程式人

十分鐘系列

區塊鏈

（比特幣背後的關鍵技術）

陳鍾誠

2020 年 10 月 1 日

這幾年

- 程式領域最熱門的技術
- 應該是這兩項：
  - 深度學習
  - 區塊鏈

昨天

- 我們已經談過深度學習背後的《梯度下降法》

https://www.slideshare.net/ccckmit/ss-238680742

今天

- 就讓我們來研究一下區塊鏈

傳統上

- 當你想學一門技術

你可能會去找書

於是你找到區塊鏈的書

https://search.books.com.tw/search/query/key/%E5%8D%80%E5%A1%8A%E9%8F%88/cat/all

還真是不少

- 光博客來網站就有 617 筆

然後你看了幾本

- 會發現自己還是搞不懂
- 到底甚麼是區塊鏈！

有人說

- 免費的東西，其實是最貴的！

但我說

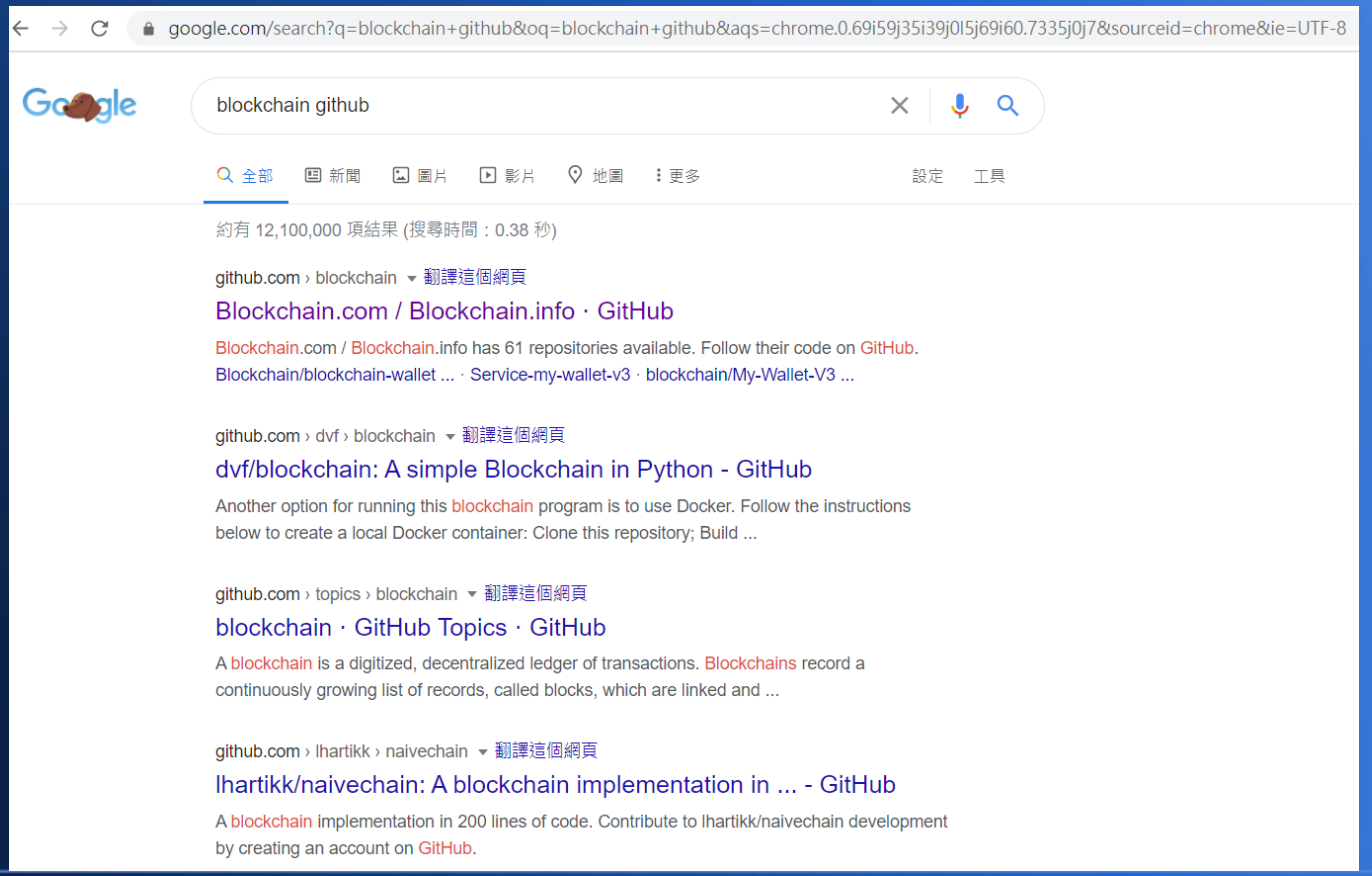
- 在程式的領域，最好的東西經常都是免費的！

只要懂得找 github

- 又何必看那些，連程式碼都沒有的書呢？

在 github 上

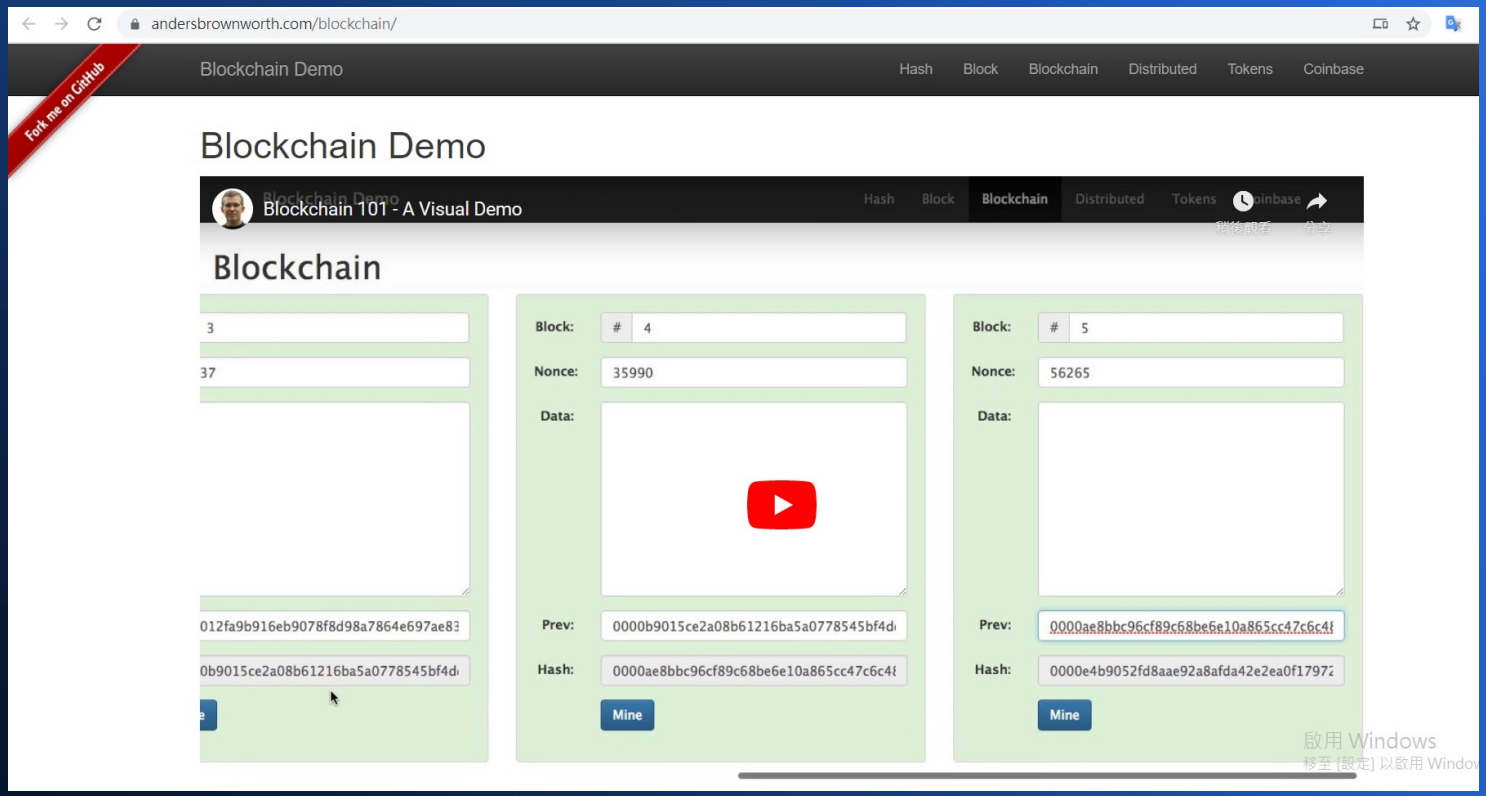
- 你可以找到各式各樣的區塊鏈實作
- 各種程式語言都有！



https://www.google.com/search?q=blockchain+github

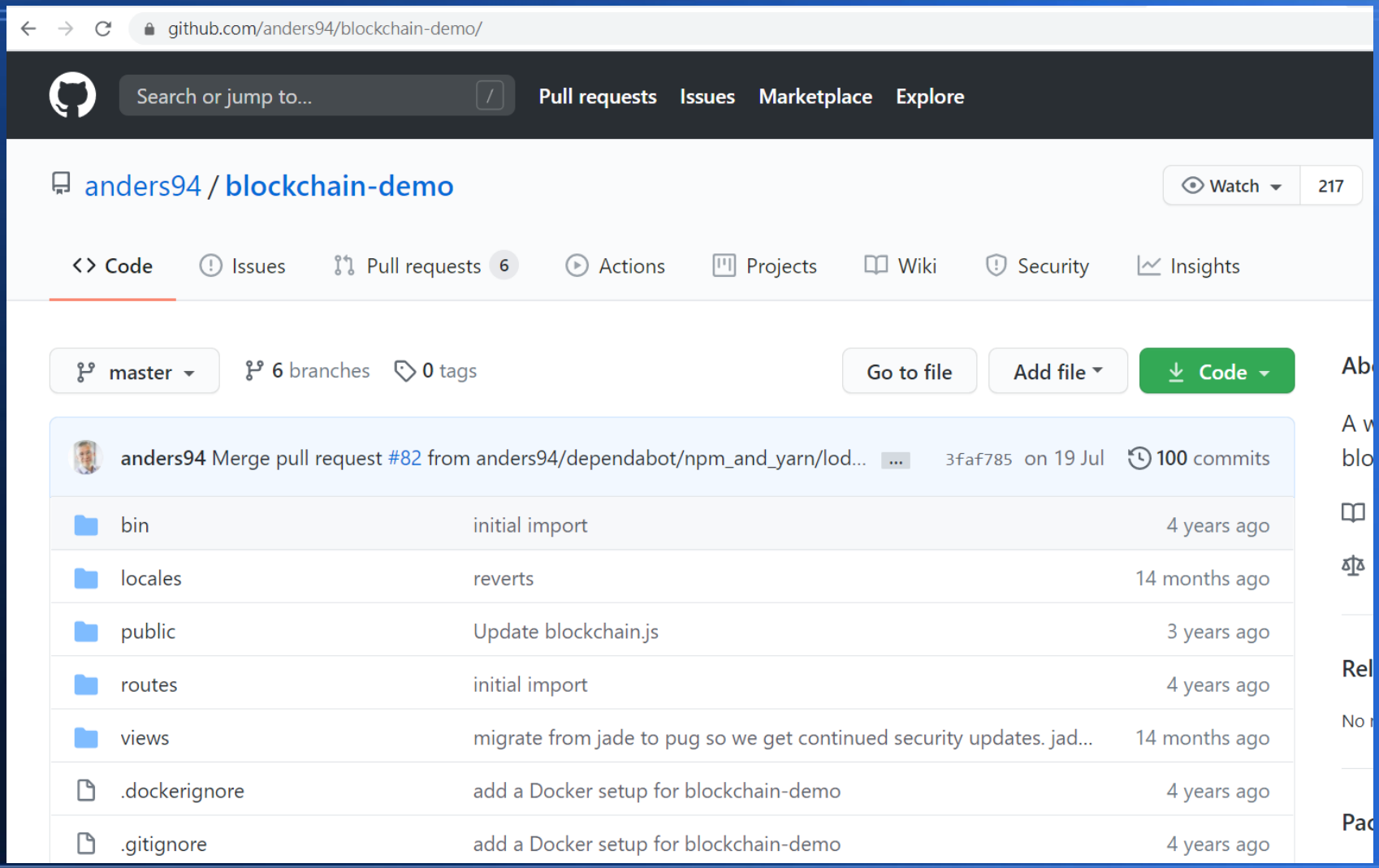
其中這個專案

- 是我最喜歡的一個！



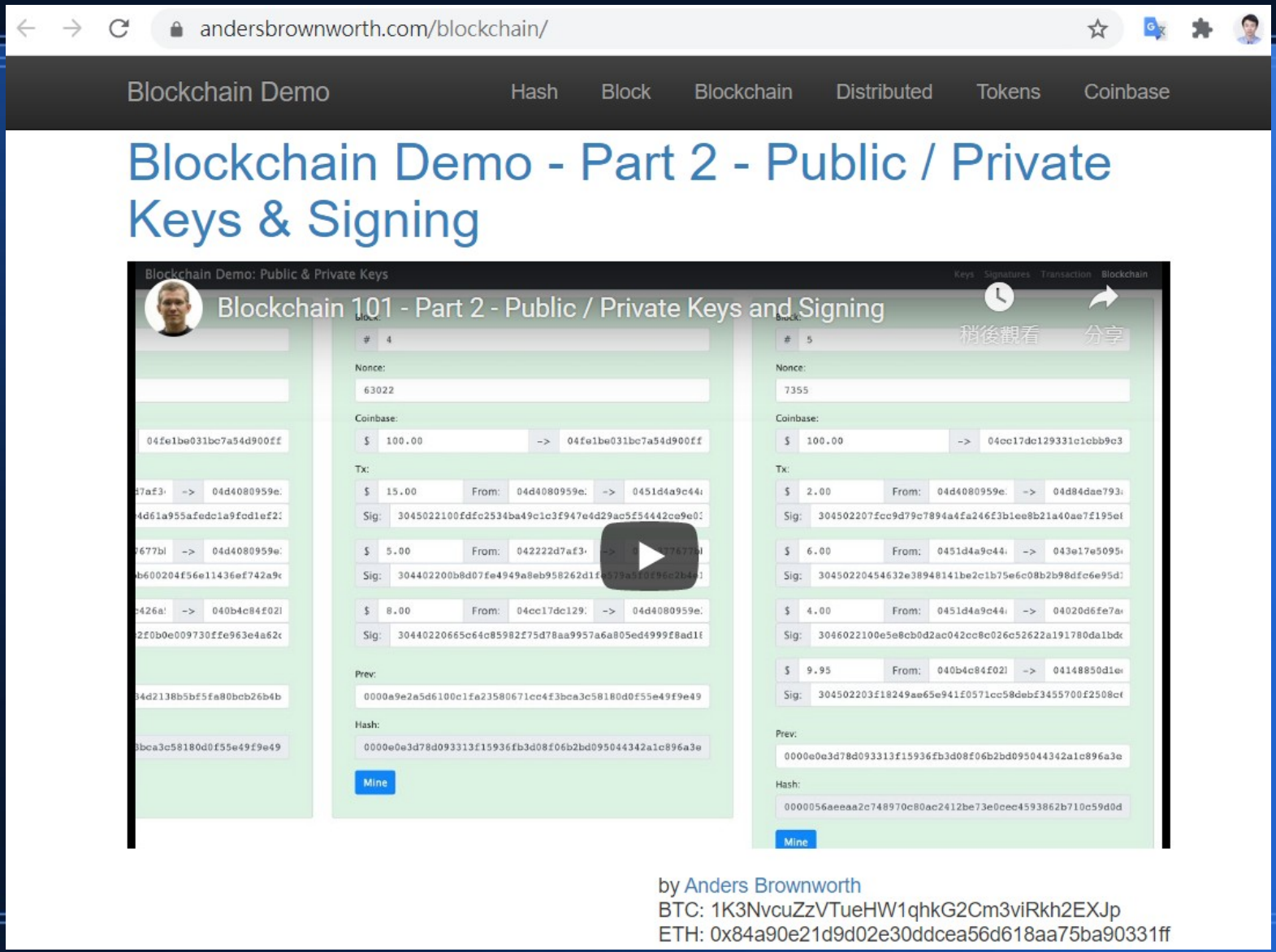
https://andersbrownworth.com/blockchain/

不只有程式碼



https://github.com/anders94/blockchain-demo/

還有影片說明



https://andersbrownworth.com/blockchain/

只要你有基本的程式觀念

- 看完這兩段影片
- 勝過看完博客來的那 617 本書！

現在

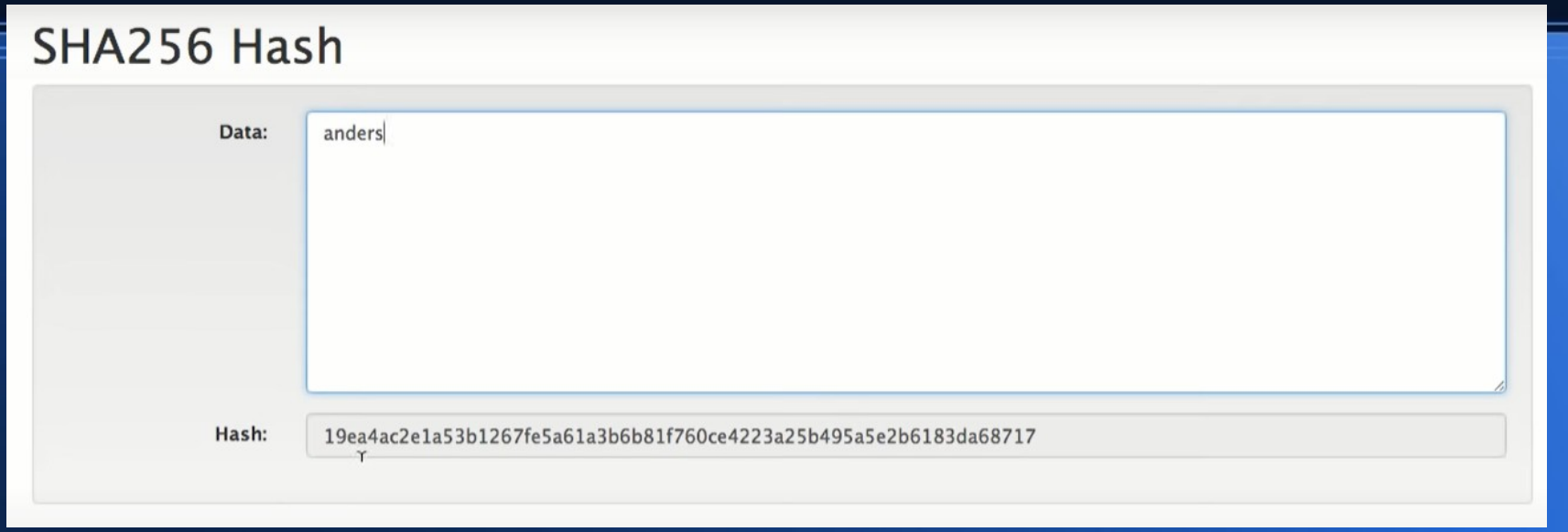
- 就讓我們跟著影片
- 來學學區塊鏈吧！

比特幣的區塊鏈運作

- 是依靠一種稱為 SHA256 的雜湊函數

https://andersbrownworth.com/blockchain/

不同的輸入會有不同的雜湊值

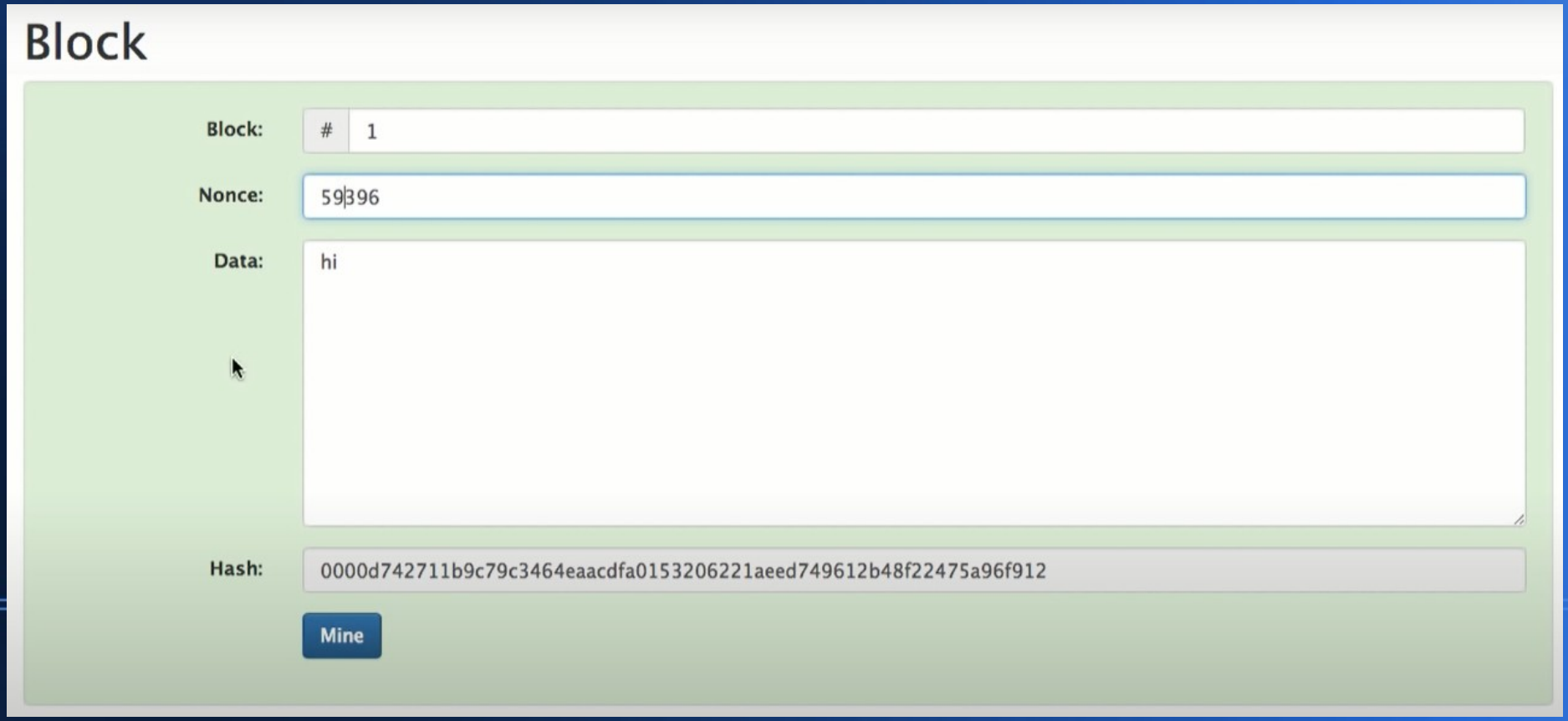


一個區塊

- 是一筆包含雜湊值的紀錄

而挖礦的過程

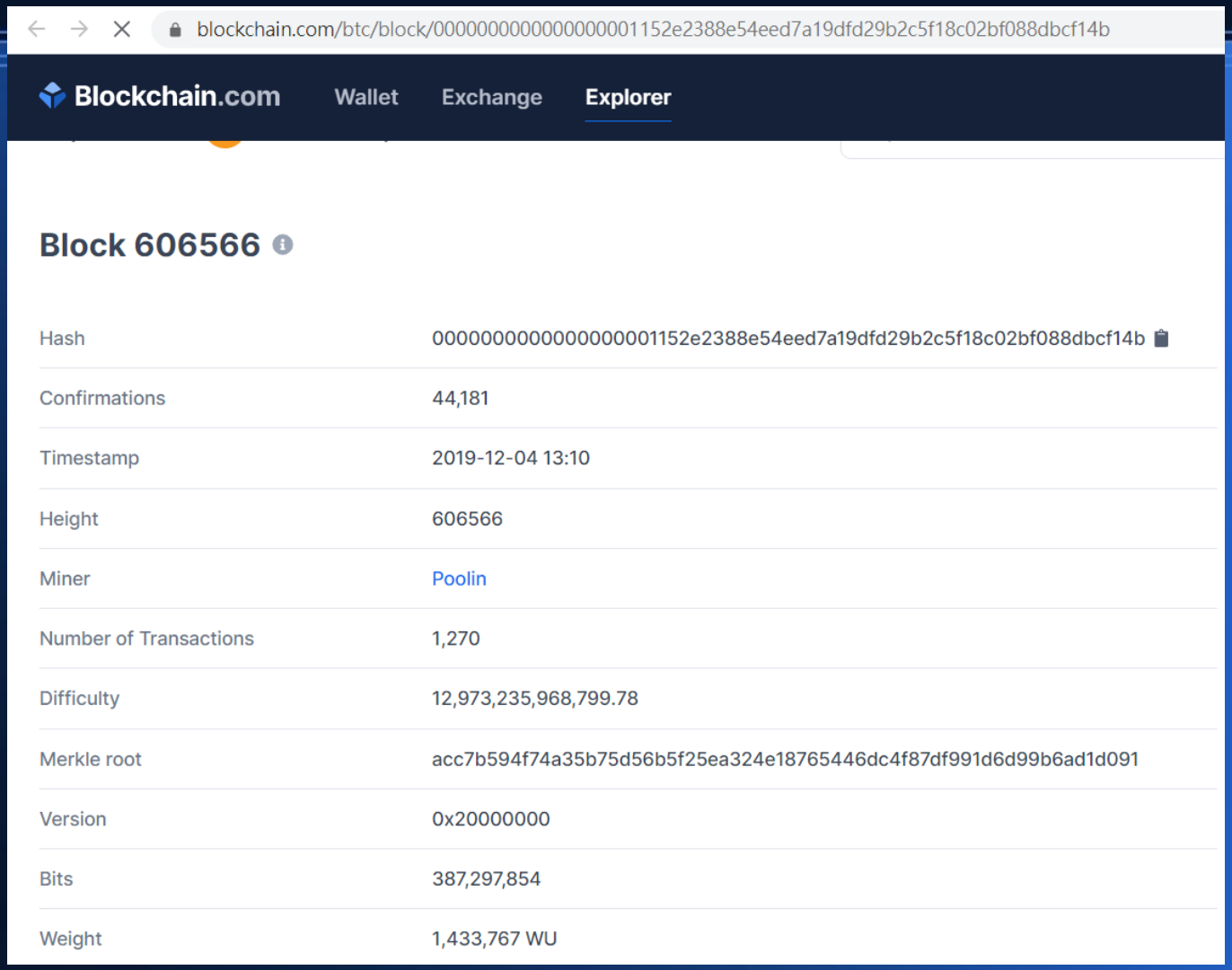
- 就是要找出紀錄上填入甚麼 nonce 值，才會讓 hash 有指定位數的前導零（例如下圖中 hash 為 0000d742....，那就是有 4 個字的前導零）



比特幣的挖礦

- 是根據前十分鐘的挖礦難易度決定下次的前導零數量

像是這個區塊就有 18 個前導零



https://www.blockchain.com/btc/block/000000000000000001152e2388e54eed7a19dfd29b2c5f18c02bf088dbcf14b

將 16 進位換成 2 進位

- 一個 16 進位對應 4 個 bits
- 所以 18 個前導零相當於 18*4=72 個 bit 的零
- 要找到這樣的 nonce，機會只有 $1/2^{72}$
- 這也是為何挖礦要很多算力的原因！
- 因為你的程式平均要嘗試 2^{72} 次才能找到一個符合的 nonce 填上。

所以

- 比特幣的挖礦，基本上就是個猜 nonce 的遊戲！

和猜謎不同的是

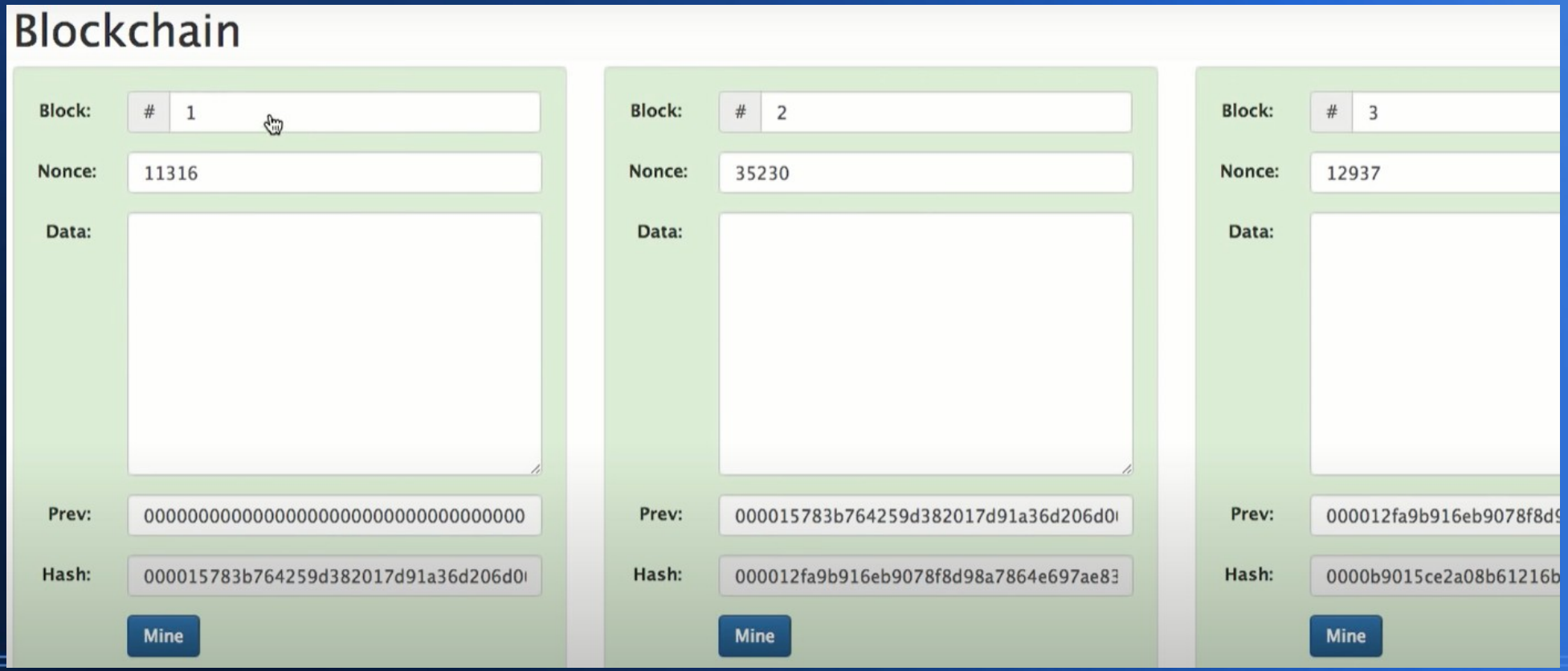
- 電腦可以用全速去猜，速度愈快的電腦，可以猜愈多次，猜中的機會也就愈大！

然後

- 我們就可以講解《區塊鏈》的觀念了！

在區塊鏈上

- 後面的區塊必須記錄前面的 hash 值
  而且填入的 nonce 還要讓自己區塊有足夠的前導零

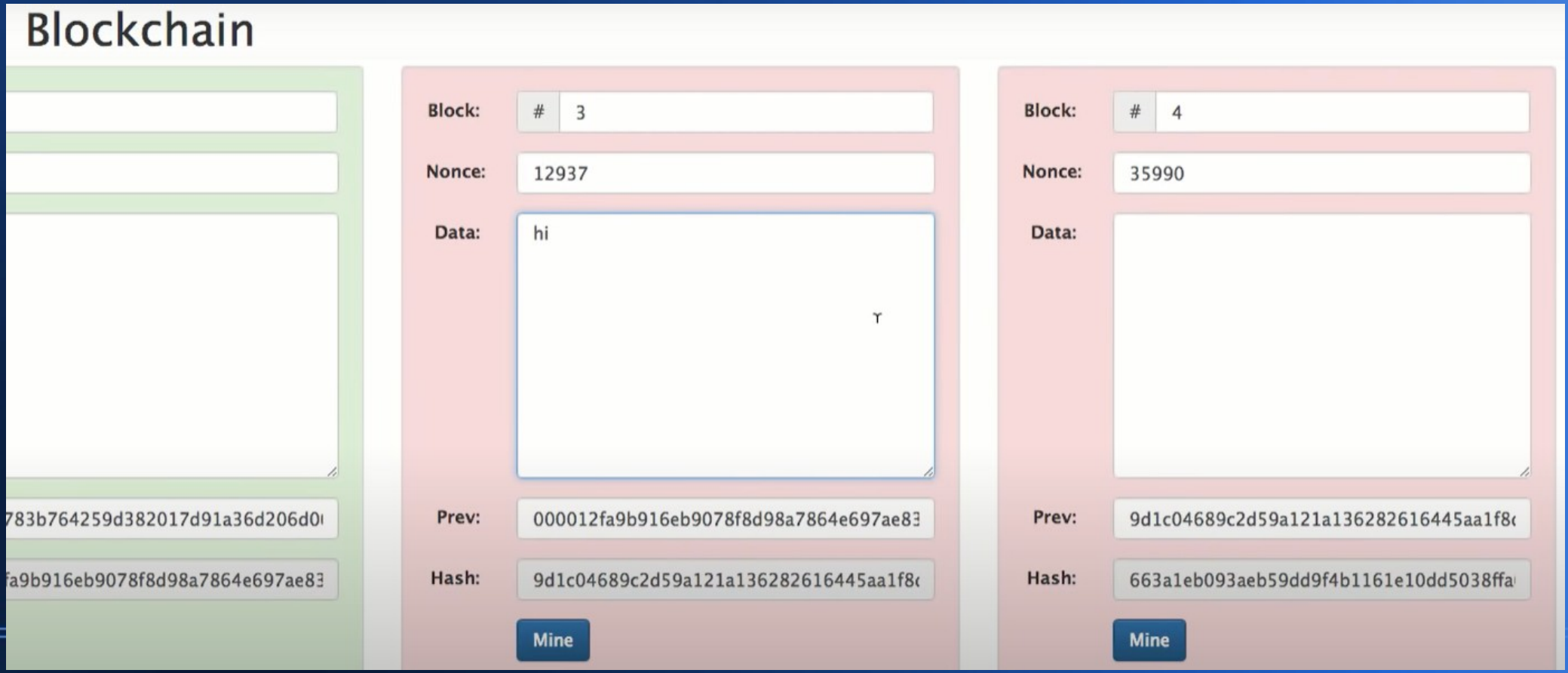


於是

- 區塊一個串一個，每個都要滿足前導零的要求
- 任何一個不滿足，都很容易被快速檢查出來
- 這就是區塊鏈的《易檢查、難填充》特性！

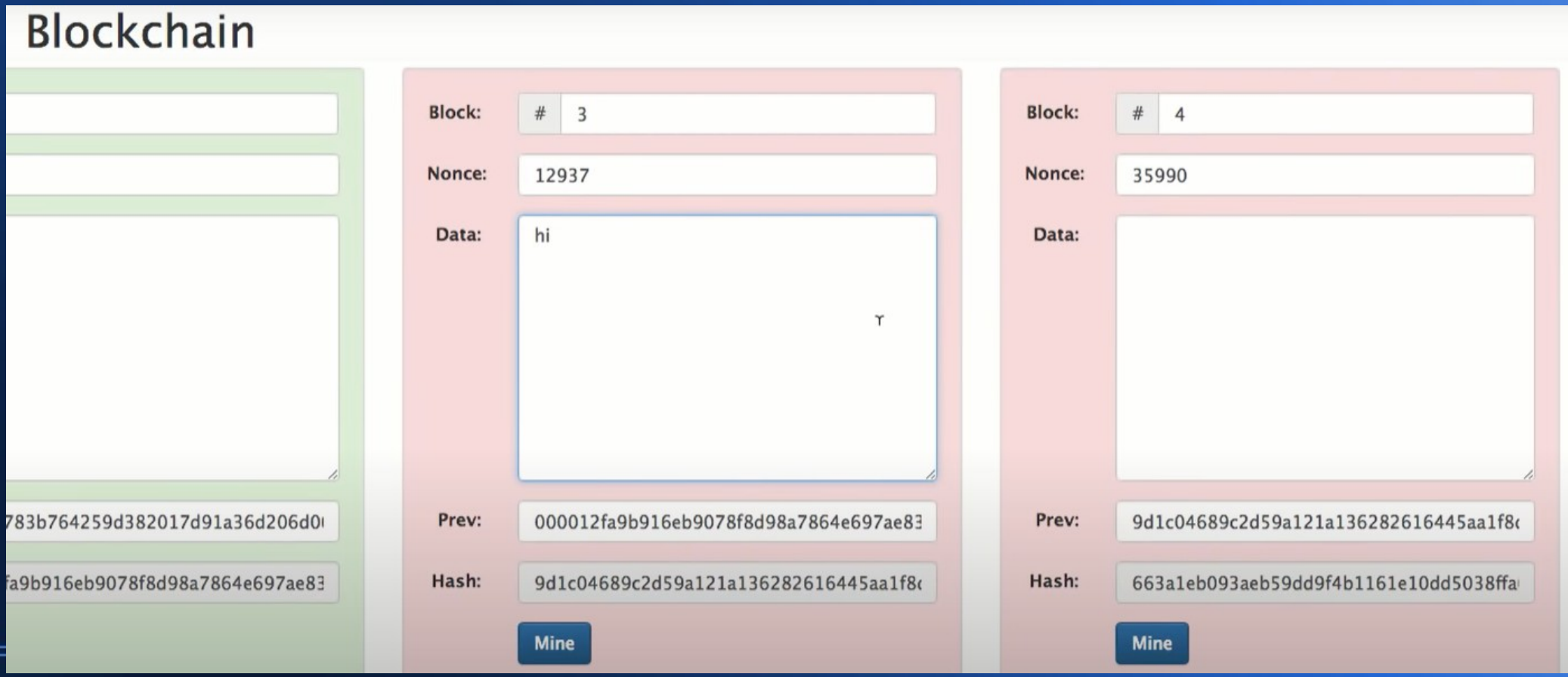
如果你想竄改區塊中的任何一個字

- 會造成雜湊值不正確



唯一的解決方法

- 就是把竄改紀錄後的那些區塊，全部重新挖一遍！



但是當你竄改了

- 其他人所記錄的區塊鏈並沒有改
- 根據多數決法則，你很容易就被發現是竄改的！

於是

- 區塊鏈的不可竄改性就得到了確保！

這就是

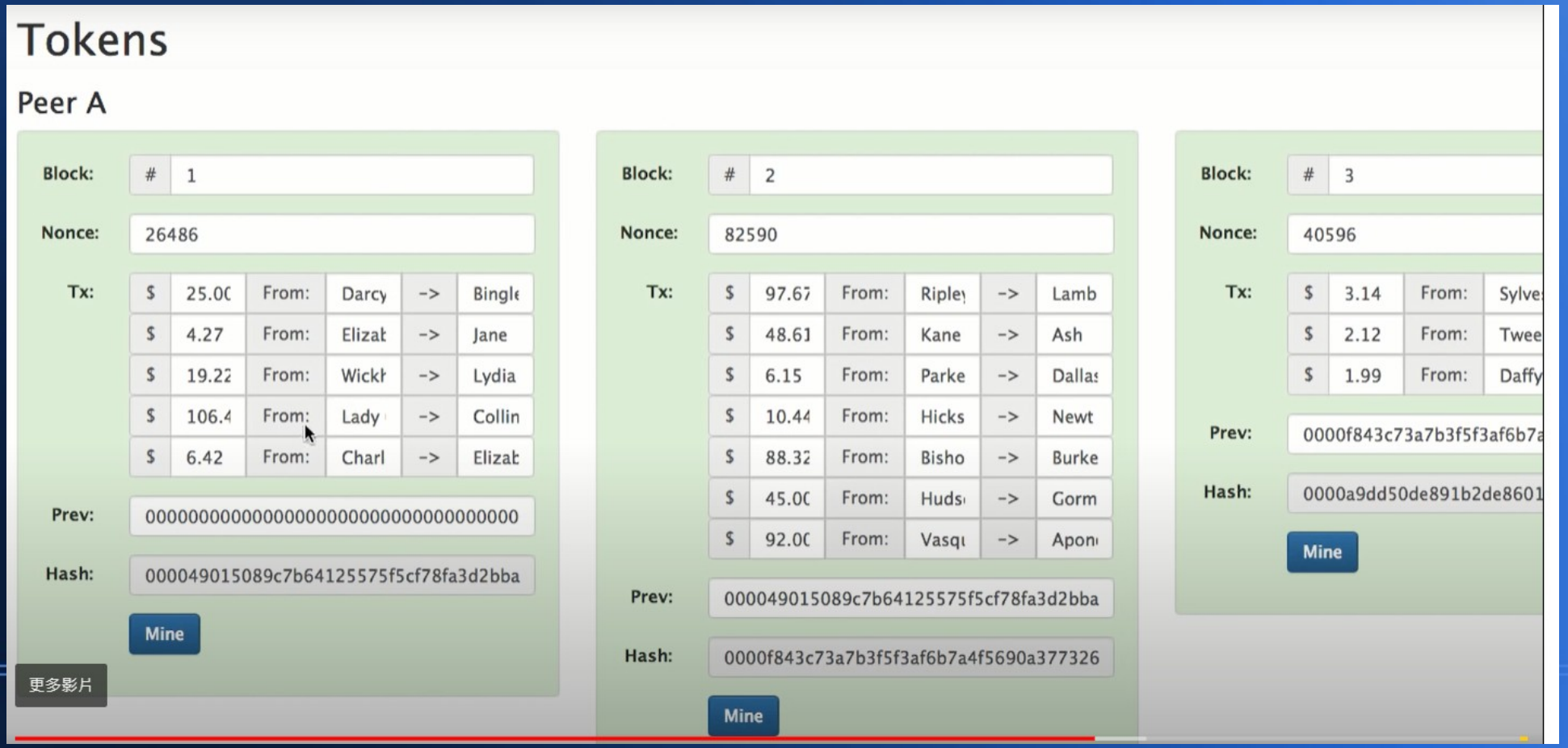
- 比特幣為何在沒有中央控管的狀況下
- 卻可以達到分散式認證的原因了！

而乙太幣

- 則是在區塊鏈上又加入了智能合約
- 導致乙太幣有可能成為分散式金融 DeFi（Decentralize Finance）的主要技術！

比特幣的紀錄

- 其實還記載了交易資訊



於是你可以買賣比特幣

- 或者用比特幣消費！

但如果你

- 想用比特幣來買珍珠奶茶

那可能會大失所望！

因為每十分鐘

- 全世界只會產生一個大小約 4MB 的比特幣區塊

如果用比特幣買珍奶

- 勢必得等到下一個區塊驗證好

於是你

- 得在奶茶店等十分鐘 …
- 才知道交易是否成功
- 你的奶茶消費有沒有成功被排入下一個區塊 . . .

所以

- 別總是想著用比特幣買珍奶

雖然

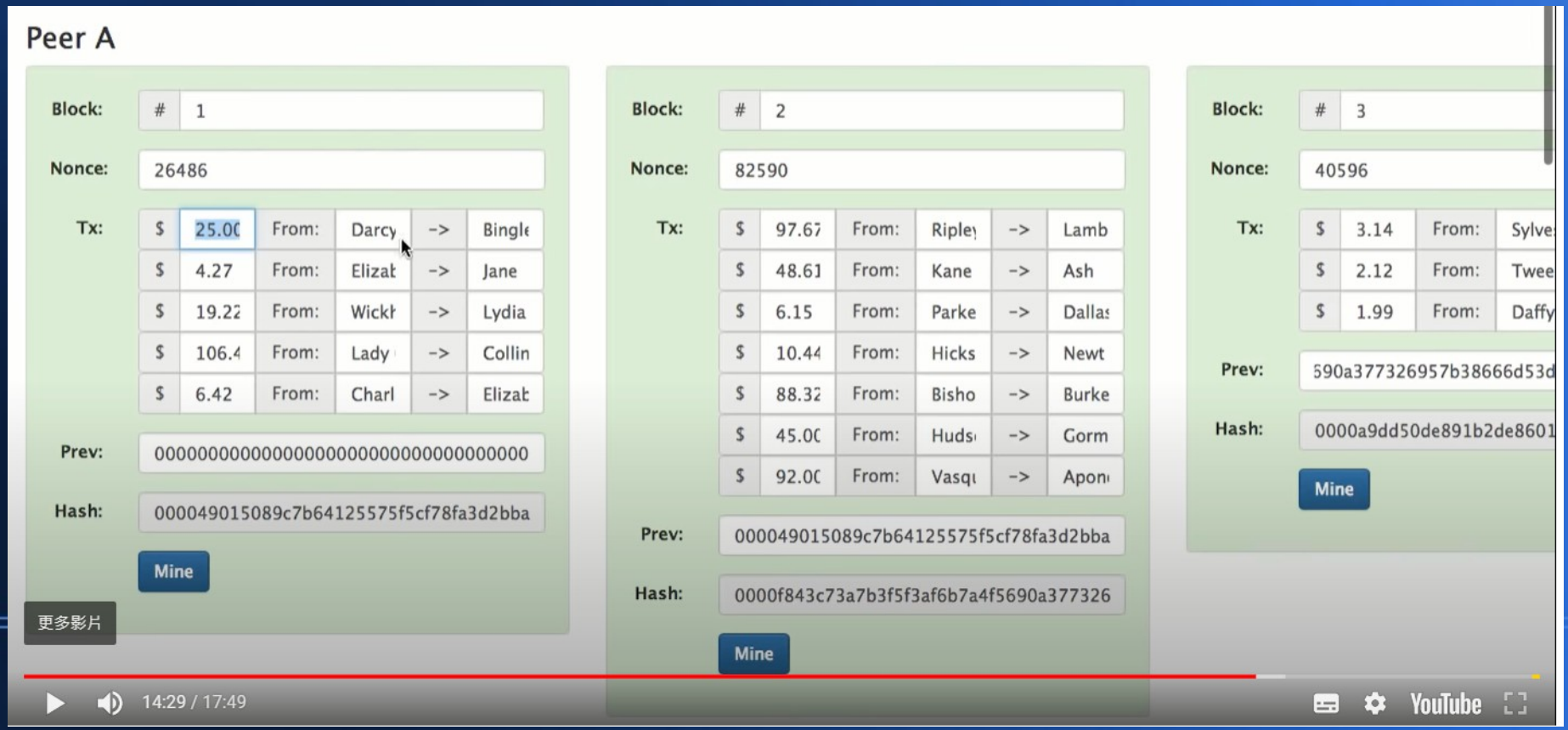
- 早期真的有人用比特幣買披薩
- 而且還真的成功了！
- 這個交易成了歷史上的傳奇

史稱《比特幣披薩日》

https://www.blocktempo.com/bitcoin-pizza-day-event-may-twenty-second/

最後要提醒大家

- 比特幣的交易紀錄，並不是像下圖中採用人名而是以《電子錢包》的《公開金鑰》紀載的，只是個代號，所以你並不會知道交易者的姓名！



還有一個技術問題是

- 如果你的比特幣錢包沒有錢，卻又假裝有錢去和別人做交易，那會怎麼樣呢？

關於這點你不用擔心

- 由於比特幣是以 coinbase 的方式控管的，所以沒錢的不能假裝有錢去交易！

這類的驗證

- 就牽涉到比特幣背後的《公鑰／私鑰》機制
- 也就是密碼學中的非對稱式加解密的 RSA 算法和橢圓演算法所確保的。

進一步的內容

- 還是請大家直接看影片，會更生動活潑且清楚！

對了

- 如果你想寫個簡易的挖礦程式，可以參考這個 node. js 範例

```
const crypto = require('crypto');

let record = {
  nonce: 0,
  data: 'john => mary : $2.7; george => john : $1.3',
}

function hash (text) {
  return crypto.createHmac('sha256', '').update(text).digest('hex')
}

function mining(record) {
  for (var nonce=0; nonce<1000000000000; nonce++) {
    record.nonce = nonce
    let h = hash(JSON.stringify(record))
    if (h.startsWith('00000')) return { nonce: nonce, hash: h }
  }
}

console.log(mining(record))
```

https://github.com/ccccourse/se/blob/master/algorithm/08-bruteForce/digitalcach/mining.js

程式非常簡單

就是一直調整 nonce
後算雜湊值，看看 hash
是否有夠多的前導零
有的話就是挖到了！

```
const crypto = require('crypto');

let record = {
  nonce: 0,
  data: 'john => mary : $2.7; george => john : $1.3',
}

function hash (text) {
  return crypto.createHmac('sha256', '').update(text).digest('hex')
}

function mining(record) {
  for (var nonce=0; nonce<1000000000000; nonce++) {
    record.nonce = nonce
    let h = hash(JSON.stringify(record))
    if (h.startsWith('00000')) return { nonce: nonce, hash: h }
  }
}

console.log(mining(record))
```

https://github.com/ccccourse/se/blob/master/algorithm/08-bruteForce/digitalcach/mining.js

當然也可以用亂數挖

- 這樣就算別人程式比你快，你還是會有點機會挖到的！

```
const crypto = require('crypto');

let record = {
  nonce: 0,
  data: 'john => mary : $2.7; george => john : $1.3',
}

function hash (text) {
  return crypto.createHmac('sha256', '').update(text).digest('hex')
}

function mining(record) {
  // for (var nonce=0; nonce<1000000000000; nonce++) {
  while (true) {
    let nonce = Math.floor(Math.random()*100000000)
    record.nonce = nonce
    let h = hash(JSON.stringify(record))
    if (h.startsWith('00000')) return { nonce: nonce, hash: h }
  }
}

console.log(mining(record))
```

https://github.com/ccccourse/se/blob/master/algorithm/08-bruteForce/digitalcach/miningRandom.js

這就是

- 比特幣中區塊鏈技術的原理

希望您會喜歡

我們今天的

- 十分鐘系列！

我們下次見

Bye bye !