Js desktop

# Setup

Đầu tiên cần cái nodejs, sau đó tạo 3 file

* Package.json :file khai báo các package cho project
* Main.js :file chạy đầu tiên khi mở project
* Index.html :file view của main.js

## *Trong file package.json:*

{

    "name": "Demo",

    "description": "demo for html app",

    "authors": "black cat",

    "version": "1.0.0",

    //tên file sẽ chạy đầu tiên

    "main": "main.js",

    "dependencies": {},

    // các package sẽ tự động được add tại đây

    "devDependencies": {

        "electron": "^7.1.5"

    },

    "scripts": {

        //lệnh đầu tiên khi bắt đầu chạy

        //lệnh này để mở pj bằng electron

        "start": "electron ."

    }

}

## Trong file main.js:

const { app, BrowserWindow } = require('electron');

const path = require("path");

const url = require("url");

function createWindow() {

    // tạo một cửa sổ mới

    let win = new BrowserWindow();

    // khai báo view cho cửa sổ đó

    win.loadURL(url.format({

        pathname: path.join(\_\_dirname, 'index.html'),

        protocol: 'file',

        slashes: true

    }));

    // mở dev tool

    win.webContents.openDevTools();

    // bắt sự kiện tắt cửa sổ

    win.on('closed', () => {

        win = null;

    })

}

// mowe app sau khi load xong

app.on('ready', createWindow);

// thoát app khi tắt cửa sổ

app.on('window-all-closed', () => {

    if (process.platform != 'darwin') {

        app.quit()

    }

});

// mở app lại

app.on('activate', () => {

    if (win == null) {

        createWindow()

    }

})

## File index.html thì code như bthg

## Khởi chạy

Gõ npm start

## Build

### Package

# for use in npm scripts

npm install electron-packager --save-dev

# for use from cli

npm install electron-packager -g

for windows

electron-packager <sourcedir> <appname> --platform=win32 --arch=x64

## Lưu ý file js

* File js được phân làm 2 dạng: main process và render process
  + Render process làm việc giống như một file js thông thường
  + Main process sẽ chịu trách nhiệm tạo và điều khiểm ui window (vd: main.js)

Vì lý do bảo mật nên sẽ ko sử dụng được lệnh node trong render, nếu muốn dùng phải thêm thuộc tính

let win = new BrowserWindow(

        {

            ...,

            webPreferences: {

                nodeIntegration: true

            }

        }

# UI window

## *BrowserWindow*

    let parentWin = new BrowserWindow();

    let win = new BrowserWindow({

        width: 400,

        height: 400,

        maxHeight: 500,

        maxWidth: 600,

        minHeight: 200,

        // màu nền

        backgroundColor: '#2c3e50',

        // bật darktheme

        darkTheme: true,

        // bỏ toolbar vs border

        frame: false,

        // đổi cái title

        title: 'demo window',

        // ẩn cửa sổ, khi gọi thì hiện

        show: false,

//nền trong suốt

        transparent: true,

        // ẩn khỏi thanh công cụ

        skipTaskbar: true,

        // ngăn chỉnh kích thước cửa sổ bằng cách kéo

        resizable: false,

        // cho phép dùng lệnh node( require, ...) trên render js

        webPreferences: {

                nodeIntegration: true

        }

        // quan hệ parent-child

        //child luôn hiển thị phía trước parent

        // chỉ định window cha

        parent: parentWin,

        // ngăn người dùng thao tác với parent khi child vẫn hoạt động

        modal: true

    });

    // còn nữa ở: https://electronjs.org/docs/api/browser-window

Render một file hay trang web

    win.loadURL('demoweb.com');

Mở cửa sổ

    win.show();

# IPC

Main ipc: native

Remder ipc: client

Có thể dùng sync

## Client

const electron = require('electron');

const ipc = electron.ipcRenderer;

document.getElementById('btn').addEventListener('click', () => {

    // send(chanel, [data])

    // data có thể không điền

    ipc.send('test', 'bello');

})

// listen

// arg là data

ipc.on('rep', (e, arg) => {

    console.log(arg);

})

## Native

    const ipc = electron.ipcMain;

    const dl = electron.dialog;

    // listen

    // arg là data

    ipc.on('test', (e, arg) => {

        //do something

        dl.showErrorBox('oh shit!', arg);

        // send(chanel, [data])

        // data có thể không điền

        e.sender.send('rep', 'send successed!')

    })

## Create window from render

const BrowerView = electron.remote.BrowserView;

const win = new BrowserView();

# Menu

App menu là thanh menu ngang mặc định của app

<https://www.youtube.com/watch?v=MGC2W-dJtYM&list=PLC3y8-rFHvwiCJD3WrAFUrIMkGVDE0uqW&index=7>

Context menu là menu hiện ra khi ấn chuột phải

<https://www.youtube.com/watch?v=MkVLaM9JAxM&list=PLC3y8-rFHvwiCJD3WrAFUrIMkGVDE0uqW&index=8>

# bắt sự kiện bàn phím

<https://www.youtube.com/watch?v=-POUsDEfI14&list=PLC3y8-rFHvwiCJD3WrAFUrIMkGVDE0uqW&index=9>

# file

const electron = require('electron');

const fs = require('fs');

const path = require('path');

const content = document.getElementById('content');

// tạo đường dẫn

let pathName = path.join(\_\_dirname, 'Files')

document.getElementById('btn').addEventListener('click', () => {

    // tên file

    var name = 'mo';

    // link file

    let file = path.join(pathName, name);

    // tạo

    fs.appendFile(file, 'Hello content!', function (err) {

        if (err) throw err;

        console.log('Saved!');

    });

    // sửa

    fs.writeFile(file, 'klasjdkashdjk', (err) => {

        if (err) {

            console.log('ERR: ' + err);

        }

        else console.log('successed!');

    })

    // đọc

    fs.readFile(file, (err, data) => {

        content.innerText = data;

    })

})

Nhiều nữa

<https://www.w3schools.com/nodejs/nodejs_filesystem.asp>

# Tips vs trick

## Drag window

#### Thêm thuộc tính css vào một element để di chuyển window bằng cách kéo thả element đó

    -webkit-user-select: none;

    -webkit-app-region: drag;

## Glass window

Bật bỏ qua sự kiện chuột bằng cách

    win.setIgnoreMouseEvents(true)