



# 이더리움 스마트 계약의 이해와 실습

서강대학교  
서강미래기술원  
글로벌 핀테크 연구소

고덕윤  
maniara.k@gmail.com

- 
1. 블록체인의 개요
  2. 블록체인의 의미
  3. 블록체인 비즈니스 사례
  4. 블록체인의 비전
  5. 블록체인의 도전

- 본 강의의 이해를 위해선 어느 정도의 블록체인 지식이 필요합니다.
- 실습은 주로 프로그래밍 코드로 구성됩니다. 따라서 어느 정도의 컴퓨터 공학 지식이 필요합니다.
- 이미지 및 내용의 라이선스를 고려하지 않았으니, 외부 유출을 금합니다.

---

## 01 스마트 계약이란?

위키피디아에 의하면,

- 특정 계약을 스스로 수립, 검증, 이행 하기 위한 컴퓨터 프로토콜
- 계약의 보안을 높이기 위한 방안
- 계약 비용을 감소하기 위한 방안
- 1994년 Nick Szabo 가 처음으로 smart contract 라는 단어를 사용
- Nick said Smart contract is for:
  - minimize malicious, accidental exceptions
  - minimize the need for trusted intermediaries
  - lowering fraud loss,
  - lower transaction, arbitrations and enforcement cost

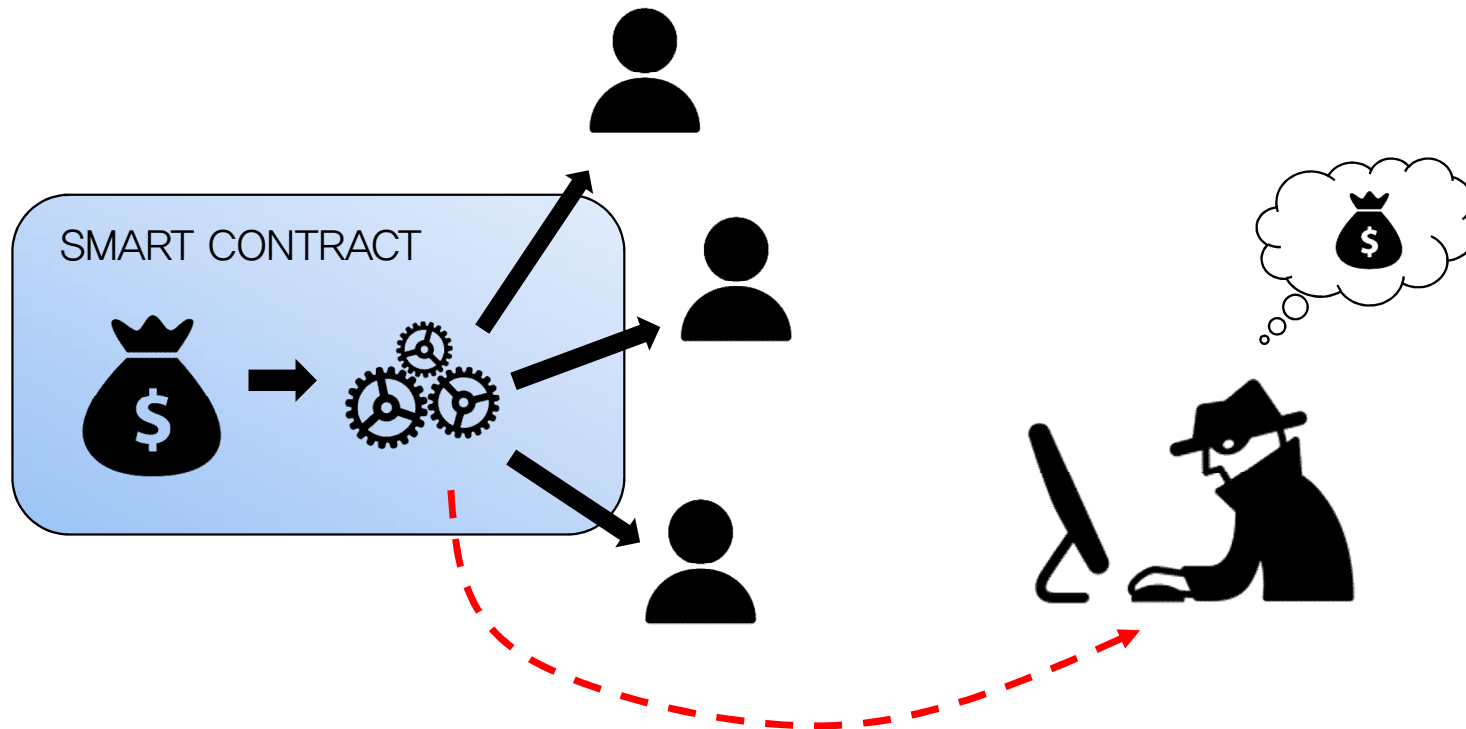


<https://www.youtube.com/watch?v=vXCOfTteQAo>

## 02

### 스마트 계약의 보안성

주로 돈을 다루기 때문에 해커의 공격 대상이 되기 쉬움



## 03 스마트 계약, 블록체인을 만나다.

비탈릭 부테린의 인터뷰 중



비탈릭 부테린 (당시 21세, 러시아)

블록체인에 스마트 계약 데이터를 기록하겠다.

- 계약 기록
- 계약 내용
- 계약 코드
- 이행 이력
- 이행 결과

해킹 하려면 해보시지!

○○

---

## 04 이더리움 스마트 계약 배포 현황

놀랄만한 성과

---



이미지 출처 : karl.tech

- 2016년 9월 8일 기준
  - 121,441 개의 스마트 계약 배포
  - 8,515,496 ETH 거래 중
  - 약 1100억원 규모
  - 배포 DApp 현황  
<http://dapps.ethercasts.com/>

## 05

# 이더리움의 탈중앙 애플리케이션 (DApp)

스마트 계약보다 한 단계 진화한 DApp



스마트 계약을 넘어 탈중앙 애플리케이션 플랫폼

- 지원 언어
  - Solidity (main) : Javascript와 유사
  - Serpent : C와 유사
  - LLL : Assembly와 유사
  - Mutal : 잘 쓰이지 않음
- 튜링 완전 언어
  - 변수/상수
  - 구조체
  - 함수
  - 반복문/분기문

## 06 블록체인 기반의 스마트 계약

이더리움 DApp의 생성, 실행 과정

### 스마트 계약 생성



스마트 계약 코드



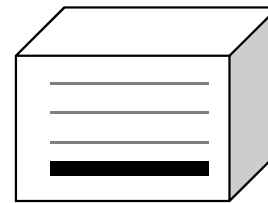
컴파일



바이트 코드



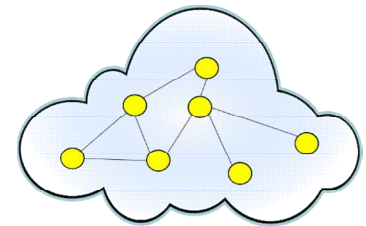
트랜잭션



블록에 삽입



배포



블록체인에 저장

### 스마트 계약 실행



사용자 주소

+



함수 주소

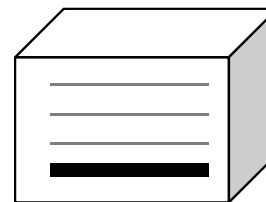
+



매개 변수



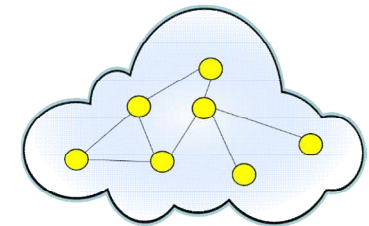
트랜잭션



블록에 삽입



실행



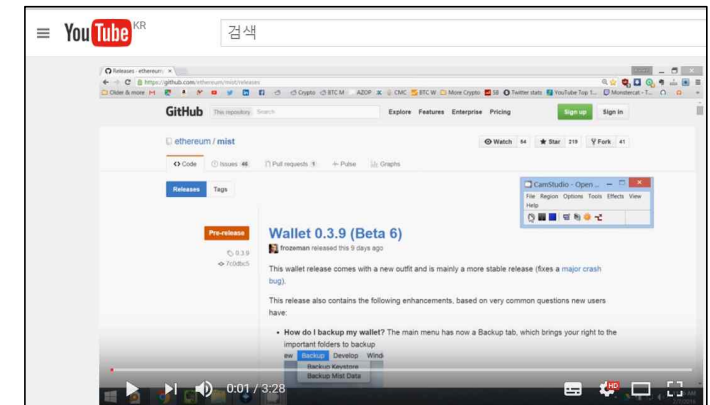
블록체인에  
실행 결과 저장



## 07 이더리움 지갑 설치

- 설치 URL : <https://github.com/ethereum/mist/releases>
- 설치 가이드 동영상 : <https://www.youtube.com/watch?v=Z6IE0Ctaeqs>

Downloads	
 <a href="#">Ethereum-Wallet-linux32-0-8-2.zip</a>	56.9 MB
 <a href="#">Ethereum-Wallet-linux64-0-8-2.zip</a>	57 MB
 <a href="#">Ethereum-Wallet-macosx-0-8-2.zip</a>	64.4 MB
 <a href="#">Ethereum-Wallet-win32-0-8-2.zip</a>	56.1 MB
 <a href="#">Ethereum-Wallet-win64-0-8-2.zip</a>	81.6 MB



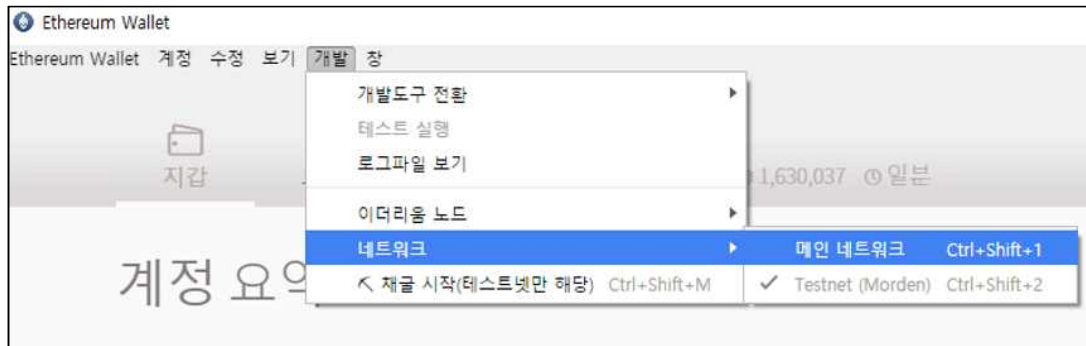
- 지갑의 설치와 동시에 블록을 다운받음

6 피어(peers) | 991,952 블록 남음 | 3%

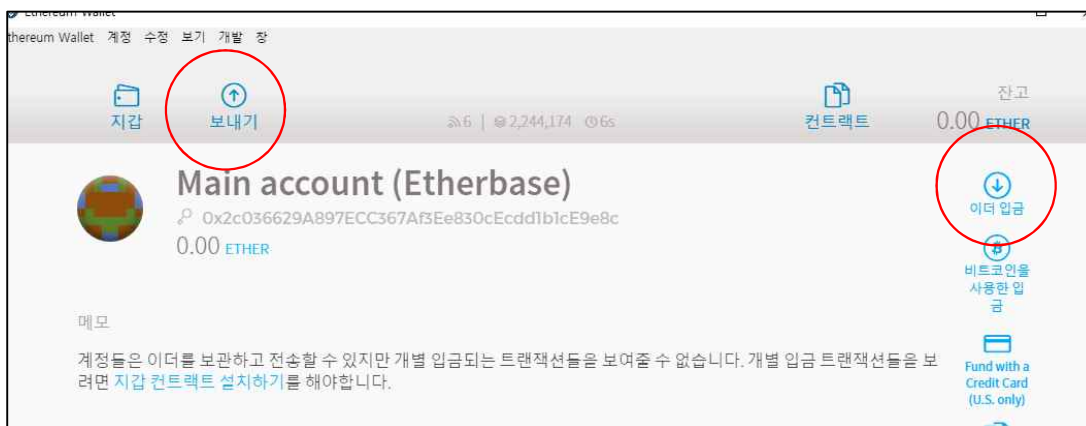
3 피어(peers) | 2,244,635 2분 마지막 블록 이후

## 08 이더리움으로 돈 주고 받기

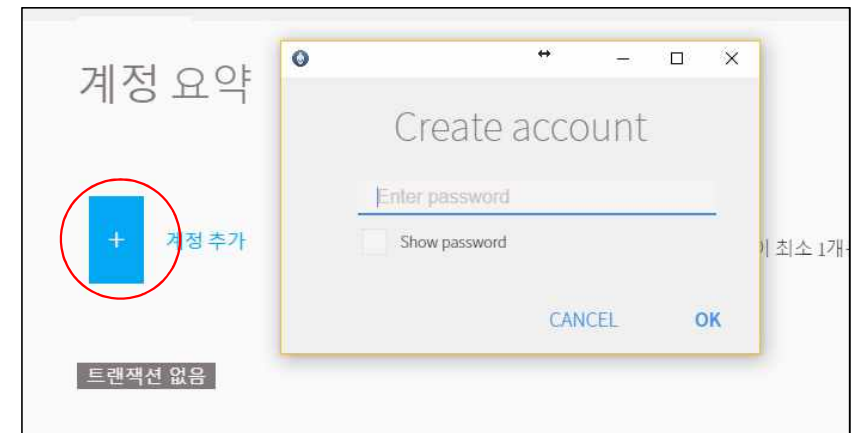
- 메인 네트워크로 전환



- Ether 주고 받기

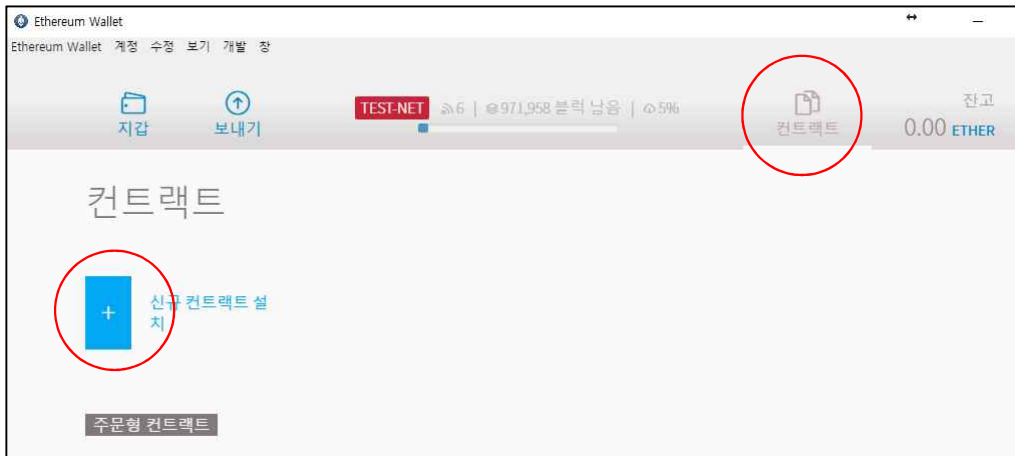


- 추가 지갑 계정 만들기

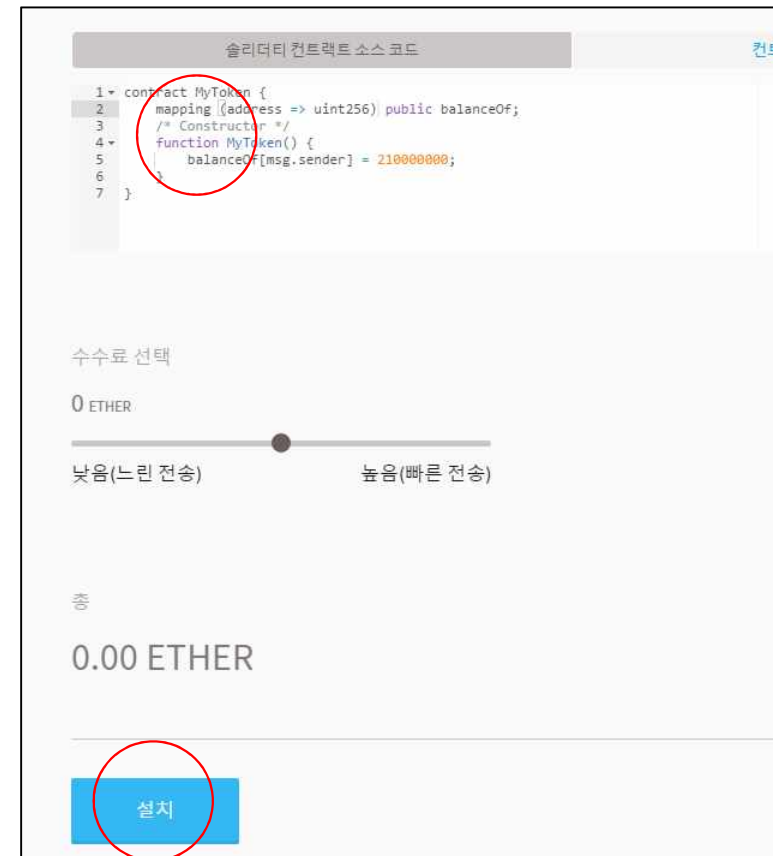


## 09 이더리움 지갑으로 스마트 계약 배포하기

- 스마트 계약 메뉴



- 배포해 보기



---

## 10 Solidity 란?

스마트 계약에서 가장 자주 쓰는 언어

---

- 이더리움을 위한 언어
- 계약 지향 언어이지만, 엄밀하게 객체지향에 속함
- 자바스크립트와 비슷
- 생성과 실행을 위해서는 수수료가 발생
  - 수수료는 Gas 단위로 산정함
  - 호출 알고리즘의 복잡도에 따라 과금됨
  - $1 \text{ Gas} = 1/100,000 \text{ Ether} \approx 0.001 \text{ 원}$
  - CREATE : 100 Gas
  - CALL : 20 Gas

---

## 11 Solidity의 자료형

기존의 언어들과 비슷합니다.

---

- uint : unsigned int
- address : Ethereum address
- bool, int, String
- String
- No float in Solidity

- Array

```
uint[] memory a = new uint[](7);  
uint[] x = [uint(1), 3, 4];
```

- Hash map (address : uint)

```
mapping (address => uint) public balances;
```

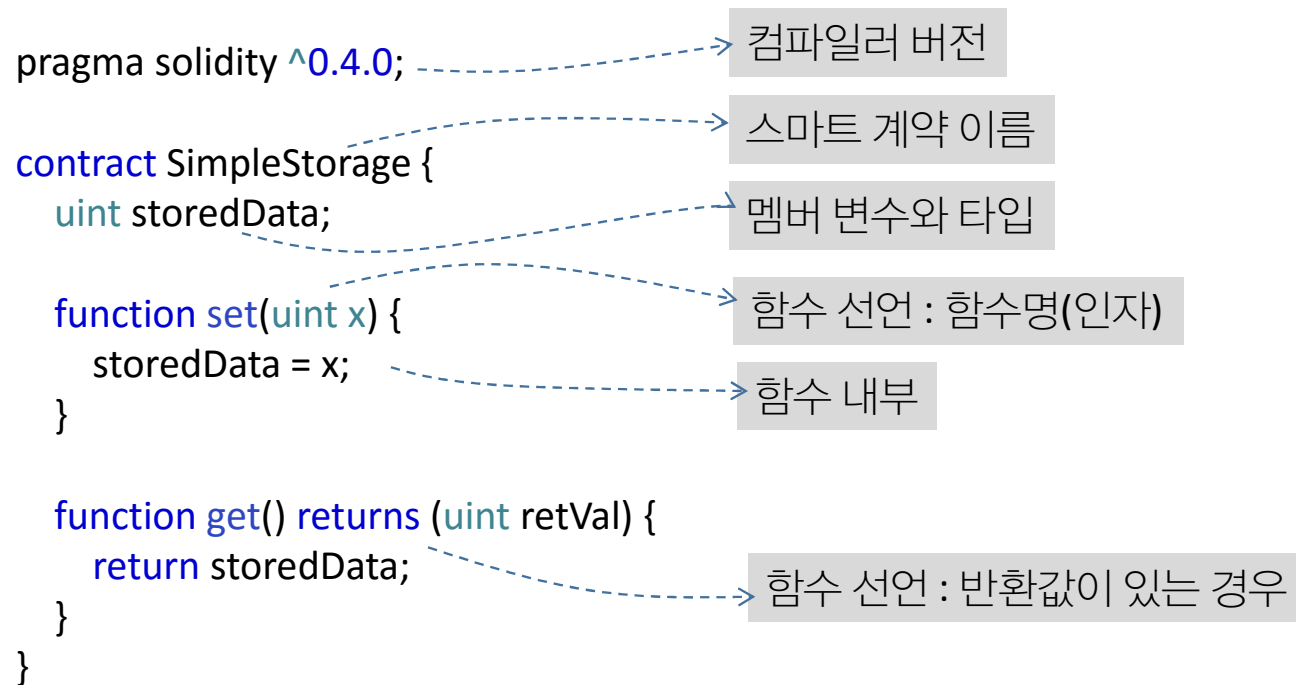
- Struct

```
struct Funder{  
    address addr;  
    uint amount;  
}
```

# 12

## 가장 간단한 예제

이더리움에 나만의 저장소 만들기



---

## 13 Solidity Web Compiler

약간 신뢰도가 떨어지지만, 간단히 작성해보긴 좋습니다.

- 
- <https://ethereum.github.io/browser-solidity/>

14

# Solidity Web Compiler 사용법

## 스마트 계약 배포 방법

The screenshot shows the Solidity IDE interface. On the left, a code editor displays a Solidity contract named `SimpleStorage` with the following code:

```
1 pragma solidity ^0.4.0;
2
3 contract SimpleStorage {
4     uint storedData;
5
6     function set(uint x) {
7         storedData = x;
8     }
9
10    function get() returns (uint retVal) {
11        return storedData;
12    }
13 }
14
```

Below the code editor, a label "스마트 계약코드 영역" (Smart Contract Code Area) is visible.

On the right, the "Compiler" panel is active, showing the "Compiler Selection" (컴파일러 선택) section. It displays the Solidity version as "0.4.1+commit.4fc6fc2c.Emscripten.clang" and the change to "0.4.2-nightly.2016.9.9". The "Auto Compile" checkbox is checked. A "Compile" button is present.

Below the compiler settings, the "SimpleStorage" contract is listed with a size of "177 bytes". There are two buttons: "At Address" and "Create". The "Create" button is highlighted with a red circle.

Below the "Create" button, the "Bytecode" section shows the contract's bytecode. The "Interface" section shows the contract's interface. The "Web3 deploy" section shows the deployment code. The "uDApp" section shows the contract's ABI.

A label "스마트 코드 정보" (Smart Code Information) is visible over the "Web3 deploy" and "uDApp" sections.



# 15 Solidity Web Compiler 사용법

## 스마트 계약 실행 방법

The screenshot displays the Solidity Web Compiler (SWC) interface. On the left, the contract code for `SimpleStorage` is shown in a text editor. The code defines a `uint` variable `storedData`, a `set` function to update it, and a `get` function to retrieve it. On the right, the deployment configuration panel is visible. A red circle highlights the 'Run' button (a paper plane icon) in the top toolbar. Below the toolbar, the 'Transaction origin' is set to `0xca35b7d915458ef540`, the 'Gas limit' is `3000000`, and the 'Value' is `0`. A box labeled '트랜잭션 정보' (Transaction Information) is overlaid on this section. Further down, the 'SimpleStorage' contract is listed with a size of 177 bytes. A box labeled '스마트 계약 생성 비용' (Smart Contract Creation Cost) is overlaid here. Below the contract list, the 'Transaction cost' is 94931 gas and the 'Execution cost' is 32263 gas. At the bottom, the 'SimpleStorage at' section shows the contract address `0x8609a0806279c94bcc5432e36b57281b3d524b9b` in memory. A box labeled '함수 호출 영역' (Function Call Area) is overlaid on this section, showing buttons for `(fallback)`, `get`, and `set` with a parameter `uint256 x`.

```
1 pragma solidity ^0.4.0;
2
3 contract SimpleStorage {
4     uint storedData;
5
6     function set(uint x) {
7         storedData = x;
8     }
9
10    function get() returns (uint retVal) {
11        return storedData;
12    }
13 }
14
```

Transaction origin: 0xca35b7d915458ef540  
Gas limit: 3000000  
Value (e.g. .7 ether or 5 wei, defaults to ether): 0

Attach Transact Call

SimpleStorage 177 bytes

At Address Create

Transaction cost: 94931 gas.  
Execution cost: 32263 gas.

SimpleStorage at 0x8609a0806279c94bcc5432e36b57281b3d524b9b (memory)

(fallback)  
get  
set uint256 x

## 16 Solidity Web Compiler 사용법

### 실행 결과 보기

- set 함수 호출

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

The screenshot shows the Solidity Web Compiler interface. At the top, there are three function buttons: '(fallback)', 'get', and 'set'. The 'set' button is circled in red. To the right of the 'set' button is a text input field containing the value '12345'. Below the input field, the text '매개 변수 설정' (Parameter Setting) is displayed. At the bottom left, the execution results are shown: 'Result: "0x"', 'Transaction cost: 41681 gas.', and 'Execution cost: 20153 gas.'. At the bottom right, a button labeled '함수 실행 결과' (Function Execution Result) is visible.

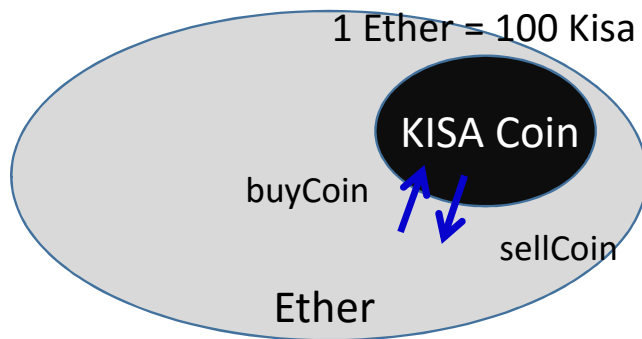
- get 함수 호출

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

The screenshot shows the Solidity Web Compiler interface. At the top, there are two function buttons: '(fallback)' and 'get'. The 'get' button is circled in red. Below the buttons, the execution results are shown: 'Result: "0x00"', 'Transaction cost: 21534 gas.', and 'Execution cost: 262 gas.'. Below these, the 'Decoded:' section shows '1. uint256 retVal: 12345'. At the bottom right, a button labeled '저장소 내용 보기' (View Storage Content) is visible.

## 17 실습 1) Alt Coin

우리 사업체를 위한 코인을 만들어 봅시다.



이더리움은 부정확성의 문제로 소수점을 사용하지 않음

- 1 Eth는 너무 크기 때문에 WEI로 바꿔서 사용한다.
- 1 Eth = 10000000000000000000 wei
- 1 Eth = 20 Kisa
- 1 Kisa = 5000000000000000000 wei

```
contract kisaCoin {  
    mapping (address => uint) public balances;  
    uint KISA_PER_WEI = 5000000000000000000;  
  
    function buyCoin() {}  
    function sellCoin(uint amount){}  
    function showMyCoin() returns(uint amount){}  
}
```

balance

ADDRESS(address)	AMOUNT(unit)
홍길동	1000000 Kisa
임꺽정	1000000 Kisa

amount = balance[홍길동]

## 18 코인 구매하기 함수

메커니즘을 이해해 봅시다.

```
function buyCoin() {  
    balances[msg.sender] += msg.value / KISA_PER_WEI;  
    if (!msg.sender.send(msg.value % KISA_PER_WEI))  
        throw;  
}
```

Transaction origin: 0xca35b7d915458ef540  
Gas limit: 3000000  
Value (e.g. .7 ether or 5 wei, defaults to ether): 0

함수실행 시 보내지는 두 가지 매개 변수

1. 보내는 사람의 주소
2. 보내는 돈의 양

`address.send(200)` → 나의 지갑에서 address로 x wei를 보냄

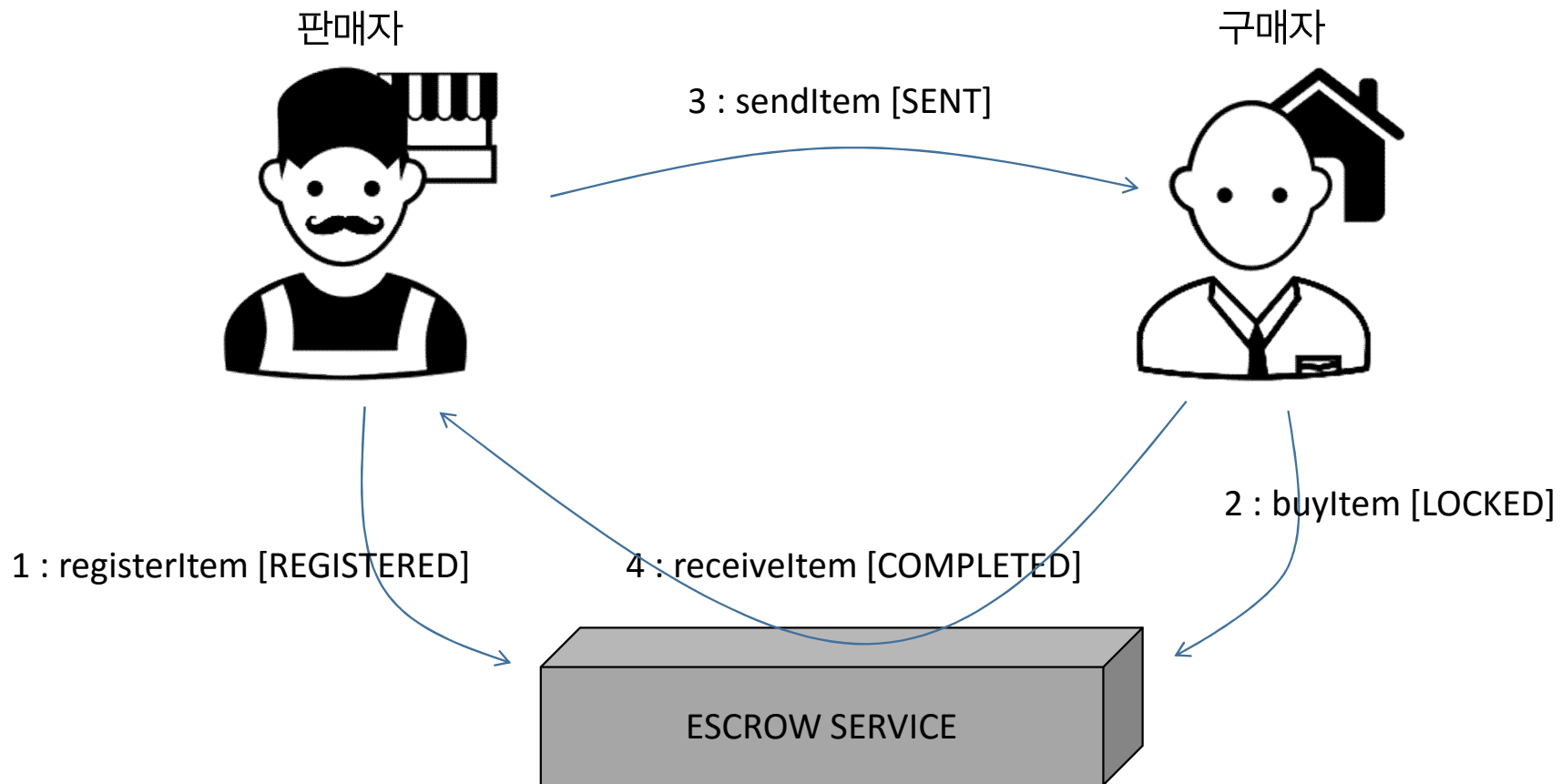
## 19 코인 팔기와 잔액 확인

한번 풀어보세요.

```
function sellCoin(uint amount){  
  if( _____ ){  
    balances[ _____ ] -= amount;  
    if( ! _____ ){  
      throw;  
    }  
  }  
}  
function showMyCoin() returns(uint amount){  
  return _____ ;  
}
```

요청 액보다 잔액이 많다면,  
해당 주소의 Kisa coin 빼기  
해당 주소에게 Ether 돌려주기  
해당 주소의 잔액 보여주기

## 20 실습 2) 안전거래 (Escrow) 서비스



---

## 21    멤버 변수와 함수 작성

한번 완성해 봅시다.

---

```
contract EscrowPurchase {  
    enum ItemState {CREATED, LOCKED, SENT, COMPLETED}  
  
    uint public price;  
    address public seller;  
    address public buyer;  
    ItemState public state;  
  
    function registerItem(uint amount) {}  
    function buyItem(){}  
    function sendItem(){}  
    function receiveItem(){}  
}
```

- 
- 감사합니다.