內容

[ electron主程式目錄結構建議 1](#_Toc192598777)

[ /render目錄 1](#_Toc192598778)

[ ipc範例說明(以logViewer為例) 2](#_Toc192598779)

[ ipc範例說明(以tabUI為例) 3](#_Toc192598780)

[ ipc模式說明 5](#_Toc192598781)

[ electron主程式建議儘量採用依賴延遲載入 5](#_Toc192598782)

# electron主程式目錄結構建議

electron主程式目錄結構建議如下

|  |  |
| --- | --- |
| 項目 | 說明 |
| /builder | electron主程式打包程序 |
| /icon | electron主程式icon檔案 |
| /render | 前端程式擺放位置 |
| /render/preload | 前端程式preload程序擺放位置 |
| /src | 主程式原始碼 |
| /src/ipc | 主程式-electron事件處理 |
| /src/modal | 主程式-互動視窗程序 |
| /src/util | 主程式-工具程式 |
| /src/main.js | 主程式-進入點 |
| /src/menu.js | 主程式-選單 |

以下提供細節說明

# /render目錄

前端程式擺放位置

前端程式負責electron應用的畫面呈現與互動部分

electron會以BrowserWindow或WebContentsView的loadFile方法載入此目錄下程式

前端程式為可在chrome瀏覽器執行之html,css,js程式

簡單的畫面程式可直接使用不需編譯的html+css+js放入此目錄

複雜的畫面程式可以使用reactjs專案方式,將reactjs編譯出之html+css+js放入此目錄

除preload子目錄外,此目錄下每個子目錄即代表一個畫面單位

preload目錄下擺放每個畫面包的preload script

preload script為BrowserWindow或WebContentsView載入畫面程式前會執行的javascript程式,也是於chrome瀏覽器端執行,但可以存取electron物件,初始化electron ipc,為electron主程式與前端程式之間的橋樑

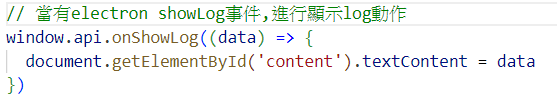
# ipc範例說明(以logViewer為例)

於\render\preload\logViewer.js設定監聽showLog事件供前端程式使用



前端程式透過preload中提供的api監聽showLog事件,與決定事件發生後要執行的動作

\render\logViewer\index.js



\src\modal\logViewer.js

Electron主程式透過BrowserWindow載入前端程式,載入完成後透過ipc發送showLog事件給前端程式驅動前端js動作



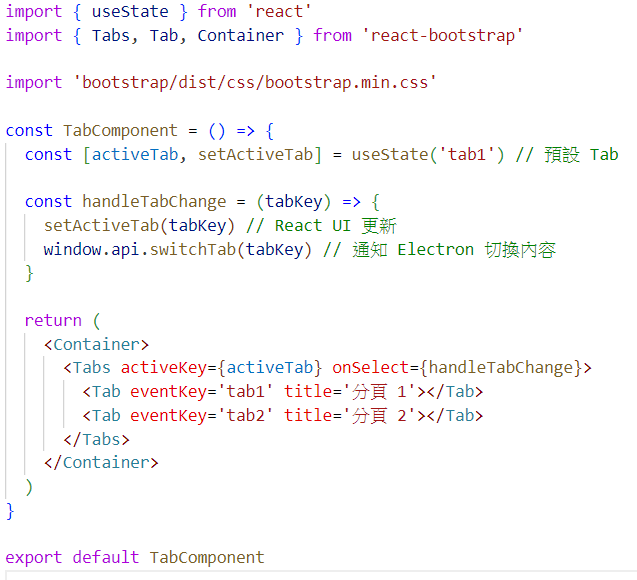
# ipc範例說明(以tabUI為例)

於\render\preload\tabUI.js設定驅動switch-tab事件供前端程式使用



前端程式透過preload中提供的api驅動switch-tab事件,並傳遞事件參數給主程式

tabUI\src\Component\Main.js



主程式使用electron ipc功能監聽switch-tab事件與決定事件發生後要執行的動作

\src\ipc\tabUI.js



# ipc模式說明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 方法 | 發起端 | 非同步/同步 | 使用場景 | 適用情境 |
| ipcRenderer.send() ->ipcMain.on() | 前端 | 非同步 | 前端主動發訊息給主進程,不取得回應 | 開啟新視窗、發送通知 |
| ipcRenderer.invoke() ->ipcMain.handle() | 前端 | 非同步 | 前端主動發訊息給主進程並取得回應 | 讀取資料、查詢設定 |
| webContents.send() ->ipcRenderer.on() | 主進程 | 非同步 | 主進程主動發送訊息給前端 | 前端初始化、前端狀態更新 |

# electron主程式建議儘量採用依賴延遲載入

electron主程式建議儘量採用依賴延遲載入以加快應用程式的啟動時間,尤其遇到較大的模組時(如資料庫,圖片處理函式庫等)

如下範例, realm的require並非放在程式最上方,而是放到下方實際使用前



Electron與node js原生模組的依賴,可以選擇不延遲載入,這樣程式比較簡潔,可讀性高

或者該依賴在程式碼中使用超過二次以上的也可以選擇不延遲載入

