WebSocket 簡介 -
用 nodejs 實作 WebSocket chat

by Fillano

2010/9/15

WebSocket Interface

- 規格網址
- 介面定義

```
[Constructor(in DOMString url, in optional DOMString protocol)]
interface WebSocket {
  readonly attribute DOMString URL;
  // ready state
  const unsigned short CONNECTING = 0;
  const unsigned short OPEN = 1;
  const unsigned short CLOSED = 2;
  readonly attribute unsigned short readyState;
  readonly attribute unsigned long bufferedAmount;
  // networking
        attribute Function onopen;
        attribute Function onmessage;
        attribute Function onclose;
  boolean send(in DOMString data);
  void close();
};
WebSocket implements EventTarget;
```

其實很簡單

- 利用 `new WebSocket('網址','次協定')` 跟伺服器建立連結。
- 連結成功後， `.readyState` 會變成 `1`
- 用 `.send()` 送出資料
- 用 `.onmessage` 事件接收資料，資料會透過 `MessageEvent` 事件傳送（參考下一頁）
- 用 `.close()` 結束連結

MessageEvent Interface

- 規格網址
- 介面定義

```
interface MessageEvent : Event {
  readonly attribute any data;
  readonly attribute DOMString origin;
  readonly attribute DOMString lastEventId;
  readonly attribute WindowProxy source;
  readonly attribute MessagePortArray ports;
  void initMessageEvent(in DOMString typeArg,
                        in boolean canBubbleArg,
                        in boolean cancelableArg,
                        in any dataArg,
                        in DOMString originArg,
                        in DOMString lastEventIdArg,
                        in WindowProxy sourceArg,
                        in MessagePortArray portsArg);
};
```

其實更簡單

- **WebSocket** 只會用到 **MessageEvent** 的 **.data** 屬性
- 所以只要這樣用事件處理函數來處理就可以：

```
var ws = new WebSocket('ws://host:port/resource','protocol');
ws.onmessage = function(e){
    alert(e.data);// 處理伺服器傳來的資料，放在 e.data 中
}
```

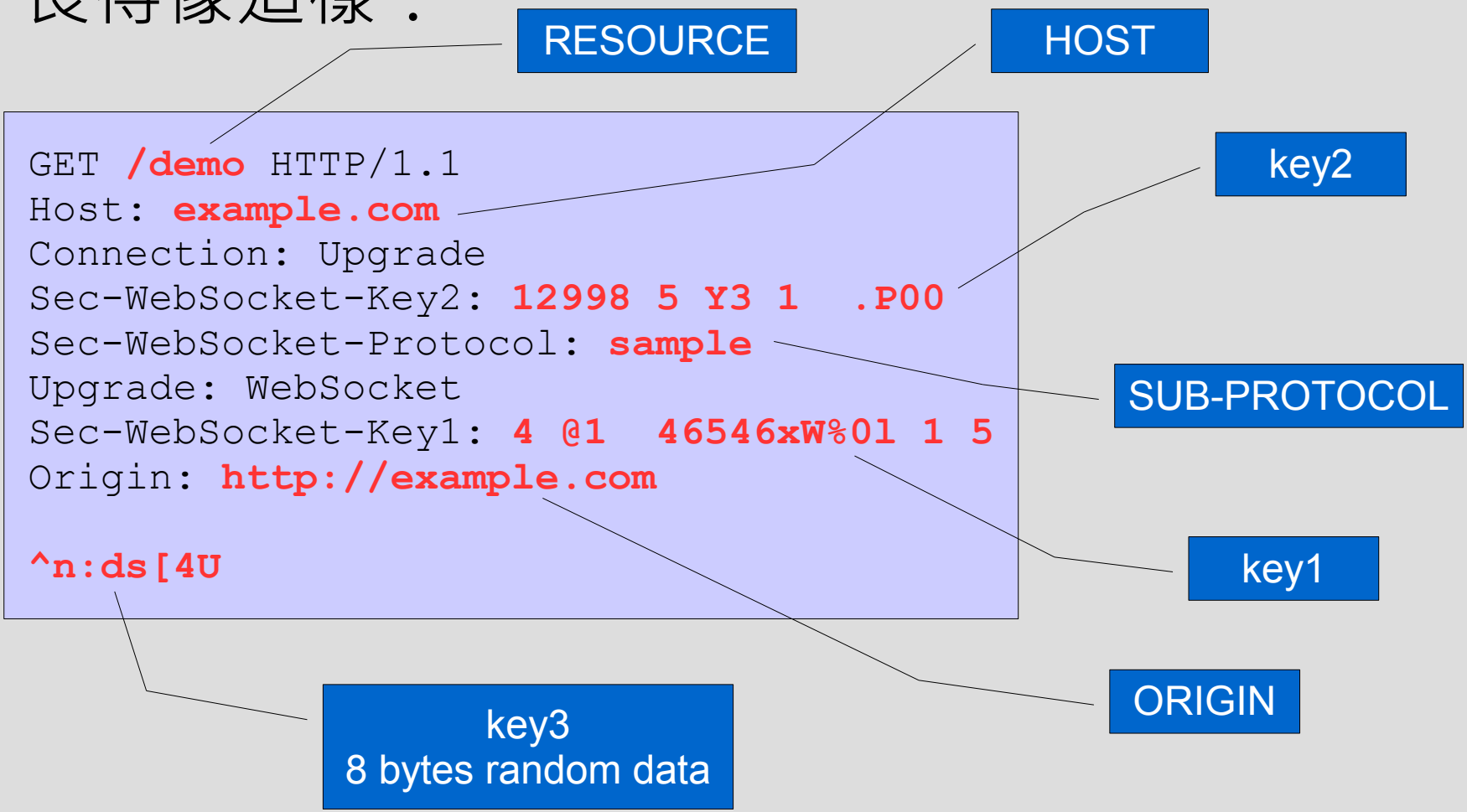
（如果玩過 WebWorker ，應該會覺得眼熟。我猜 html5/webapp 規格會盡量讓這些使用方式相近，使用會更直覺。另一個會眼熟的是使用在 cross-document messaging 上的 postMessage() 函數）

WebSocket Protocol

- 規格網址（目前版本是 draft 76 ，會在 2010-11-7 失效）
- Client / Server 之間，透過交握建立連線，之後就可以雙向傳遞資料
- draft 76 的交握方式與在 draft 76 之前的版本有很大的差異
- 資料傳輸透過 data frame 資料結構，這部份的規格還需要完善，目前只能傳送 utf-8 字串

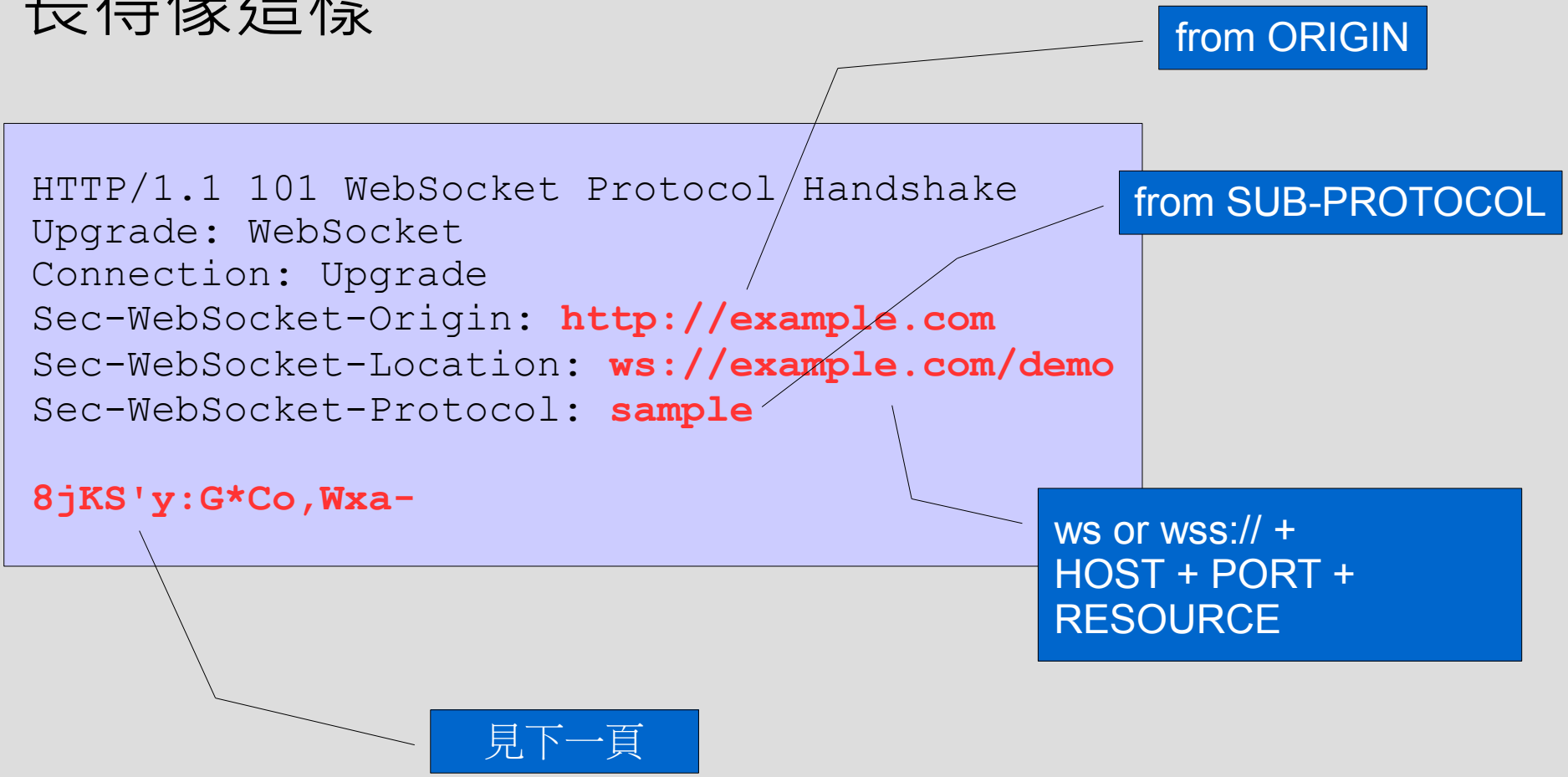
WebSocket 協定： client 送出交握

- 長得像這樣：



WebSocket 協定： server 送出交握

- 長得像這樣



WebSocket 協定交握中的挑戰 / 回應演算法

- 把 key1, key2 字串中的數字依序組成一個數字，除以字串中空白的個數，轉換成 32 位元無號整數 part1 及 part2 （沒有空白或是無法整除，就結束連線）
- 把 key1 key2 key3 組成長度 16 位元組的 Octet
- 用 md5 算出上述 Octet 的 hash ，長度也是 16 位元組
- 放在 response body 中回傳

WebSocket 協定： data frame

- 一般的 data frame
  - 1 byte 0x00
  - utf-8 編碼的資料
  - 1 byte 0xFF
- 結束通訊的 data frame （據說是這樣，但是瀏覽器收到都是出現錯誤）
  - 1 byte 0xFF
  - 1 byte 0x00

nodejs ？

- 結合 V8 Javascript 引擎及 event I/O 的伺服器端 Javascript 環境
- 支援 commonjs module 1.0 的規格，內建的模組與外掛都可以利用 require() 掛進來
- 官網： http://nodejs.org
- 目前 (2010-9-15) 最新版本是 0.2.1
- 我在這個範例中使用的是 0.2.0
- 0.1.x 的更板速度很快，介面變動也很大，希望 0.2.x 之後會穩定一些

nodejs 的網路通訊功能

- 直接看官網的簡單範例

```
var net = require('net');
var server = net.createServer(function (stream) {
  stream.setEncoding('utf8');
  stream.on('connect', function () {
    stream.write('hello\r\n');
  });
  stream.on('data', function (data) {
    stream.write(data);
  });
  stream.on('end', function () {
    stream.write('goodbye\r\n');
    stream.end();
  });
});
server.listen(8124, 'localhost');
```

net.Server

- net.Server 有兩個事件
  - connection
  - close
- net.Server 從 EventEmitter 繼承了 on 方法，可以用這個方法來指派事件處理函數
- 當 tcp 連線建立後， net.Server 會觸發 connection 事件，並且把 net.Stream 傳給事件處理函數
- createServer() 的參數，就是 connection 事件處理函數

net.Stream

- 他也繼承了 EventEmitter
- 最重要的是 data 事件，會在收到資料時觸發，讀取的資料會傳給事件處理函數的第一個參數，之後就可以用這個參數處理接收的資料
- 透過 write 方法，則可將資料寫入這個 stream
- 如果想要自己寫 Server 來處理連線，也可以自己利用 createConnection 來在聆聽指定的 port 與網址

處理 WebSocket 的陷阱

- 不要像前面範例這樣 stream.setEncoding('utf8')
- 因為 WebSocket 協定有用到 binary 的資料，把他轉成 utf8 後， stream.data 事件收到的資料也會變成 utf8 ，這樣會出問題
- nodejs 有內建 Buffer 物件，用來處理這類的資料

聊天室的構想

- 為求簡便，聊天室不儲存資料，只轉發收到的資料
- client 發出特定格式的 json 字串，送出資訊
- server 端在收到資訊時，送出更新資訊給所有的 net.Stream
- net.Server 並沒有所有連線的 stream 資料，所以必須另外維護，目前的構想是透過 stream.connect 與 stream.close 這兩個事件把 stream 加入 / 移出 server scope 中的陣列

demo 時間

- Live demo
  - 開三個瀏覽器來測試
  - 為了方便，寫了簡單的管理伺服器，用來關掉伺服器程式
- 程式碼說明
- 相關程式檔案及簡報，我會上傳到討論群組
  （這還有做很多修改才可能實用）

程式可以改進的地方

- 支援 wss
- 進一步做 refactoring
- 改成 commonjs module
- 支援多聊天室
  - 利用網址指定 or
  - 動態指定
- 支援資料儲存
- 測試 Cookie 支援
- 更多 ...（大概沒時間改，不過網路上已經有許多既有的 nodejs websocket 模組，也內建了 chat 機制，我只是拿來做 nodejs 的練習）

WebSocket 的展望

- 在 WebSocket API 應該不太會變動了
- 但是 WebSocket 協定距離完成還很遙遠， draft 76 會在 2010-11-7 過期，而下一個 draft 版本會在 2011-2-17 過期
- http://www.whatwg.org/specs/web-socket-protocol/
- 下一個版本在 data frame 會有大改動，這些改動可能是為了在未來支援更多種 data frame

nodejs 展望

- 未來會利用 WebWorker 的方式加入多工機制
- 其他就不知道了，不過用的人越來越多 ...
- 除了看官方文件，可以訂閱他的討論群組：
  - http://groups.google.com/group/nodejs?hl=en

寫 nodejs 的一點感想

- 需要多習慣「非同步」程式的寫法跟考量
- Javascript 的函數是 atomic 的 :D
- Closure 真好用

結束