Smart Contract

- 陳博宇 @ 東吳大學 -

關於我

- 資工系 @ 交通大學
- 學術長 @ 清華大學區塊鏈研究社
- 核心成員 @ 台灣密碼龐克
- 區塊鏈工程師 @ 密碼貨幣交易所

目錄

```
pragma solidity ^0.6.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
}
```

- 什麼是智能合約？ -

區塊鏈上的智能合約

- Bitcoin
  - 支援條件判斷
  - 不支援迴圈
  - 不保存狀態
  - 無法做複雜的計算設計
- Ethereum
  - 支援條件判斷
  - 支援迴圈
  - 狀態儲存
  - 可模擬任何程式執行
  - 如何避免無窮迴圈？

以太坊狀態儲存

- EOA (External Owned Account)
  - Address 20 Bytes
  - Balacne
  - Nonce

- 合約帳戶 (Contract Account)
  - Address 20 Bytes
  - Balacne
  - Nonce
  - Storage (empty by default)
  - Code

Address + Nonce → RLP-encode → SHA3 → 後40

Gas的設計

- 每種運算都有其相對應的成本
- Gas Price
  - 每個單位 Gas 的價格
  - 1 Gwei = 0.000000001 ETH
- Gas Limit
  - 單筆交易中所願意支付 Gas 單位的最大數量
- Tx Fee
  - 最多為 Gas Limit * Gas Price

1. 礦工的選擇
2. 超鉅額手續費

合約的部署

1. 寫好合約
2. 編譯合約
   - Bytecode
   - ABI
3. 透過線上 IDE 部署 or 其他

呼叫合約

1. Function 的識別碼
2. 放上需要的參數

Function : setName(string)

↓

Kecaak-256

↓

c47f0027.....

優點

1. 提供可信任應用
2. 流程自動化
3. 運行成本降低

缺點

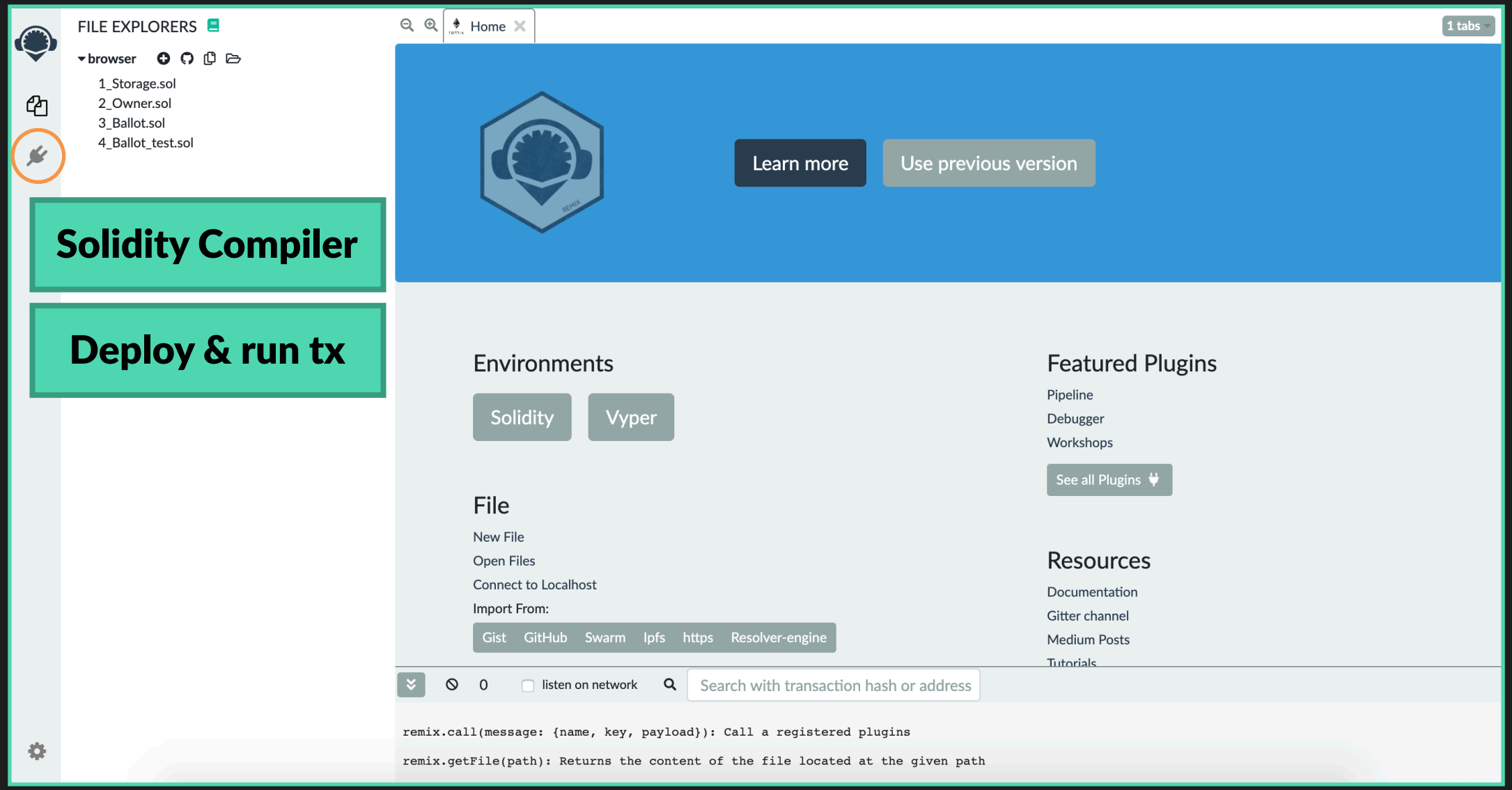
1. 安全性議題
2. 交易處理速度
3. 不可篡改的延伸問題

- 開發環境介紹 -

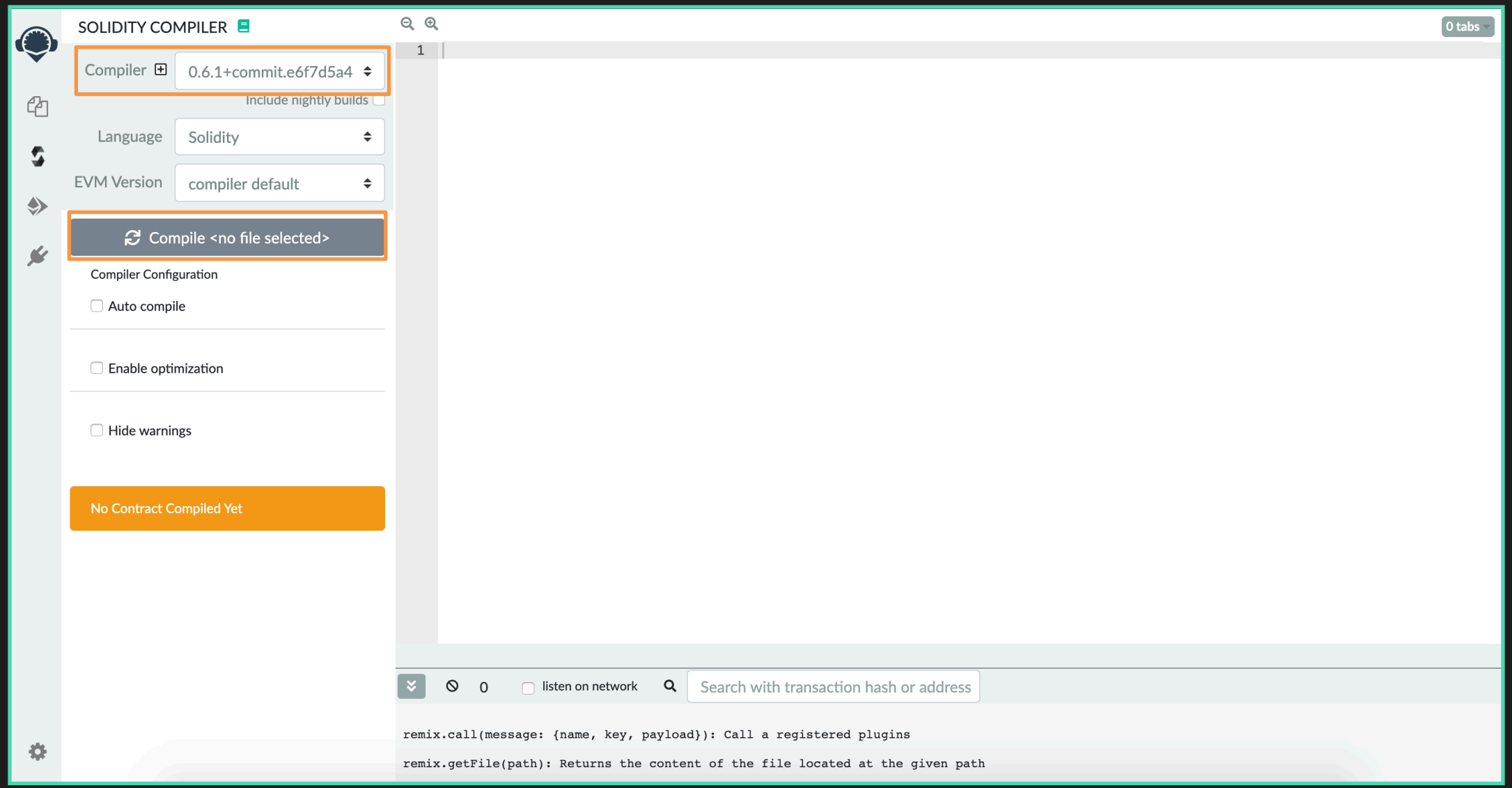
編譯合約

- Remix - 線上 IDE
  - http://remix.ethereum.org
- 安裝本機版 Remix IDE
  - `npm install remix-ide -g`
- Solc (Solidity Compiler) - localhost
  - https://www.npmjs.com/package/solc

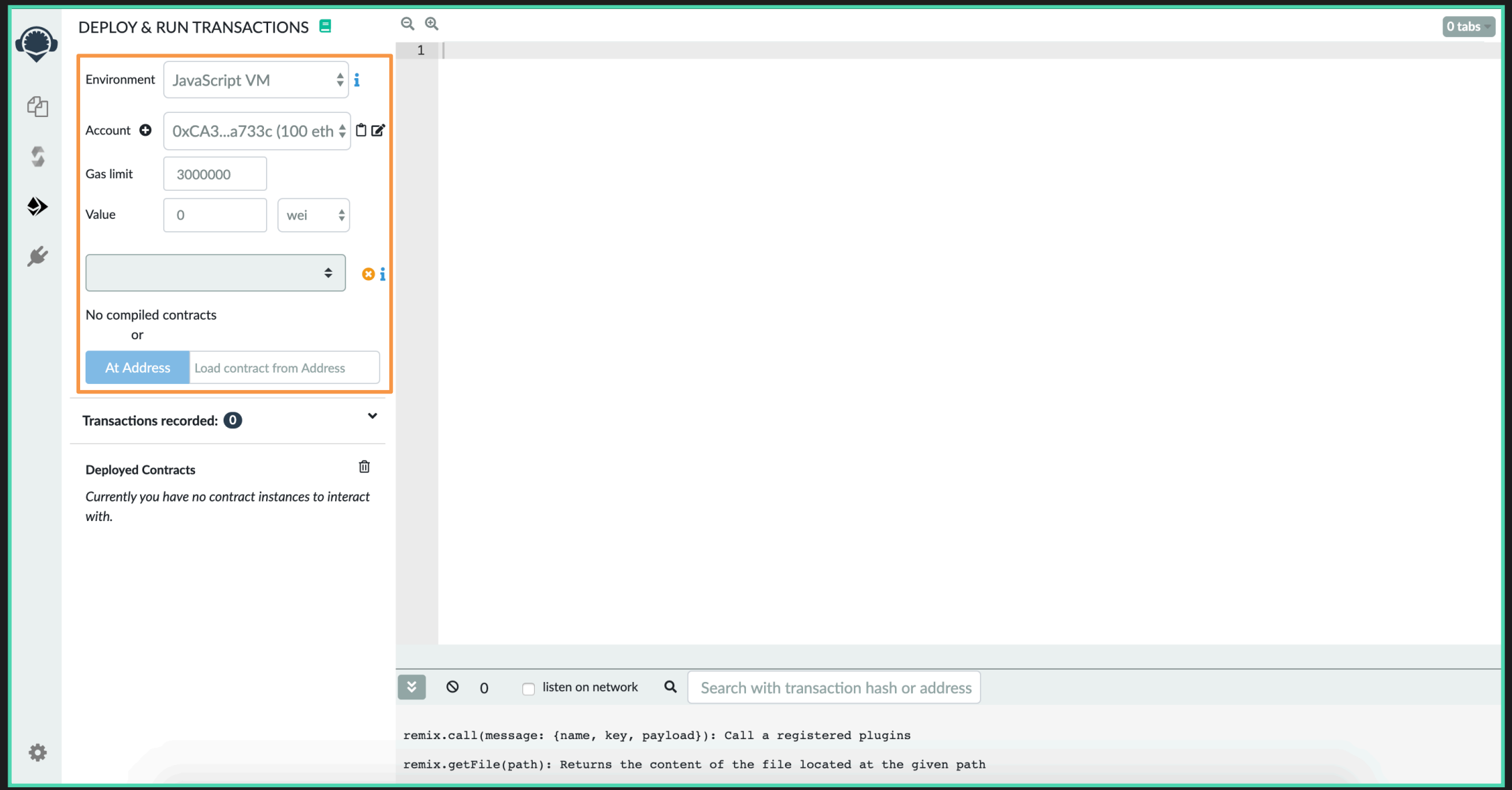
Remix - 線上 IDE


FILE EXPLORERS
browser
1_Storage.sol
2_Owner.sol
3_Ballot.sol
4_Ballot_test.sol
Solidity Compiler
Deploy & run tx
Home
1 tabs
Learn more
Use previous version
Environments
Solidity
Vyper
File
New File
Open Files
Connect to Localhost
Import From:
Gist
GitHub
Swarm
Ipfs
https
Resolver-engine
Featured Plugins
Pipeline
Debugger
Workshops
See all Plugins
Resources
Documentation
Gitter channel
Medium Posts
Tutorials
0
listen on network
Search with transaction hash or address
remix.call(message: {name, key, payload}): Call a registered plugins
remix.getFile(path): Returns the content of the file located at the given path

Remix - 線上 IDE


SOLIDITY COMPILER
Compiler
0.6.1+commit.e6f7d5a4
Include nightly builds
Language
Solidity
EVM Version
compiler default
Compile <no file selected>
Compiler Configuration
Auto compile
Enable optimization
Hide warnings
No Contract Compiled Yet
0 tabs
1
0
listen on network
Search with transaction hash or address
remix.call(message: {name, key, payload}): Call a registered plugins
remix.getFile(path): Returns the content of the file located at the given path

Remix - 線上 IDE


DEPLOY & RUN TRANSACTIONS
Environment
JavaScript VM
Account
0xCA3...a733c (100 eth
Gas limit
3000000
Value
0
wei
No compiled contracts
or
At Address
Load contract from Address
Transactions recorded: 0
Deployed Contracts
Currently you have no contract instances to interact with.
0 tabs
1
0
listen on network
Search with transaction hash or address
remix.call(message: {name, key, payload}): Call a registered plugins
remix.getFile(path): Returns the content of the file located at the given path

- 智能合約架構簡介 -

基礎架構

- 官方文檔：https://solidity.readthedocs.io/en/v0.6.0/index.html

```
pragma solidity ^0.6.0;

contract SimpleStorage {
    uint256 storedData;

    function set(uint256 x) public {
        storedData = x;
    }

    function get() public view returns (uint256) {
        return storedData;
    }
}
```

變數宣告

很多函數

很多函數

變數宣告

- 變數型態
  - bool
  - int / uint
  - bytes
  - address
  - string
  - array
  - mapping
- \+
- 能見度
  - public
  - private
  - internal
- \+
- 變數名稱

```
int8 public age;
bool private isOwner;
string name;
```

變數宣告

- address

```
address payable public bank;
```

- mapping

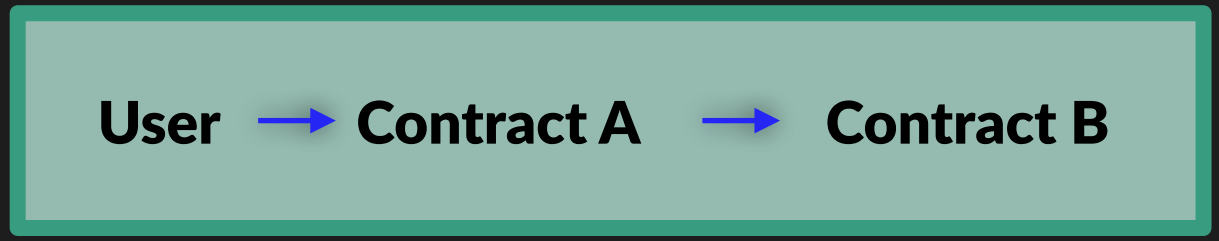
```
mapping(address => uint256) public balances;
balances[address] = 10;
uint256 balance = balances[address];
```

- array
  - push
  - pop
  - length

```
uint256[4] fixArr;
uint256[] dynamicArr;
```

特殊變數

- Coin
  - wei
  - gwei
  - finney
  - ether
- Time
  - now
  - seconds
  - minutes
  - hours
  - days
  - weeks
  - years
- Tx
  - tx.orgin
  - tx.gasPrice
- msg
  - msg.sender
  - msg.value
  - msg.data



特殊變數

- Address
  - address.balance
  - address.tranfser
  - address.send
  - address.call
- Block
  - block.number
  - block.timestamp
  - block.difficulty
  - blockhash (uint)

函數宣告

Storage ⟷ Memory ⟷ Calldata

- 函數名稱(參數) + 能見度 + 回傳值
  - public
  - private
  - internal
  - external ← this.funtion( )

```
function funName() private {…}
function funName2(uint num) external returns(uint8) {…}
function deposit() public payable {…}
```

函數宣告

- View function
- Pure function
  - 不改變合約狀態
  - 函數執行不消耗 gas
  - 不需經過礦工驗證

```
function viewFun(uint256 a, uint256 b) public view returns (uint256) {
    return a * (b + 42) + now;
}

function pureFun(uint256 a, uint256 b) public pure returns (uint256) {
    return a * (b + 42);
}
```

Error Handling

- Assert
  - 燒掉所有 gas
  - 常用於處理非變量
  - 常用於處理溢位
  - 驗證改變後的狀態
  - 一般用於函數結尾
- Require
  - 退回剩餘 gas
  - 常用於驗證 input
  - 常用於驗證條件狀態
  - 一般用於函數開頭
  - 允許 error message
- Revert
  - 退回剩餘 gas
  - 搭配 if / else
  - 允許 error message

特殊函數

revert →

```
function buy(uint amount) public payable {
    if (amount > msg.value / 2 ether)
        revert("Not enough Ether provided.");
}
```

require、assert

↓

```
function sendHalf(address payable addr) public payable returns (uint256 balance) {
    require(msg.value % 2 == 0, "Even value required.");
    uint256 balanceBeforeTransfer = address(this).balance;
    addr.transfer(msg.value / 2);
    assert(address(this).balance == balanceBeforeTransfer - msg.value / 2);
    return address(this).balance;
}
```

特殊函數

- Constructor
  - 合約建構子
  - 只會執行一次
  - 非必須
- Selfdestruct
  - 合約自殺
  - 唯一參數為地址
  - 把合約剩餘的錢給該地址

```
contract shop {
    address payable owner;

    constructor() public {
        owner = msg.sender;
    }

    function close() public {
        require(owner == msg.sender);
        selfdestruct(owner);
    }

}
```

特殊函數

- Fallback / Receive [payable]
  - 沒有 function 宣告
  - 沒有參數與回傳值
  - 必須是 external
  - 預設只有 2300 gas
  - 非必要
  - 觸發條件：
    1. 單純的轉帳
    2. 呼叫合約沒有的函數

```
contract StandardFallback {

    receive() external payable {}

    fallback() external {}
}
```

- 撰寫第一個智能合約 -

練習1

- 理解 external 和 public 的實際差異
  - 變數宣告：
    - `mapping (字串 → 地址) public students;`
  - 函數宣告：
    - `function publicFun(memory 字串, 地址) public {…}`
    - `function externalFun(calldata 字串, 地址) external {…}`
    - `function callPublicFun(calldata 字串, 地址) external {…}`
    - `function callExternalFun(calldata 字串, 地址) external{…}`

練習2

- 理解 array 操作 with view
  - 變數宣告：
    - `address[] students;`
  - 函數宣告：
    - `function addStudent(地址) {…}`
    - `function deleteStudent(地址) {…}`
    - `function getStudentLen() view returns(長度){…}`

練習3

- 理解 Constructor 和 Fallback 函數 with msg
  - 變數宣告：
    - `address public payable owner;`
  - 函數宣告：
    - `constructor () {owner = sender}`
    - `fallback () {如果觸發者為 owner 則自殺並且把錢轉給 owner}`
    - `receive () {只要觸發就把錢轉給特定地址}`

練習4

- 在合約中呼叫其他合約
  - 合約1：
    - `function sqr(數字) {回傳平方值};`
    - `function mul(數字1,數字2) {回傳相乘值};`
  - 合約2：
    - `合約1 名稱 = new 合約1();`
    - `function callSqr(數字) {呼叫合約1};`
    - `function callMul(數字1,數字2) {呼叫合約1};`

- Demo Time -

- 作業公布 -

作業公布

- 實作一個銀行合約
  - 函數功能要有：
    - 存錢
    - 提錢
    - 轉帳
    - 查詢餘額
    - 查詢銀行餘額
    - 帳戶註冊
  - 要有 Fallback (捲款潛逃)
  - 要有 Constructor
  - 基本防呆（ with error message ）
  - 使用 mapping 存取帳本

函數詳細內容

- Constructor → 設定合約擁有者
- 存錢 → `function deposit()`
- 提錢 → `function withdraw(uint withdrawAmount)`
- 轉帳 → `function tnasfer(uint transerAmount)`
- 餘額查詢 → `function getBalance()`
- 銀行資產查詢 → `function getBankBalance()`
- 帳戶註冊 → `function enroll(string accountName)`
- Fallback → 確認是 Owner 卷款錢逃
- Constructor → 設定 Owner
- 防呆說明 → Error message

- END -