https://goo.gl/CzXpph

Ethereum / Solidity Workshop

2017/02/17

Outline

- 加入以太坊 : Ethereum 環境建置
- 使用電子錢包 : Ethereum-Wallet
- 智能合約範例
- Solodity 程式架構/練習

加入以太坊 : Ethereum 環境建置

- Geth
  - Ethereum node
  - 使用Go語言實作Ethereum協議的程式
- 創世塊 (Genesis Block)
- 建立帳戶
- 建立私有鏈
- 進行挖礦
- 使用電子錢包
- 進行交易
- 部屬合約

創世塊 Genesis Block

genesis-block.json(點我下載)

```
{
  "nonce": "0x0000abcdef",
  "difficulty": "0x0",
  "mixhash": "0x00000000000000000000000000000000000000ffffffffffffffffffffffffff",
  "coinbase": "0x0000000000000000000000000000000000000000",
  "timestamp": "0x00",
  "parentHash":
"0x0000000000000000000000000000000000000000000000000000000000000000",
  "extraData": "0xffffff",
  "gasLimit": "0xfffffff"
}
```

加入私有鏈

static-nodes.js(點我下載)

```
admin.addPeer("enode://b9f46f29aa21c43e2b74ff81c7cd44f567ca08e74bfef78c2015919df161d961b53d388d79e02544e7cb4afbe8934c6686b99ed64ade9da9ca79207d3898ebb2@140.119.164.155:50505")
```

Geth環境建置---[Mac](Installing with Homebrew)

- Step 0.0 ─安裝Homebrew
  - ○ https://brew.sh/index_zh-tw.html
- Step 0.1 ─清空原有帳號資料(若曾經安裝過)
  - ○ rm -r ~/Library/Ethereum
  - ○ rm -r ~/.ethash
- Step 1 ─安裝Geth
  - ○ brew tap ethereum/ethereum
  - ○ brew install ethereum
- Step 2 ─取得 private chain 連線資訊
  - ○ 將genesis.json ,static-node.js與geth擺至同目錄下 -- 預設路徑為/usr/local/Cellar/ethereum/1.5.9/bin
  - ○ mv ~/Downloads/genesis-block.json /usr/local/Cellar/ethereum/1.5.9/bin
  - ○ mv ~/Downloads/static-nodes.js /usr/local/Cellar/ethereum/1.5.9/bin
- Step3 ─初始化
  - ○ cd /usr/local/Cellar/ethereum/1.5.9/bin
  - ○ geth init genesis-block.json
- Step 4─新增帳號
  - ○ geth account new
- Step 5 ─ 啟動
  - ○ geth --networkid 16888 --port 30303 --nodiscover --maxpeers 25 --nat "any" --rpc --rpccorsdomain "*" --rpcapi "eth,net,web3,debug" --targetgaslimit 8888888888 --preload static-nodes.js console
  - ○ 註 : --preload fileName.js表示 先執行該script後再啟動Geth

Geth環境建置---[Windows]

- Step 0.1 –清空原有帳號資料(若曾經安裝過)
  - 刪除 C:\Users\使用者\AppData\Roaming\Ethereum
  - 刪除 C:\Users\使用者\AppData\Ethash

- Step 1 –安裝Geth
  - 點我下載 備份連結64bit 備份連結32bit
- Step 2 –取得 private chain 連線資訊
  - 將genesis-block.json ,static-node.js與geth擺至同目錄下 (C:\Program Files\Geth)

- Step3 –初始化
  - cd C:\Program Files\Geth
  - geth init genesis-block.json

- Step4–新增帳號
  - geth account new
  -
- Step5 – 啟動
  - geth --networkid 16888 --port 30303 --nodiscover --maxpeers 25 --nat "any" --rpc --rpccorsdomain "*" --rpcapi "eth,net,web3,debug" --targetgaslimit 8888888888 --preload static-nodes.js console
  - 註 : --preload fileName.js表示 先執行該script後再啟動Geth
  -

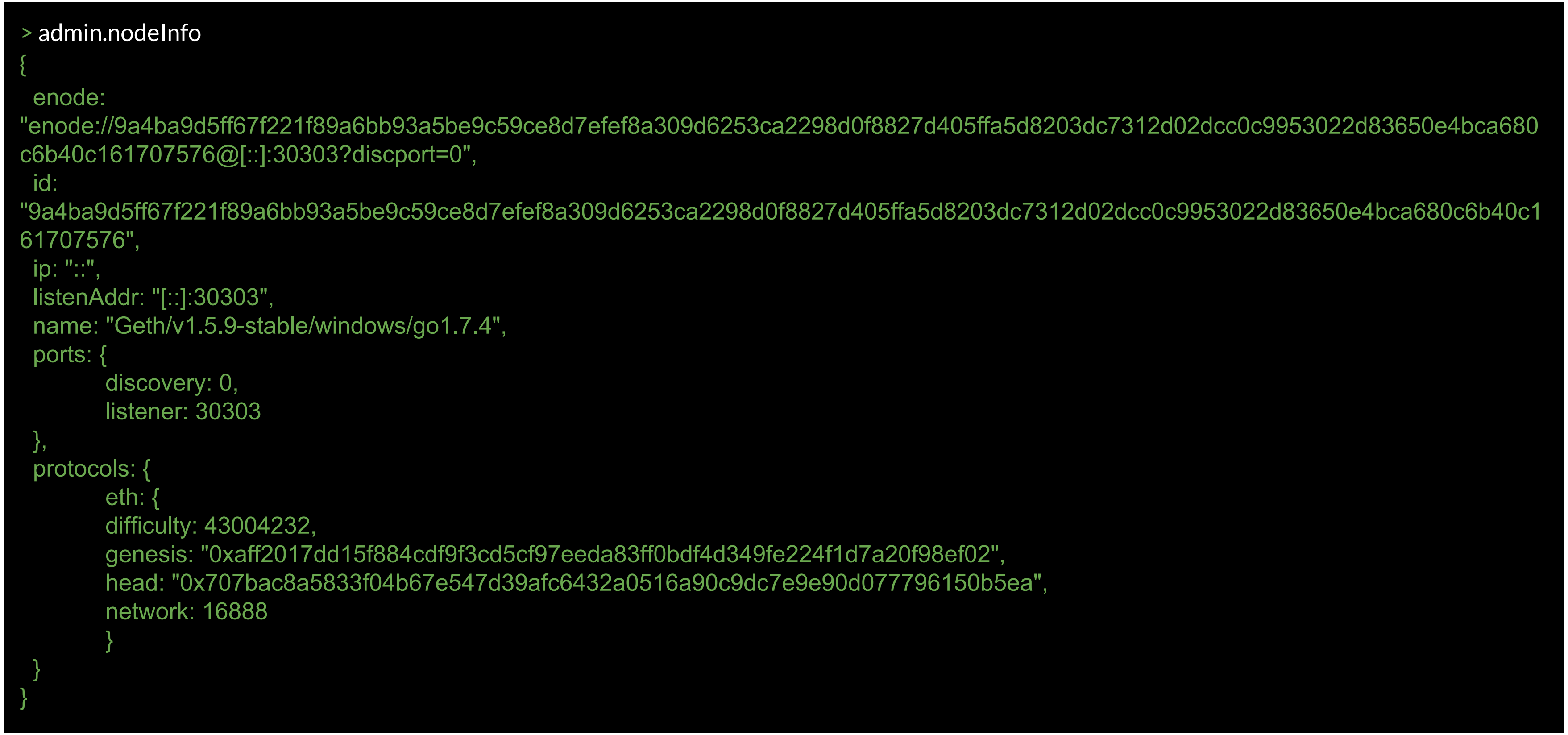
Geth---常用指令 [查詢,設定連線資訊]

- **進入本地端 JavaScript Console**
  - geth attach (connect to node) / geth console
- **加入 peer (於JavaScript Console)**
  - 法一 (於JavaScript Console) :
    - admin.addPeer("邀請者之Enode資訊") example
  - 法二 :
    - 加入js檔於geth目錄資料夾
    - 啟動Geth時指令加入 --preload static-nodes.js console (表示先執行該js檔再啟動Enode)
-

```
>admin.addPeer("enode://b9f46f29aa21c43e2b74ff81c7cd44f567ca08e74bfef78c2015919df
161d961b53d388d79e02544e7cb4afbe8934c6686b99ed64ade9da9ca79207d3898ebb2@14
0.119.164.155:50505")
>true
```

Geth---常用指令 [查詢,設定連線資訊]

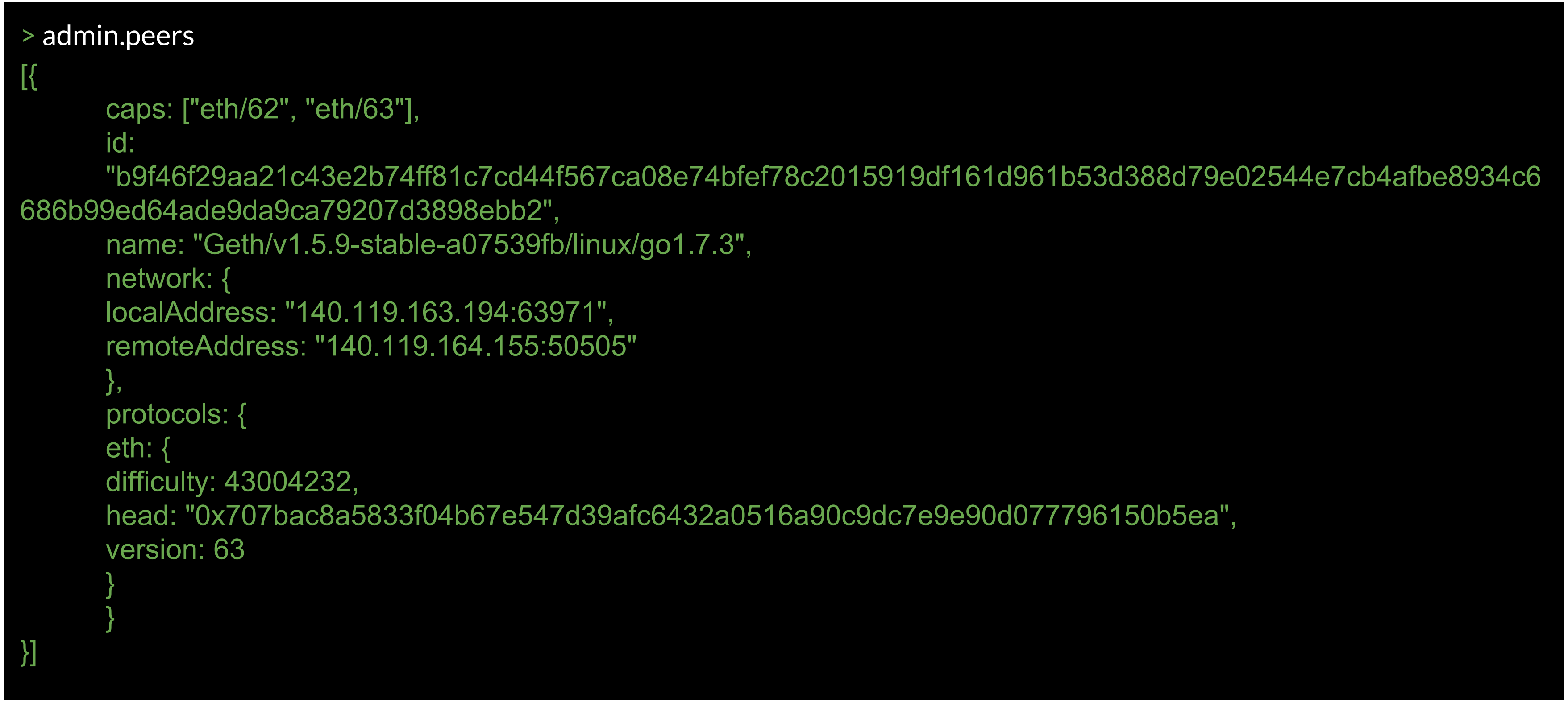
- 查詢本機Enode資訊



```
> admin.nodeInfo
{
  enode:
"enode://9a4ba9d5ff67f221f89a6bb93a5be9c59ce8d7efef8a309d6253ca2298d0f8827d405ffa5d8203dc7312d02dcc0c9953022d83650e4bca680
c6b40c161707576@[::]:30303?discport=0",
  id:
"9a4ba9d5ff67f221f89a6bb93a5be9c59ce8d7efef8a309d6253ca2298d0f8827d405ffa5d8203dc7312d02dcc0c9953022d83650e4bca680c6b40c1
61707576",
  ip: "::",
  listenAddr: "[::]:30303",
  name: "Geth/v1.5.9-stable/windows/go1.7.4",
  ports: {
            discovery: 0,
            listener: 30303
  },
  protocols: {
            eth: {
            difficulty: 43004232,
            genesis: "0xaff2017dd15f884cdf9f3cd5cf97eeda83ff0bdf4d349fe224f1d7a20f98ef02",
            head: "0x707bac8a5833f04b67e547d39afc6432a0516a90c9dc7e9e90d077796150b5ea",
            network: 16888
            }
  }
}
```

Geth---常用指令 [查詢,設定連線資訊]

- 查詢目前peer對象



```
> admin.peers
[{
        caps: ["eth/62", "eth/63"],
        id:
        "b9f46f29aa21c43e2b74ff81c7cd44f567ca08e74bfef78c2015919df161d961b53d388d79e02544e7cb4afbe8934c6
686b99ed64ade9da9ca79207d3898ebb2",
        name: "Geth/v1.5.9-stable-a07539fb/linux/go1.7.3",
        network: {
        localAddress: "140.119.163.194:63971",
        remoteAddress: "140.119.164.155:50505"
        },
        protocols: {
        eth: {
        difficulty: 43004232,
        head: "0x707bac8a5833f04b67e547d39afc6432a0516a90c9dc7e9e90d077796150b5ea",
        version: 63
        }
        }
}]
```

Geth---常用指令 [查詢,設定連線資訊]

- 查詢目前node中的accounts
  - >eth.accounts
- 新增帳號
  - >personal.newAccount("密碼")
- 挖礦
  - >miner.start(2)
    - 表示使用 2個thread mining
  - >miner.stop()

Geth---常用指令 (Web3.js Javascript API)

- 查詢帳號餘額
  - >eth.getBalance(YOUR_ACCOUNT_ADDRESS) [此時單位是 wei]
  - >web3.fromWei(eth.getBalance(YOUR_ACCOUNT_ADDRESS), "ether")
- 查詢當前挖礦預設帳號(coinbase)
  - >eth.coinbase
- 切換coinbase為第1個帳戶(預設為第0個)
  - **>miner.setEtherbase(eth.accounts[1])**
  -

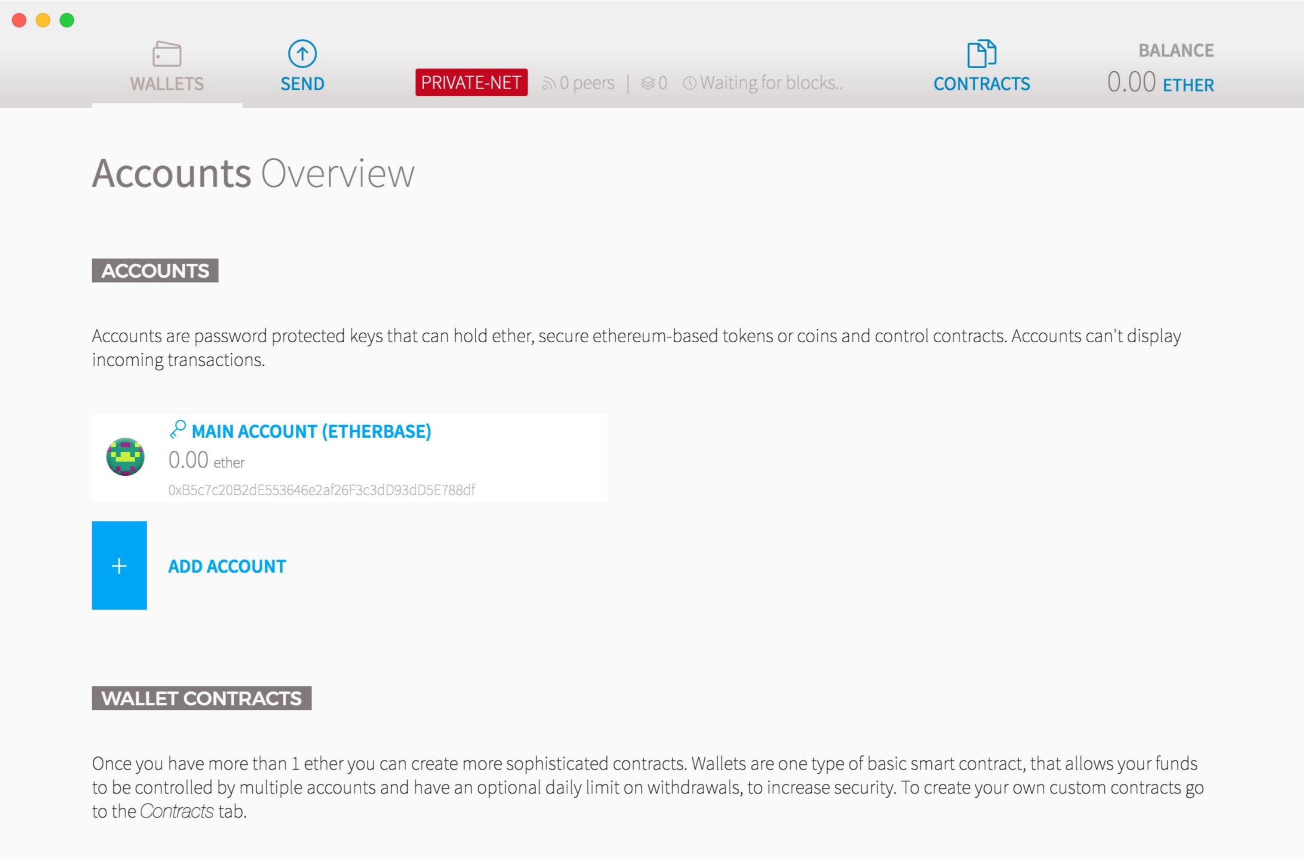
Geth---第一個 Transaction

- UnlockAccount
  - 指令
    - geth --unlock <YOUR_ACCOUNT_ADDRESS> --password <YOUR_PASSWORD>
  - JavaScript Console (API)
    - >personal.unlockAccount("address", "password")

- Send Transaction(需要先UnlockAccount)
  - Send ethCoin
    - 格式
      - >eth.sendTransaction({from:senderAddr, to:receiverAddr, value: amount})
    - 範例
      - >eth.sendTransaction({from:eth.coinbase, to:eth.accounts[1], value: web3.toWei(0.05, "ether")})

使用電子錢包：Ethereum Wallet

最新版本



Accounts Overview

ACCOUNTS

Accounts are password protected keys that can hold ether, secure ethereum-based tokens or coins and control contracts. Accounts can't display incoming transactions.

MAIN ACCOUNT (ETHERBASE)
0.00 ether
0xB5c7c20B2dE553646e2af26F3c3dD93dD5E788df

WALLET CONTRACTS

Once you have more than 1 ether you can create more sophisticated contracts. Wallets are one type of basic smart contract, that allows your funds to be controlled by multiple accounts and have an optional daily limit on withdrawals, to increase security. To create your own custom contracts go to the *Contracts* tab.

Solidity 基本介紹

- 乙太坊智能合約語言
- 風格類似JavaScript
- 強型別

Solidity 開發環境

- 線上IDE(推薦)
  - Browser-Solidity (via http, not https, for local enode connection)
- Solc編譯器
  - `npm install solc -g`
  - `solcjs --help`
  - `solcjs contract.sol`
- 文字編輯器套件
  - Atom : linter-solidity
  - Vim : vim-solidity
  - Code : vscode-solidity

First Contract

```
pragma solidity ^0.4.8;

contract myFirstContract {
  /* define variable */
  int myNumber;
  string myName;

  /* this runs when the contract is executed */
  function myFirstContract(string name) public {
    myName = name;
  }

  /*  function */
  function setNumber(int newNumber)  returns (bool) {
    myNumber=newNumber;
    return true;
  }

  function getMyName() constant returns (string) {
    return myName;
  }

  function getNumber() constant returns (int) {
    return myNumber;
  }
}
```

web3 deploey 請先解鎖 (personal.unlock("帳號","密碼"))

```
            var name = "Ruby";
            var untitled_myfirstcontractContract =
web3.eth.contract([{"constant":true,"inputs":[],"name":"getMyName","outputs":[{"name":"","type":"string"}]
,"payable":false,"type":"function"},{"constant":false,"inputs":[{"name":"newNumber","type":"int256"}],"nam
e":"setNumber","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},{"constant":true,"inp
uts":[],"name":"getNumber","outputs":[{"name":"","type":"int256"}],"payable":false,"type":"function"},{"inpu
ts":[{"name":"name","type":"string"}],"payable":false,"type":"constructor"}]);
            var untitled_myfirstcontract = untitled_myfirstcontractContract.new(
              name,
              {
                from: web3.eth.accounts[0],
                data:
'0x6060604052341561000c57fe5b60405161034a38038061034a833981016040528080518201919050
505b806001908051906020019061003f929190610047565b505b506100ec565b828054600181600116
1561010002031660029004906000526020600020906011f016020900481019282601f10610088578
05160ff1916838001178555610b6565b82800160010185558215610b6579182015b828111156100b
```

Contract ↣ watch contract

Address: 部署成功後的address

Interface:

```
[{"constant":true,"inputs":[],"name":"getMyName","outputs":[{"name":"","type":"string"}],"payable":false,"type":"function"},{"constant":false,"inputs":[{"name":"newNumber","type":"int256"}],"name":"setNumber","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},{"constant":true,"inputs":[],"name":"getNumber","outputs":[{"name":"","type":"int256"}],"payable":false,"type":"function"},{"inputs":[{"name":"name","type":"string"}],"payable":false,"type":"constructor"}]
```

Solidity Layout

- Version Pragma
    - `pragma solidity ^0.4.10;`
- Comments

    -
      ```
      // This is a single-line comment.

      /*
      This is a
      multi-line comment.
      */
      ```

Solidity 基本架構(1/4)

- State Variables(狀態變數)
  - 儲存狀態

    -
      ```
      pragma solidity ^0.4.0;

      contract SimpleStorage {
          uint storedData; // State variable
          // ...
      }
      ```

Solidity 基本架構(2/4)

- Functions(函式)
  - 改變狀態、與其他合約互動、建構子

    -
      ```
      pragma solidity ^0.4.0;

      contract SimpleAuction {
          function bid() payable { // Function
              // ...
          }
      }
      ```

Solidity 基本架構(3/4)

- Function Modifiers(函式描述子)
  - 前置條件(前提)

```
pragma solidity ^0.4.0;
contract modifierSample {
    address public Owner;
    modifier onlyOwner() {  // Modifier
        if (msg.sender != Owner) throw;
        _;
    }
    function modifierSample(){Owner=msg.sender;}
    function close() onlyOwner {  // Modifier usage
        selfdestruct(Owner); // ...
    }
}
```

Solidity 基本架構(4/4)

- Events(事件)
  - 與鏈外事件互動

    -
      ```
      pragma solidity ^0.4.0;

      contract eventSample {
          event payEvent(address from,address to,uint amount);  //
      Event

          function payOthers(address targetAddr,uint amount)  payable {
          targetAddr.send(this.balance);
          payEvent(this, targetAddr,this.balance);  // Triggering event
              }
          }
      ```

Solidity 實值型別(1/2)

- Booleans
    - `true` and `false`
    - Operators
        - `!` logical negation
        - `&&` logical conjunction, “and”
        - `||` logical disjunction, “or”
        - `==` equality
        - `!=` inequality
- Integers
    - `int`
    - `uint (unsigned integer)`

- address
    - 20 byte value (size of an Ethereum address).
- *`string`*
    - *Dynamically-sized UTF-8-encoded string.*
- 常數修飾詞
    - `constant`

Solidity 實值型別(2/2)

- uint x;
- int constant a = 8;
    - *int256 contant a = 8; // 與上行效果一樣*
- bool b = true;
    - *var b = true; // 與上行效果一樣*
    - *var 可用於型別推論*
- address public owner;
    - *帳戶有兩種*
        - *合約帳戶(Contract Account)*
        - *外部帳戶(External Account)*
    - *合約帳戶指合約，外部帳戶指外部個體帳戶*
- owner.balance;
    - *得到 owner 的帳簿餘額(無論此為合約帳戶或外部帳戶)*

練習(請使用First Contract)

- 練習1.
  - 請加入myOwner變數以及修改FirstContract建構子,
  - 當合約部署時將部署者之名稱記錄至 myName及Address記錄至myOwner
    - function Name:FirstContract(string name,_____ Owner)
    - parameters: name,Owner
- 練習2.
  - 回傳Owner之address，並判斷是否可加入 ”constant“,若否則不用加入。
    - function Name: getOwner()
    - parameters: none
    - return s: address 。
- 練習3.
  - 請加入event ,並使用wallet觀察contract所發出event
  - 當setNumber被呼叫時則發出 event告知外部(1)myNumber(2)newNumber(2)由呼叫此function
    - event name: numberChangeEvent
    - parameter: oldNumber,newNumber,from

Solidity 參考型別(1/2)

- Arrays
  - `T[]       t;` // an array of dynamic size
  - `T[N]      t;` // N-length array
  - `T[N][M]   t;` // M*N array

Solidity 參考型別(2/2)

- Structs

  -
    ```
    pragma solidity ^0.4.0;
    contract Test {
        struct Student { // Struct
            uint id;
            string name;
        }

        Student A = Student({
            id = 1234,
            name = ”John”;
        });
        //or
        Student B = Student(1234, ”John”);
    }
    ```

Solidity 其他型別

- 型別轉換
  - 隱含轉換
    - `uint8 -> uint16 (O)`
    - `int8 -> int16 (O)`
    - `int8 -> uint8 (X)`
  - 明確轉換
    - `int8 y = -3; uint x = uint(y);  // x is now 0xfffff..fd`
    - `uint32 a = 0x12345678; uint16 b = uint16(a);  // b is now 0x5678`

Mappings

- `mapping(_KeyType => _ValueType)`
    - `_KeyType` can be almost any type except for a mapping
    - `_ValueType` can actually be any type, including mappings

    -
      ```
      mapping (string => uint) public balances;
      balances["John"] = 10 ether;
      ```

Solidity 全域變數

- block
  - `block.blockhash(uint blockNumber) returns (bytes32)`
  - `block.coinbase   // miner's address`
  - `block.gaslimit   // current block gaslimit`
  - `block.number     // current block number`
  - `block.timestamp // current block time (before commit)`
- msg
  - `msg.sender   //sender of the message (current call)`
  - `msg.value    //number of wei sent with the message`
- now
  - alias for `block.timestamp`
- tx
  - `tx.origin    //sender of the transaction (full call chain)`

Solidity 流程控制(1/2)

- Control Structure
    - 邏輯
        - if
        - else
        - ?:
    - 迴圈
        - while
        - do
        - for
        - break
        - continue
    - 函式
        - return
        - return *VALUE*,..;
        - function *NAME*(*ARG*,..) *VISIBILITY* returns (*TYPE*,...) {}

Solidity 流程控制(2/2)

- Assignment

  -
    ```
    function f() returns (uint, bool, uint) {
            return (7, true, 2);
    }
    function g() {
        var (x, b, y) = f(); // x = 7, b = true, y = 2
        (x, y) = (2, 7); // x = 2, y = 7
        (x, y) = (y, x); // x = 7, y = 2
    }
    ```
- Exception
  - `throw`
  - 目前 solidity 暫時無法處理例外

Solidity 單位

- Ether Units
  - 1 = 1 wei
  - 1 ether = 1000 finny
  - 1 ether = 1000000000000000000 wei = 10^{18} wei
- Time Units
  - 1 == 1 seconds
  - 1 minutes == 60 seconds
  - 1 hours == 60 minutes
  - 1 days == 24 hours
  - 1 weeks = 7 days
  - 1 years = 365 days

Example 1

```
pragma solidity ^0.4.8
contract ReferenceTypeTest {
    struct Student {
        uint id;
        string name;
        uint age;
    }

    mapping (uint => Student) students;

    function putStudent(uint _id, string _name, uint _age) {
        students[_id] = Student({
            id: _id,
            name: _name,
            age: _age
        });
    }
    function getStudent(uint _id) returns (uint id, string name, uint age) {
        var student = students[_id];
        id = student.id;
        name = student.name;
        age = student.age;
    }
}
```

Example 2

```
pragma solidity ^0.4.8
contract Test{
        struct coinWallet {
                uint redCoin;
                uint greenCoin;
                string userName;
        }
        coinWallet public myWallet;
        mapping (address => coinWallet) public allWallets;

        function Test(){
                myWallet.redCoin = 500;
                myWallet.greenCoin = 250;
                myWallet.userName = "me";

                allWallets[tx.origin] = myWallet;
        }
        function GetRed() returns (uint redAmount){
                return myWallet.redCoin;
        }
        function GetTotal() returns (uint totalAmount){
                return myWallet.redCoin + myWallet.greenCoin;
        }
}
```

練習

- 請在右邊程式碼加入 function
- newUser()
  - 新增使用者
- getUserInformation()
  - 查詢使用者資訊

```
pragma solidity ^0.4.5;

contract phoneSample{
struct user{
   string userName;
   bool oftenToUse;
   uint monthlyFees;
}

mapping (address => user)  users;

function phoneSample(string myName,bool isOftenToUse,uint myMonthlyFees){
   users[msg.sender].userName = myName;
   users[msg.sender].oftenToUse = isOftenToUse;
   users[msg.sender].monthlyFees = myMonthlyFees;

}


function getMyUserInformation() constant returns (string myName,bool
isOftenToUse,uint myMonthlyFees){
   myName = users[msg.sender].userName;
   isOftenToUse = users[msg.sender].oftenToUse;
   myMonthlyFees = users[msg.sender].monthlyFees;
}


}
```