」做成對照表，再給幾個一鍵修正建議。看完就知道兩種做法各自的取捨。

差異一覽（重點）

面向	我們剛剛的封裝	Jimmy 法（你圖中的做法）	影響/說明
入口輸出	**ESM + CJS 兩份：** `dist/index.js`、`dist/index.cjs`	**只有 ESM：** `dist/index.js`	我們的包對舊的 CommonJS（`require()`）也安全；Jimmy 法目前 `exports.require` 指到 ESM，純 CJS 環境會報錯。
`exports` 子路徑	明確列出 `./proto/FeedAPI`、`./proto/HDAPI` …	**萬用萬用字元：** `"./proto/*"`、`"./lib/*"`	Jimmy 法更精簡、好維護；我們的較嚴謹、IDE 自動補全較清晰。
lib(UMD) 處理	直接 **打進 bundle**（index.js 中）	**同時**：index 會 import `src/lib/dq2apilib.js`，且用 Copy 把整個 `src/lib` 複製到 `dist/lib/`	兩者都能用；Jimmy 法多了 `dist/lib/*` 可供深路徑引入。
型別	我們用 **shims** 或 `@ts-expect-error`，最小侵入	Jimmy 法在 `src/lib/dq2apilib.d.ts` **提供型別**，`index.ts` 直接 `import dq2wasmapilib, { type IDQ2WasmLib }`	Jimmy 型別體驗較好；但要維護一份 d.ts。
Rspack 插件	`CopyRspackPlugin`（**內建**，`@rspack/core`）	同樣（`@rspack/core`）	你的修正正確：新版已內建。
Rspack 設定	兩份 config（ESM / CJS），`target: 'web'`，`performance.hints=false`，細粒度 patterns	**單份 config**，`output.module + experiments.outputModule=true`，patterns 直接整夾 copy	Jimmy 版更短；我們的雙輸出更相容。
tsconfig	產 `dist/index.d.ts`（`emitDeclarationOnly`），`moduleResolution: 'Bundler'`，嚴格度適中	更嚴（`noImplicitAny` …），且 `include` 同時包含 `*.ts`、`*.js`	都 OK；重點是能產出 `dist/index.d.ts`。
警告處理	避免 `.default` 讀取 → 無 `ESModulesLinkingWarning`	直接 `default` import（因為有 d.ts），不會噴 LinkingWarning	兩種都能乾淨通過。
相容性風險	低（ESM/CJS 皆可）	**CJS 消費者有風險**（`exports.require` 指到 ESM）	若後端或舊腳本用 `require('@tvc/dq2apilib')` 會炸。

什麼情境下選哪個？

- **要最廣相容（前端/Node/CJS/老工具）** → 用我們的做法或在 Jimmy 法上**加 CJS**（上面的小修正）。
- **要最少檔案、最短配置、只跑 ESM 生態** → Jimmy 法保持不變（但確定所有消費者都是 ESM/Bundler）。
- **需要深路徑引入**（`@tvc/dq2apilib/lib/*`、`/proto/*`） → Jimmy 法的萬用字元 `exports` 很好維護；我們也可改用相同寫法。

小 checklist（避免踩雷）

- ☐ `exports.require` **不要** 指向 ESM（否則 `require()` 會掛）
- ☐ `moduleResolution: 'bundler'` 保持，VSCode 才會正確解 `exports`
- ☐ `dist/index.d.ts` 要產出（`emitDeclarationOnly: true`）
- ☐ Rspack 複製：`lib/` 與 `proto/` 都要到 `dist/`（你已做）
- ☐ 若要完全靜默，`performance.hints=false`（你 ESM 單包通常會超 244 KiB）

一句話總結

- **Jimmy 法**：簡潔、好維護、型別佳，但**預設只有 ESM**；補一個 `index.cjs` 並把 `exports.require` 指過去，就跟我們的相容性一致了。
- **我們的法**：相容性最強、零驚喜；若想更容易維護，可把 `exports` 也改成 Jimmy 的萬用字元寫法。