1. 區塊鏈就是一群區塊組成的資料庫，區塊的意思就是存放交易紀錄的檔案，具有匿名功用，由所有節點共享

比特幣的主鏈即資料連結中最長的鏈，而連結通常有一定關係，交易資料的連結點在於紀錄金錢的流向，紀錄金錢的來源上下游的關係。

區塊鏈是一種不依賴第三方通過網路存取驗證的技術，在老師的舉例中就是人人有一本記帳本，是一種去中心化的大型網路記帳簿。

在區塊連結到前一個區塊時持續數個以上時，該交易就有所謂的不可逆轉的確認這筆交易的存在，即注重多數人驗證來確實交易的存在。

1. 根據老師課堂所說的是透過雜湊的方式去防止竄改資料，交易紀錄透過雜湊轉換成一個區塊，這個區塊寫入上一個區塊的雜湊值，一連串產生連鎖效應，如果中途的資料被竄改會導致後面資料的雜湊值解析不出來，因此只要每個區塊連接點有儲存所有交易結果的雜湊值，透過對照可以以此來判斷哪些交易資料被修改過。

創始區塊:A Hash = 50 preHash = 0

A – B B Hash = 250 preHash = 50

A – C C Hash = 200 preHash = 50

C – D D Hash = 400 preHash = 200

A-C-D 為主鏈 B成為支鏈(孤立)

如果雜湊時A-C的資料被修改(紅字) 改成190

A Hash = 50 preHash = 0

A – B B Hash = 250 preHash = 50

A – C C Hash = 190 preHash = 50

C – D D Hash = 400 preHash = 200

那麼C-D這條就會對應不回去就能判斷出資料的修改了。

1. 比特幣的手續費計算是可以依照交易金額大小計費

不過另一種說法是可以自定義的 手續費高低和確認時間的長短有關係，手續費越多礦工就越會把你的交易紀錄加入區塊中。

通常比特幣的手續費不能低於某個值，會有一個基本的默認值

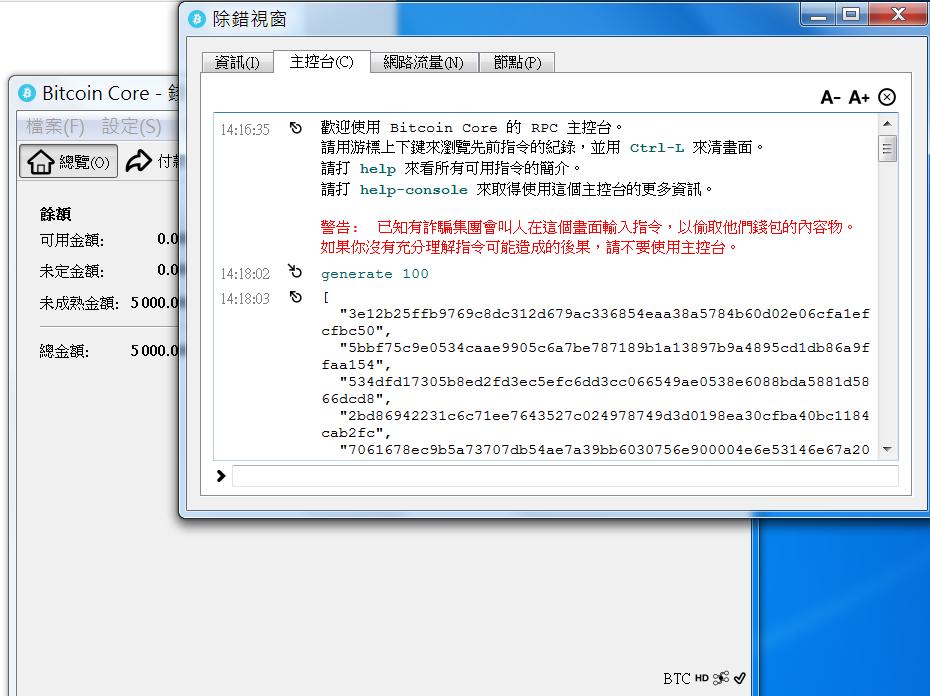
以太坊的手續費是依照計算資源計費

根據不同的計算資源會有不同的價格換算表

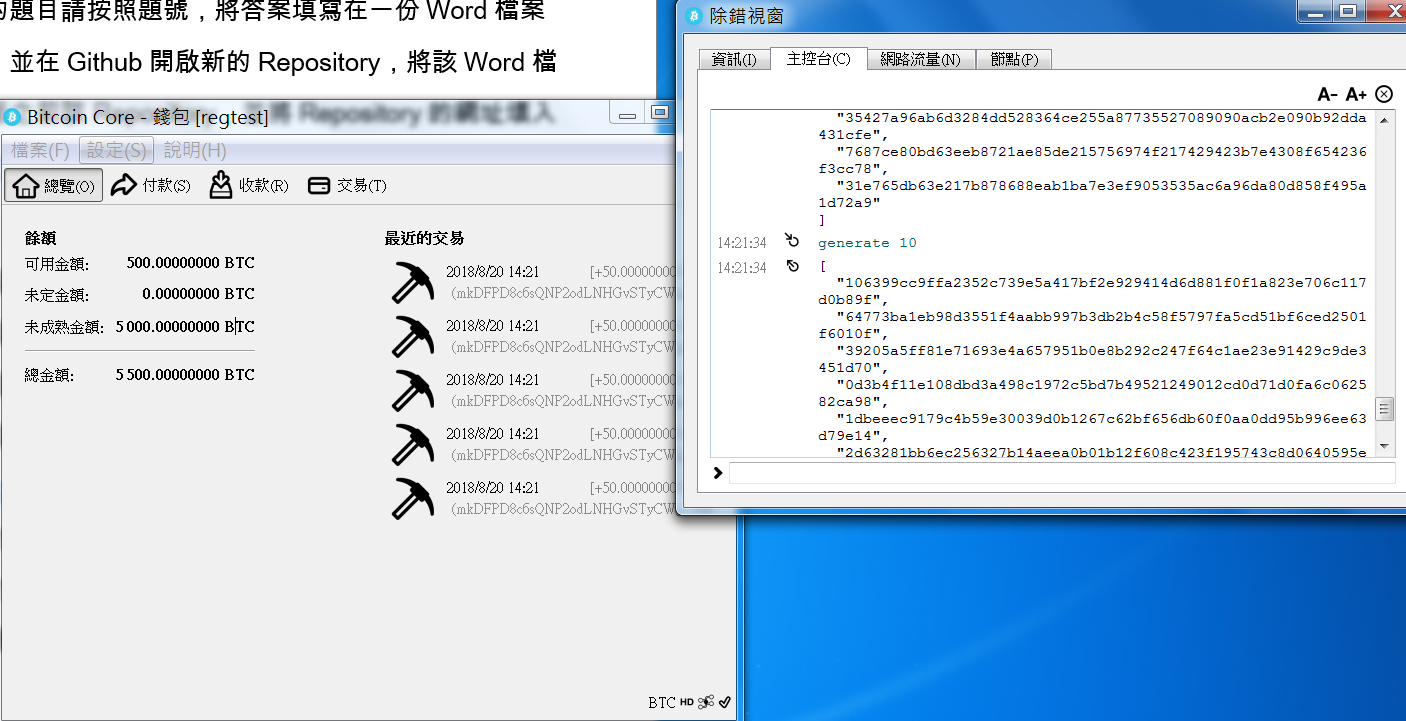
而透過這些價格與運算所用的次數來計算出最後手續費的金額。

cost = gas Price × gas Limit

手續費多少通常透過找零的金額往回推算即可得知

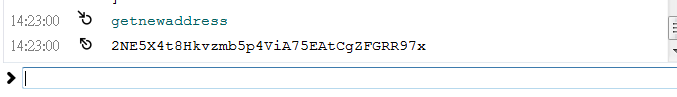
**4.** 

先產生錢幣(製造區塊100個)拿到收入



在一次10讓他成熟

接著下指令: getnewaddress 取得一個位址

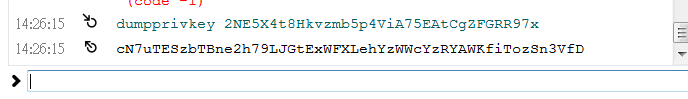


自己的位置為: 2NE5X4t8Hkvzmb5p4ViA75EAtCgZFGRR97x

進行私鑰加密取得PK

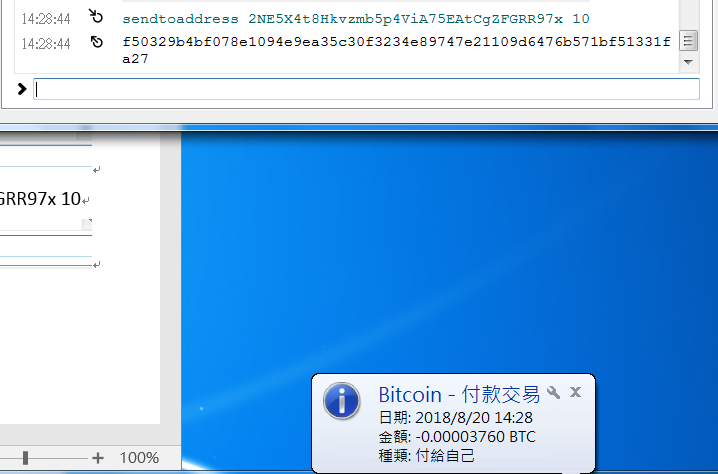


該位置的PK為 cN7uTESzbTBne2h79LJGtExWFXLehYzWWcYzRYAWKfiTozSn3VfD

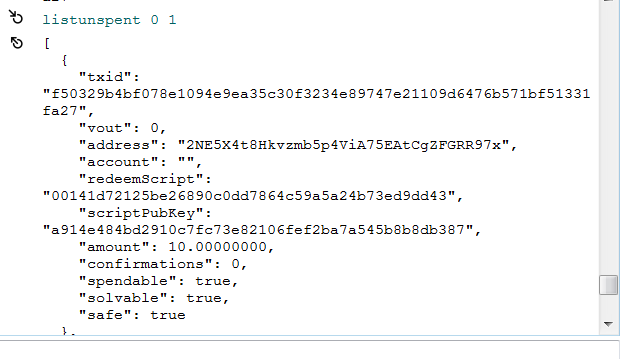


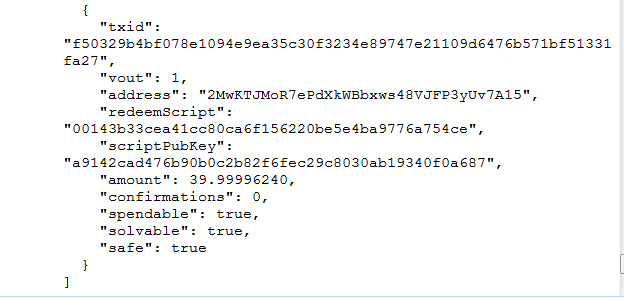
寄給這個位置10元 sendtoaddress 2NE5X4t8Hkvzmb5p4ViA75EAtCgZFGRR97x 10





listunspent 0 1 查看交易資訊





第一圖為交易內容

第二圖為餘額

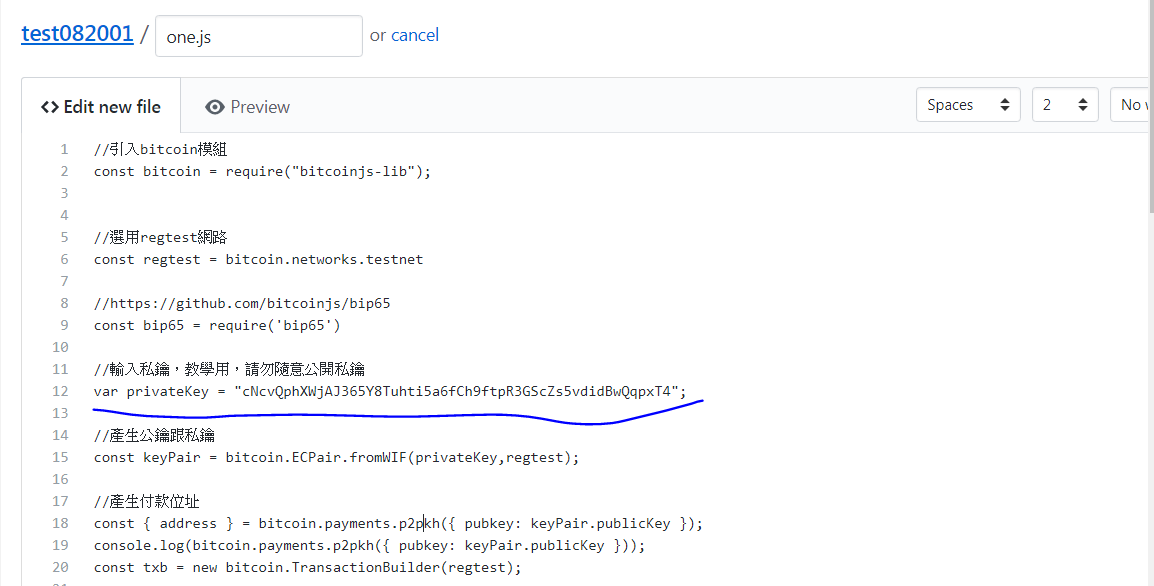
記錄下此交易後的餘額資訊:

"txid":"f50329b4bf078e1094e9ea35c30f3234e89747e21109d6476b571bf51331fa27

"vout": 1

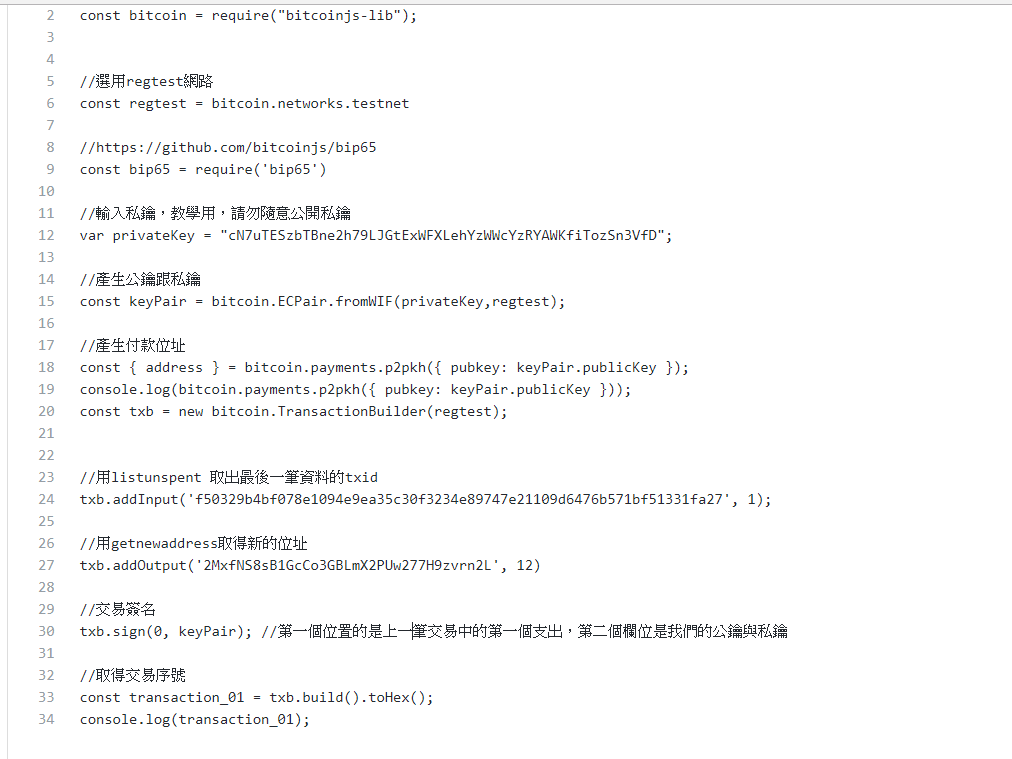
**5.編輯程式碼:**

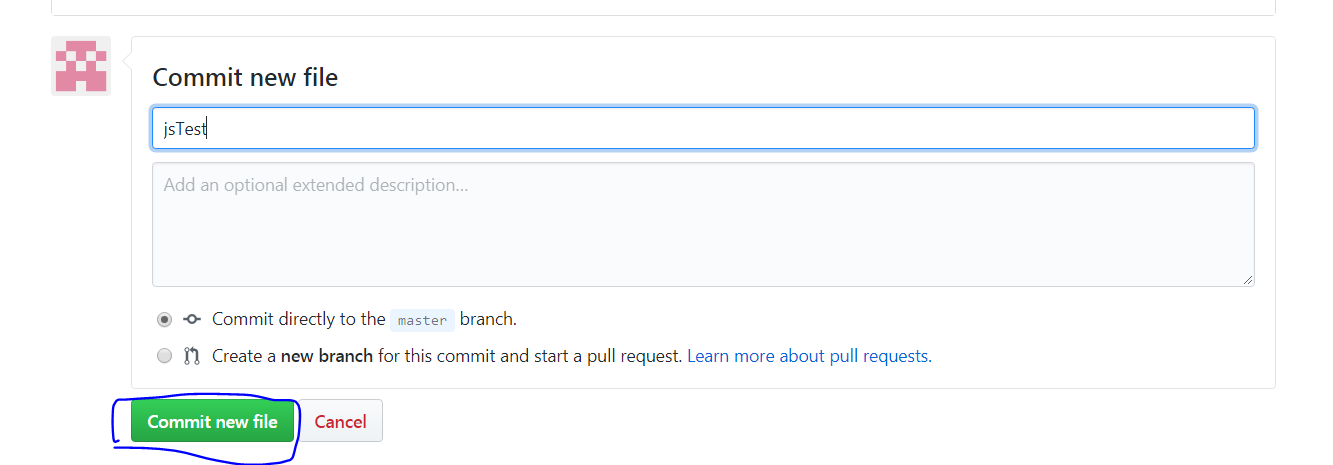
改寫兩個區塊，第一個為address私鑰，第二個為交易txid



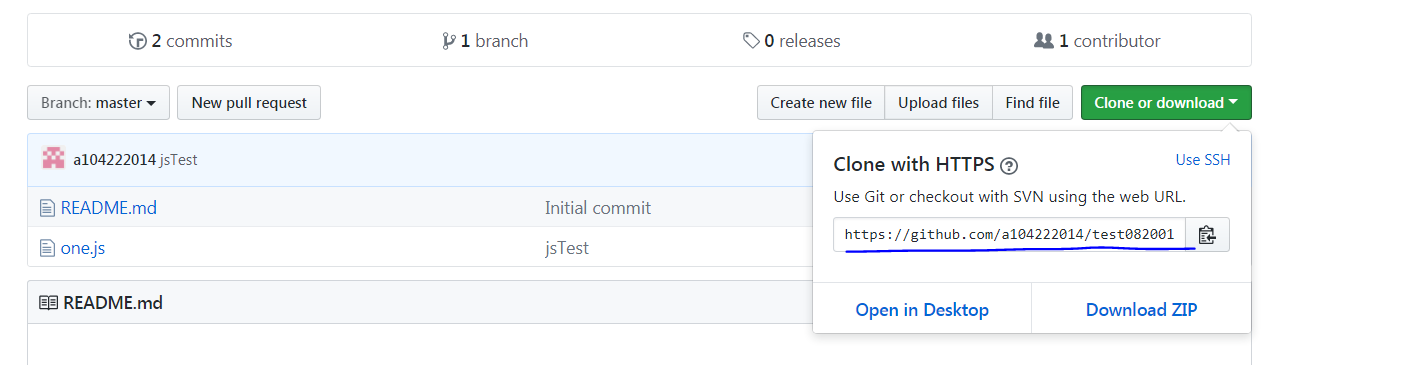


根據上方紅字修改完的結果:

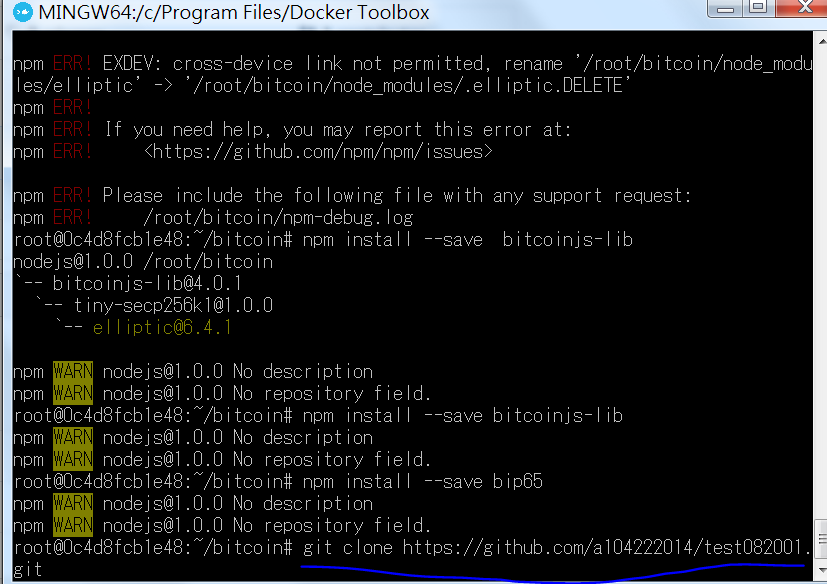




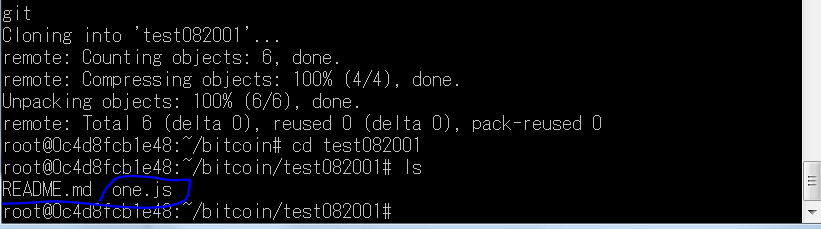
儲存後成功上傳



複製網址



Docker上貼上網址後成功下載



確定下載成功後，用NodeJS執行one.js檔(即剛剛編寫的程式)

輸入nodejs one.js



執行結果。

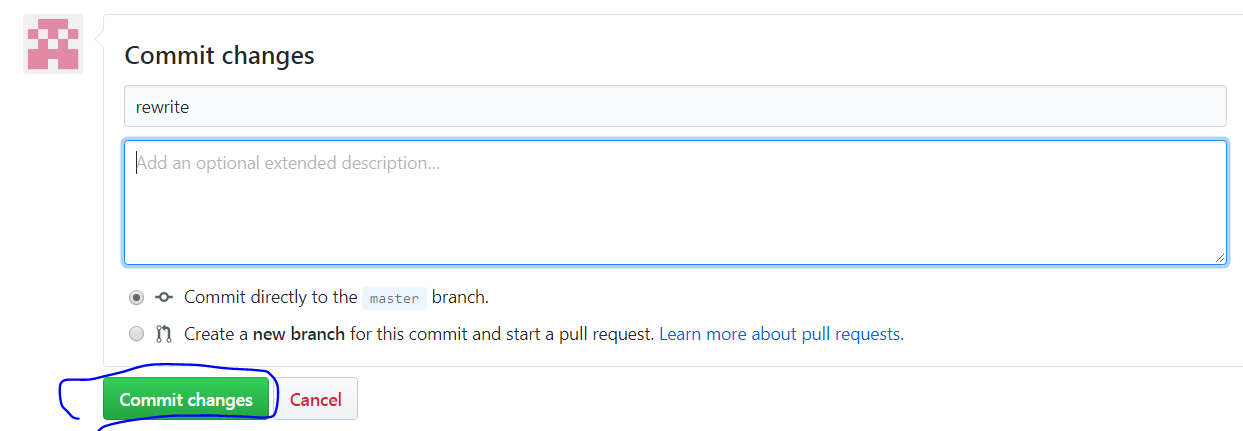
修正問題:



輸出的程式改寫，在gitHub上編輯

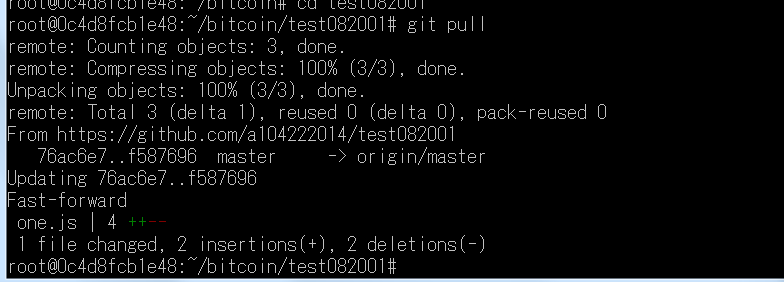


修改為上題的地址，後面的數字應該是代表交易25元



再一次儲存

在Docker上輸入 git pull



檔案更新，輸入 nodejs one.js 運行

執行結果:



// Code:

//引入bitcoin模組

const bitcoin = require("bitcoinjs-lib");

//選用regtest網路

const regtest = bitcoin.networks.testnet

//https://github.com/bitcoinjs/bip65

const bip65 = require('bip65')

//輸入私鑰，教學用，請勿隨意公開私鑰

var privateKey = "cN7uTESzbTBne2h79LJGtExWFXLehYzWWcYzRYAWKfiTozSn3VfD";

//產生公鑰跟私鑰

const keyPair = bitcoin.ECPair.fromWIF(privateKey,regtest);

//產生付款位址

const { address } = bitcoin.payments.p2pkh({ pubkey: keyPair.publicKey });

console.log(bitcoin.payments.p2pkh({ pubkey: keyPair.publicKey }));

const txb = new bitcoin.TransactionBuilder(regtest);

//用listunspent 取出最後一筆資料的txid

txb.addInput('f50329b4bf078e1094e9ea35c30f3234e89747e21109d6476b571bf51331fa27', 1);

//用getnewaddress取得新的位址

txb.addOutput('2NE5X4t8Hkvzmb5p4ViA75EAtCgZFGRR97x', 25)

//交易簽名

txb.sign(0, keyPair); //第一個位置的是上一筆交易中的第一個支出，第二個欄位是我們的公鑰與私鑰

//取得交易序號

const transaction\_01 = txb.build().toHex();

console.log(transaction\_01);