

Smart Contract

강사 소개

- 마경욱
 - 스마일 게이트 딥러닝 연구팀 재직 중
 - 모바일/백엔드/블록체인 프로젝트 진행
 - kyeongwook.ma@gmail.com
 - <https://github.com/shwarzes89>

목차

1. 개요
2. 개발 환경 설정
3. Solidity
4. 예제

개요

개요

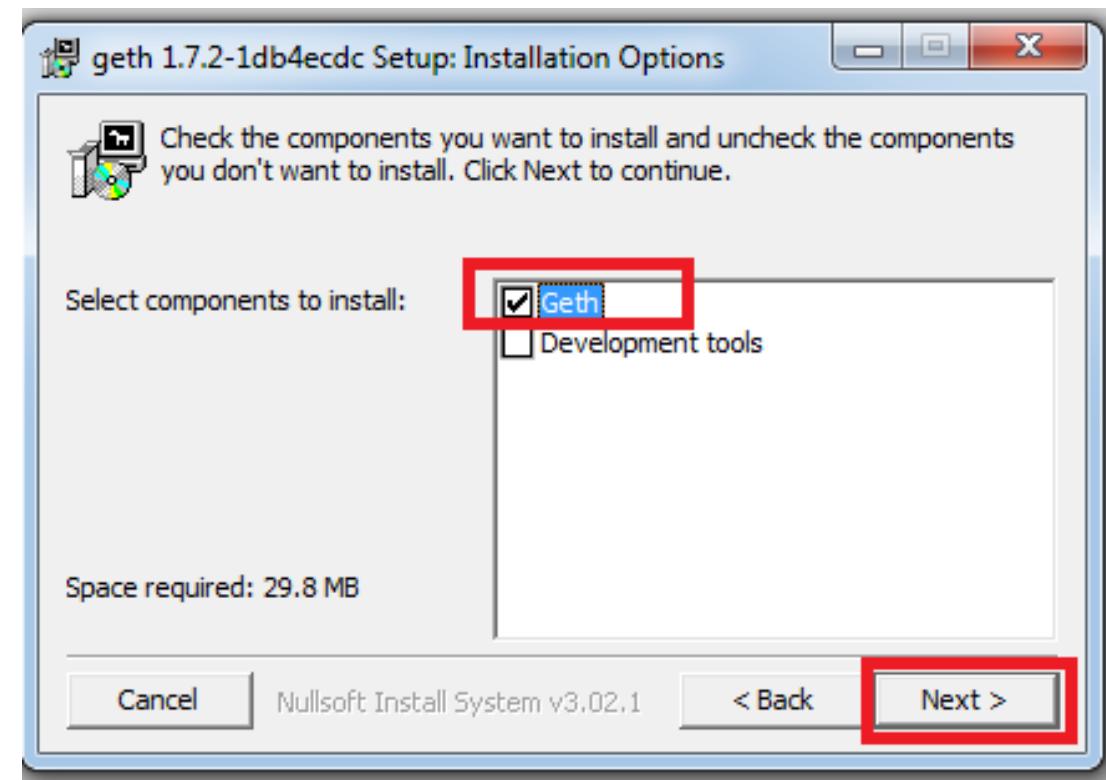
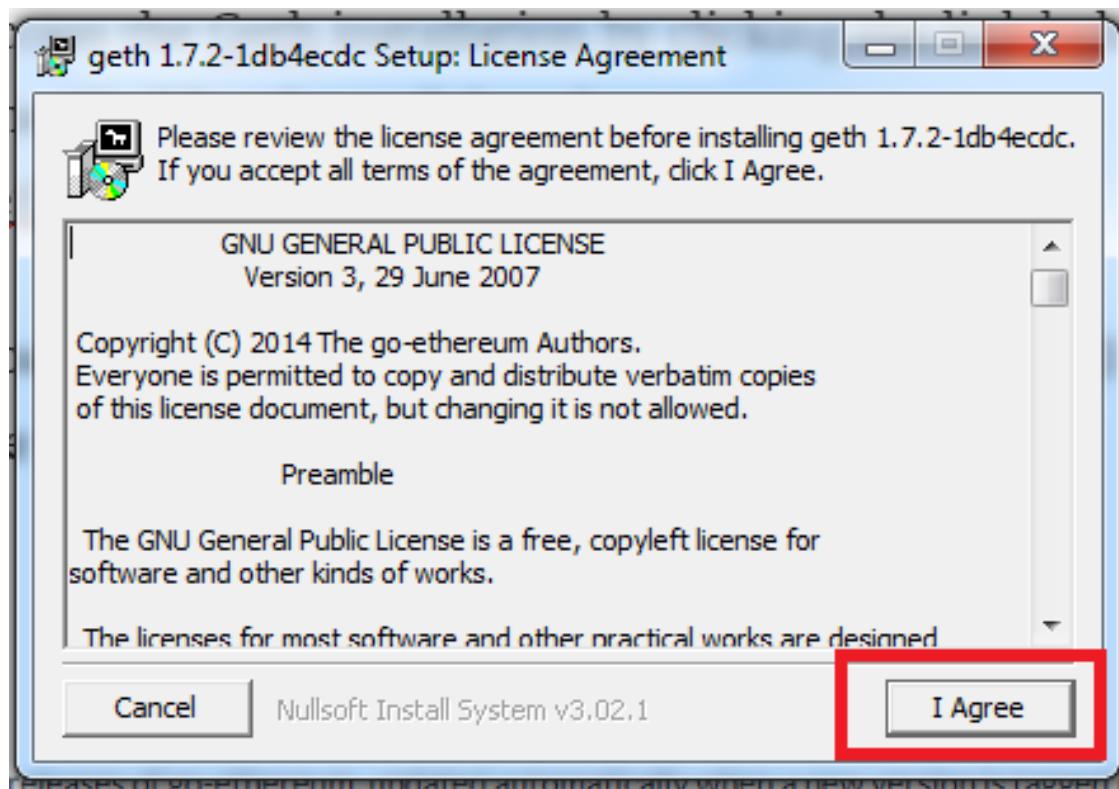
- 스마트 컨트랙트
 - 디지털 명령어로 계약을 작성하면 조건에 따라 계약 내용을 자동 실행
 - 복잡한 프로세스를 간소화
 - 조작 불가
 - 즉각 이행 가능

개발 환경 설정

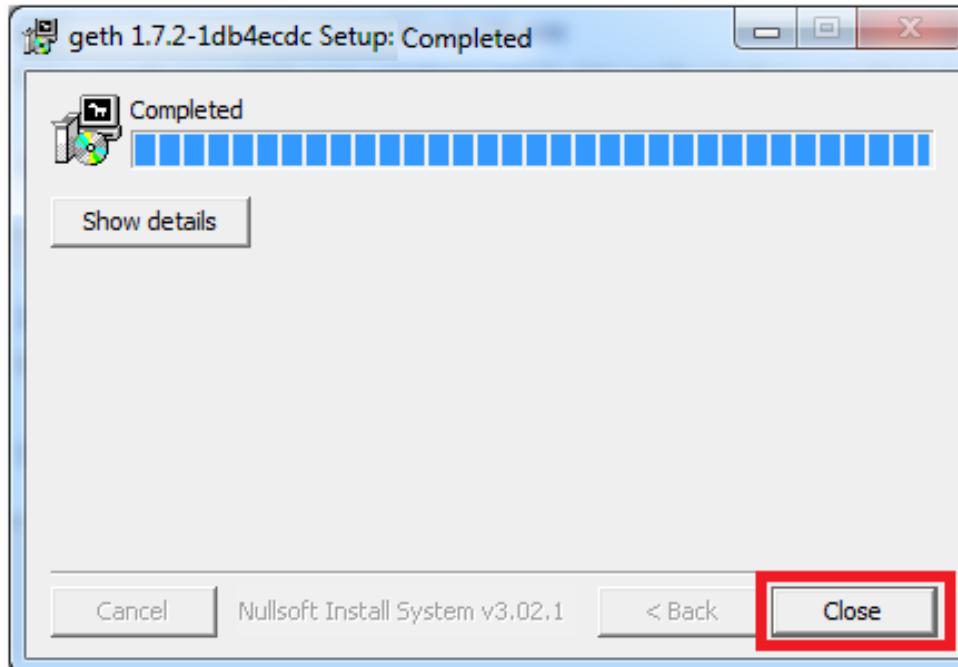
Geth 설치 및 실행

- <https://geth.ethereum.org/downloads/>
- Go언어로 작성된 Ethereum node
- 운영체제에 맞는 설치 파일 다운로드
- 환경 변수 설정
 1. `~/.bash_profile` 수정
 2. `export GETH=~/geth` 추가
 3. `export PATH=${PATH}:${GETH}` 추가
 4. `source ~/.bash_profile`
- 실행

Geth 설치 및 실행



Geth 설치 및 실행



The screenshot shows a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe - geth". The window displays the output of the "geth" command, which is the Ethereum node daemon. The log output includes various INFO messages detailing the node's initialization, including the start of a peer-to-peer node, allocation of memory and file handles, initial chain configuration, and loading of local headers and blocks. It also mentions DAO support, EIPs, and disk storage for ethash caches and DAGs. The logs conclude with a warning about the blockchain not being empty and fast sync being disabled, followed by the start of P2P networking.

```
C:\Program Files\Geth>geth
INFO [11-04|22:28:32] Starting peer-to-peer node instance=Geth/v1.6.7-stable-ab5646c5/windows-amd64/go1.8.3
INFO [11-04|22:28:32] Allocated cache and file handles database=C:\Users\Reggie\AppData\Roaming\Ethereum\geth\chaindata cache=128 handles=1024
INFO [11-04|22:28:33] Initialised chain configuration config={"ChainID:1 Homestead: 1150000 DAO: 1920000 DAOSupport: true EIP150: 2463000 EIP155: 2675000 EIP158: 2675000 Metropolis: 9223372036854775807 Engine: ethash}"
INFO [11-04|22:28:33] Disk storage enabled for ethash caches dir=C:\Users\Reggie\AppData\Roaming\Ethereum\geth\ethash count=3
INFO [11-04|22:28:33] Disk storage enabled for ethash DAGs dir=C:\Users\Reggie\AppData\Ethash count=2
INFO [11-04|22:28:33] Initialising Ethereum protocol versions="[63 62]"
"network=1
INFO [11-04|22:28:33] Loaded most recent local header number=4370000 hash=57bfb5.46c979 td=1196768510090289371283
INFO [11-04|22:28:33] Loaded most recent local full block number=4370000 hash=57bfb5.46c979 td=1196768510090289371283
INFO [11-04|22:28:33] Loaded most recent local fast block number=4370000 hash=57bfb5.46c979 td=1196768510090289371283
WARN [11-04|22:28:33] Blockchain not empty, fast sync disabled
INFO [11-04|22:28:33] Starting P2P networking
```

Private blockchain network 개설

1. mkdir private_chain
2. mkdir private_chain/chaindata && cd private_chain/chaindata
3. vim **genesis.json**
 1. Copy
<https://gist.github.com/shwarzes89/fffbc91cbce71f716155020e21185d3>
 2. Paste

Private blockchain network 개설

- Network 초기화
 - geth --datadir=./chaindata/ init ./genesis.json

```
INFO [03-02|09:55:49] Maximum peer count
INFO [03-02|09:55:49] Allocated cache and file handles
INFO [03-02|09:55:49] Writing custom genesis block
INFO [03-02|09:55:49] Persisted trie from memory database
INFO [03-02|09:55:49] Successfully wrote genesis state
INFO [03-02|09:55:49] Allocated cache and file handles
INFO [03-02|09:55:49] Writing custom genesis block
INFO [03-02|09:55:49] Persisted trie from memory database
INFO [03-02|09:55:49] Successfully wrote genesis state
```

```
ETH=25 LES=0 total=25
database=/Users/user/private_chain/chaindata/geth/chaindata cache=16 handles=16
nodes=0 size=0.00B time=9.415µs gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
database=chaindata hash=2fb1a7..f0181a
database=/Users/user/private_chain/chaindata/geth/lightchaindata cache=16 handles=16
nodes=0 size=0.00B time=2.466µs gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
database=lightchaindata hash=2fb1a7..f0181a
```

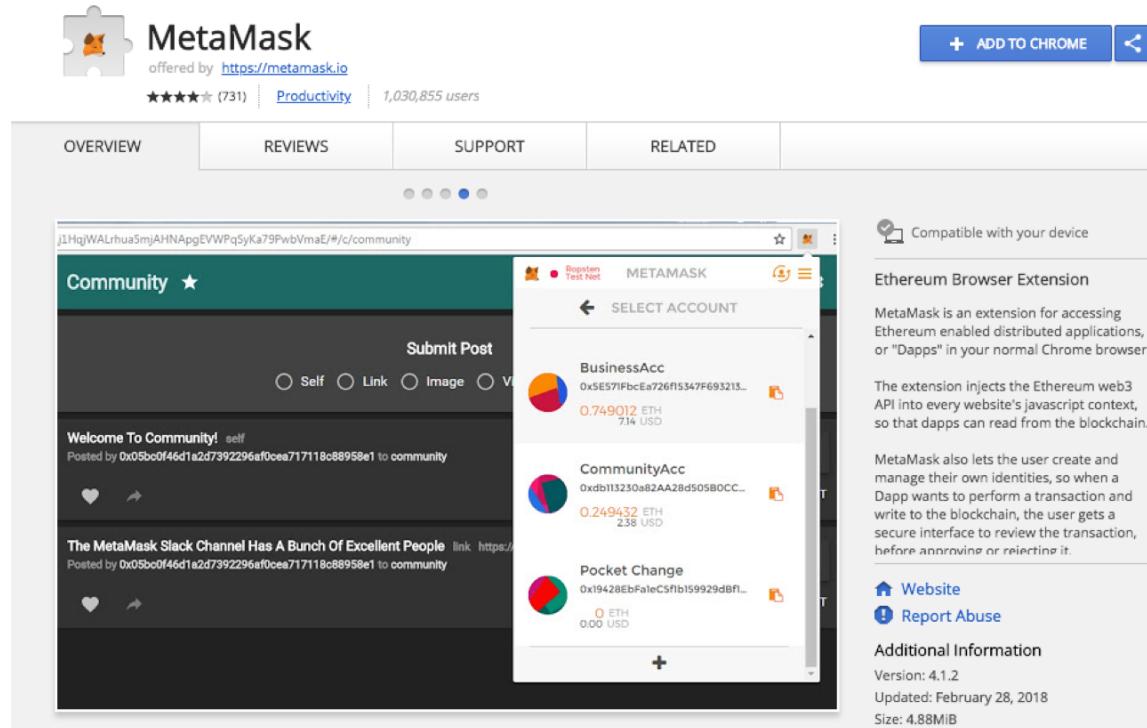
Private blockchain network 개설

- 실행
 - geth --datadir=./chaindata/

```
INFO [03-02|10:04:38] Starting P2P networking
INFO [03-02|10:04:40] UDP listener up
a0c10e4644ad757302f03bd50c74992@[::]:30303
INFO [03-02|10:04:40] RLPx listener up
a0c10e4644ad757302f03bd50c74992@[::]:30303
INFO [03-02|10:04:40] IPC endpoint opened
[...]
self=enode://2636ea60b4c3eb9db6241b9708b88e13cc000fab6cee67001392196d9dbdabf21038aae395c4b2735f3a4cdf501ac84b8
self=enode://2636ea60b4c3eb9db6241b9708b88e13cc000fab6cee67001392196d9dbdabf21038aae395c4b2735f3a4cdf501ac84b8
url=/Users/user/private_chain/chaindata/geth.ipc
```

MetaMask

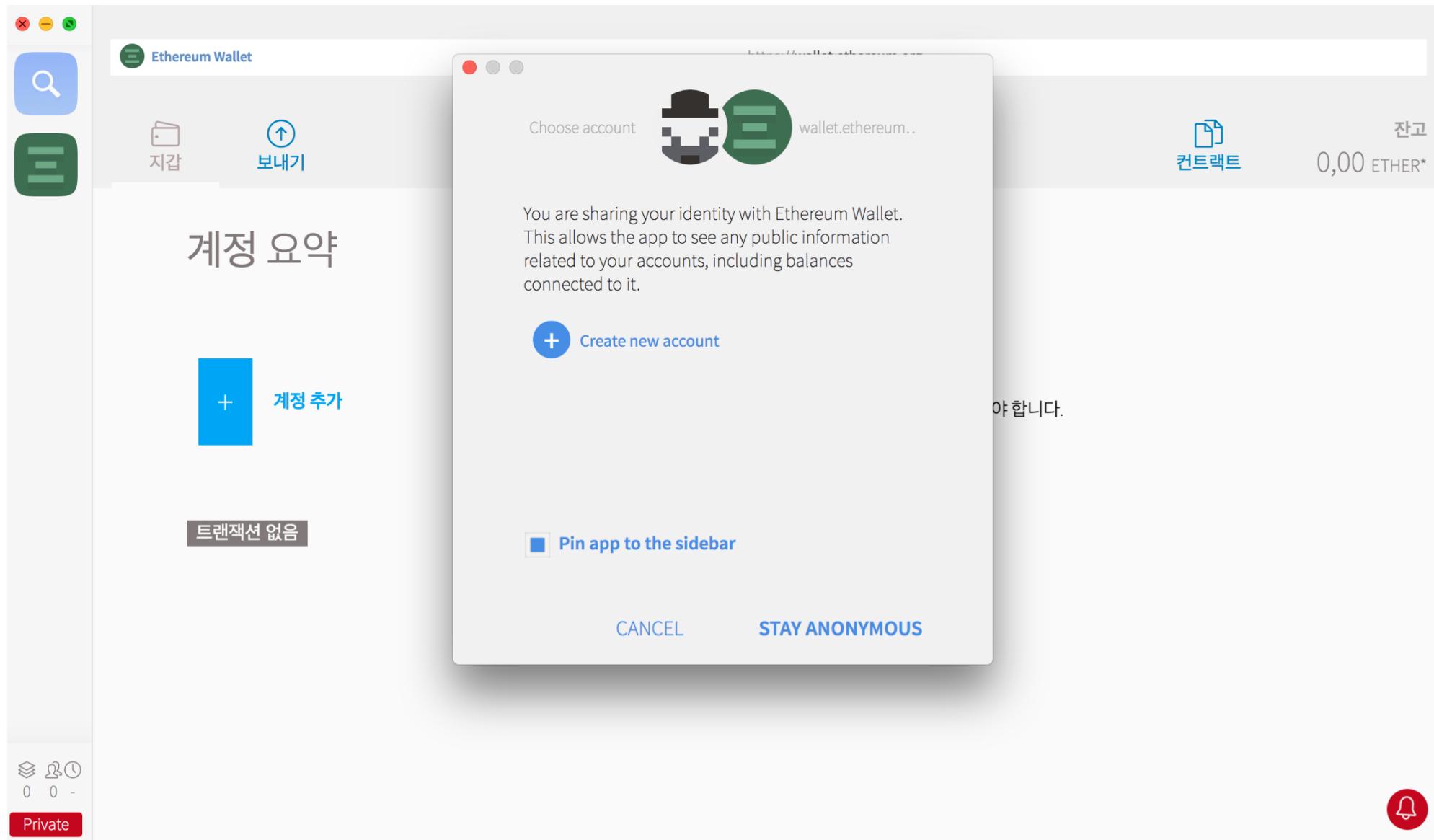
- Ethereum 브라우저 & 지갑
- Blockchain을 다운받을 필요 없이 Dapp, smart contract와 연동
- <https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn?hl=en>



Mist

- Ethereum GUI 지갑
- 설치
 - <https://github.com/ethereum/mist/releases>
- 실행
 - Geth 실행
 - geth --datadir=./chaindata/
 - Mist 실행
 - ❖ Mac의 경우 /Applications/Mist.app/Contents/MacOS/Mist --rpc ./private_chain/chaindata/geth.ipc

Mist



Mist

The screenshot shows the Ethereum Wallet Mist interface. At the top, there's a navigation bar with a magnifying glass icon, the text "Ethereum Wallet", and a URL "https://wallet.ethereum.org › account › 0xa7b5038635a6142c471fc5dd9ea055f44709cb57". Below the navigation bar, there are several icons: a folder (지갑), an upward arrow (보내기), a document (컨트랙트), and a balance indicator (잔고) showing "0,00 ETHER*". On the left side, there's a sidebar with a green "M" icon, a magnifying glass icon, and a green circular icon with a pixelated pattern. The main content area displays the "Main account (Etherbase)" with the address "Oxa7B5038635A6142C471Fc5DD9ea055F44709cB57" and a balance of "0,00 ETHER*". To the right of the account details, there are three buttons: a downward arrow labeled "이더 입금" (Ether Deposit), a clipboard icon labeled "주소 복사" (Address Copy), and a camera icon labeled "QR 코드 보기" (QR Code View). A note at the bottom left says "메모" (Note) and provides instructions about transaction visibility. A bold note at the bottom center reads "If your balance doesn't seem updated, make sure that you are in sync with the network."

Ethereum Wallet

https://wallet.ethereum.org › account › 0xa7b5038635a6142c471fc5dd9ea055f44709cb57

지갑

보내기

잔고
0,00 ETHER*

Main account (Etherbase)

Oxa7B5038635A6142C471Fc5DD9ea055F44709cB57

0,00 ETHER*

메모

계정들은 이더를 보관하고 전송할 수 있지만 개별 입금되는 트랜잭션들을 보여줄 수 없습니다. 개별 입금 트랜잭션들을 보려면 [지갑 컨트랙트 설치하기](#)를 해야합니다.

If your balance doesn't seem updated, make sure that you are in sync with the network.

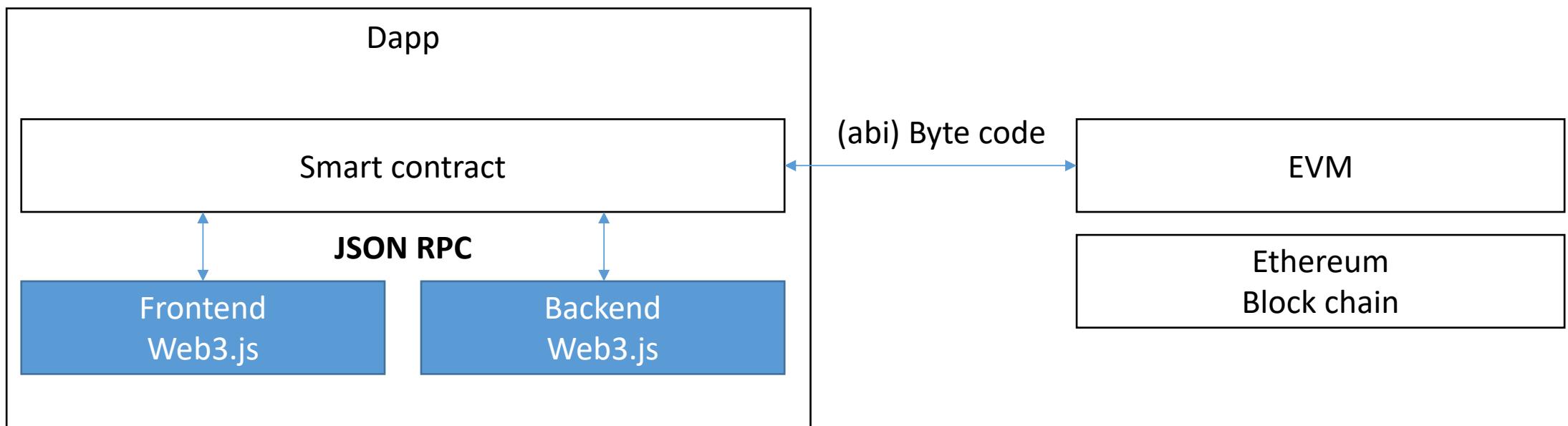
이더 입금

주소 복사

QR 코드 보기

Web3.js

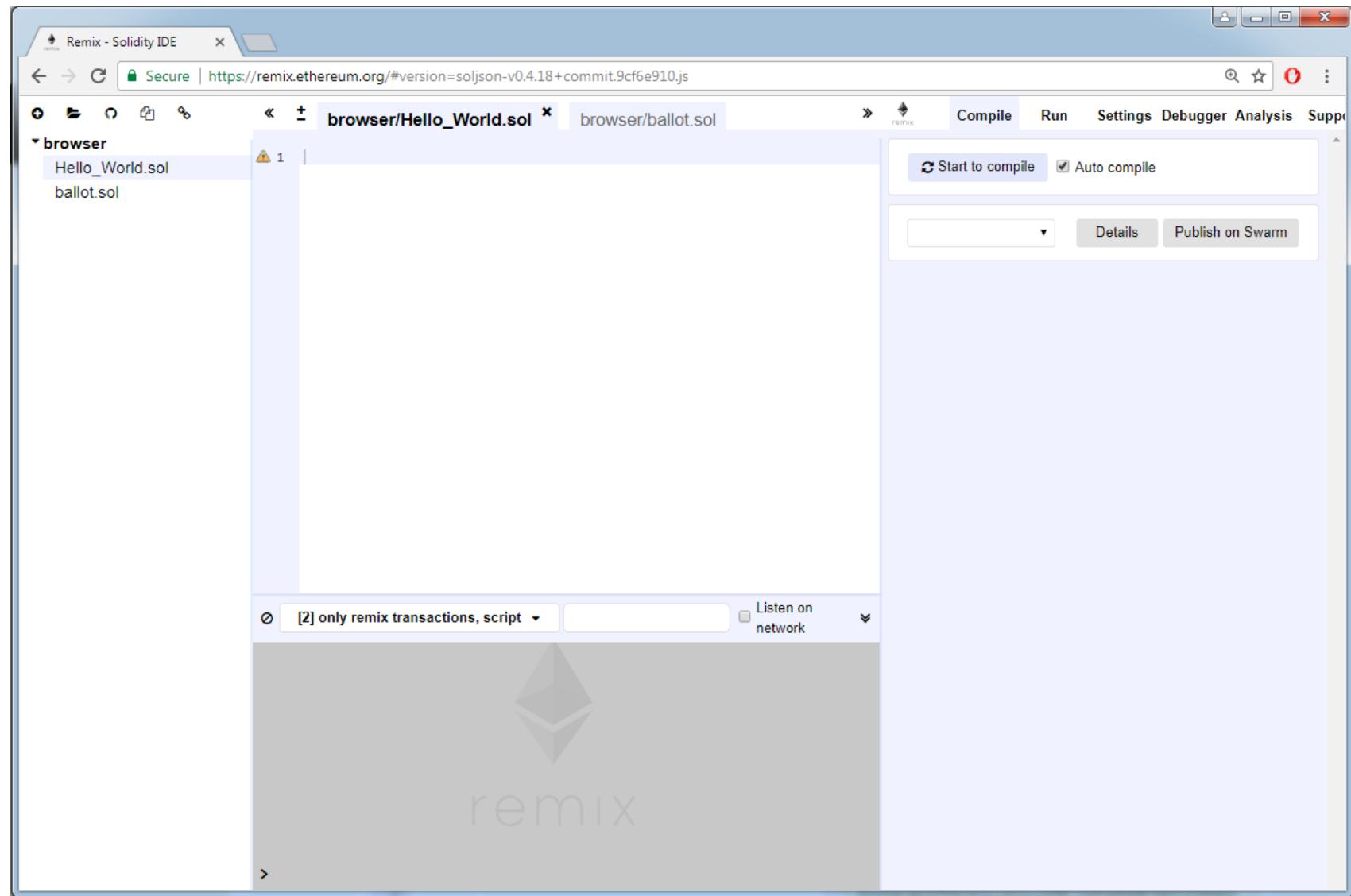
- Ethereum 노드와 브라우저간의 통신
- HTTP 나 IPC 연결을 통해 JSON RPC를 통하여 ethereum과 통신
- <https://github.com/ethereum/wiki/wiki/JavaScript-API>



Remix

- IDE (정적 분석, 구문 검사, ...)
- Web3 와 연동 가능
- MetaMask 나 Mist 에 직접 배포 가능
- <https://remix.ethereum.org/>

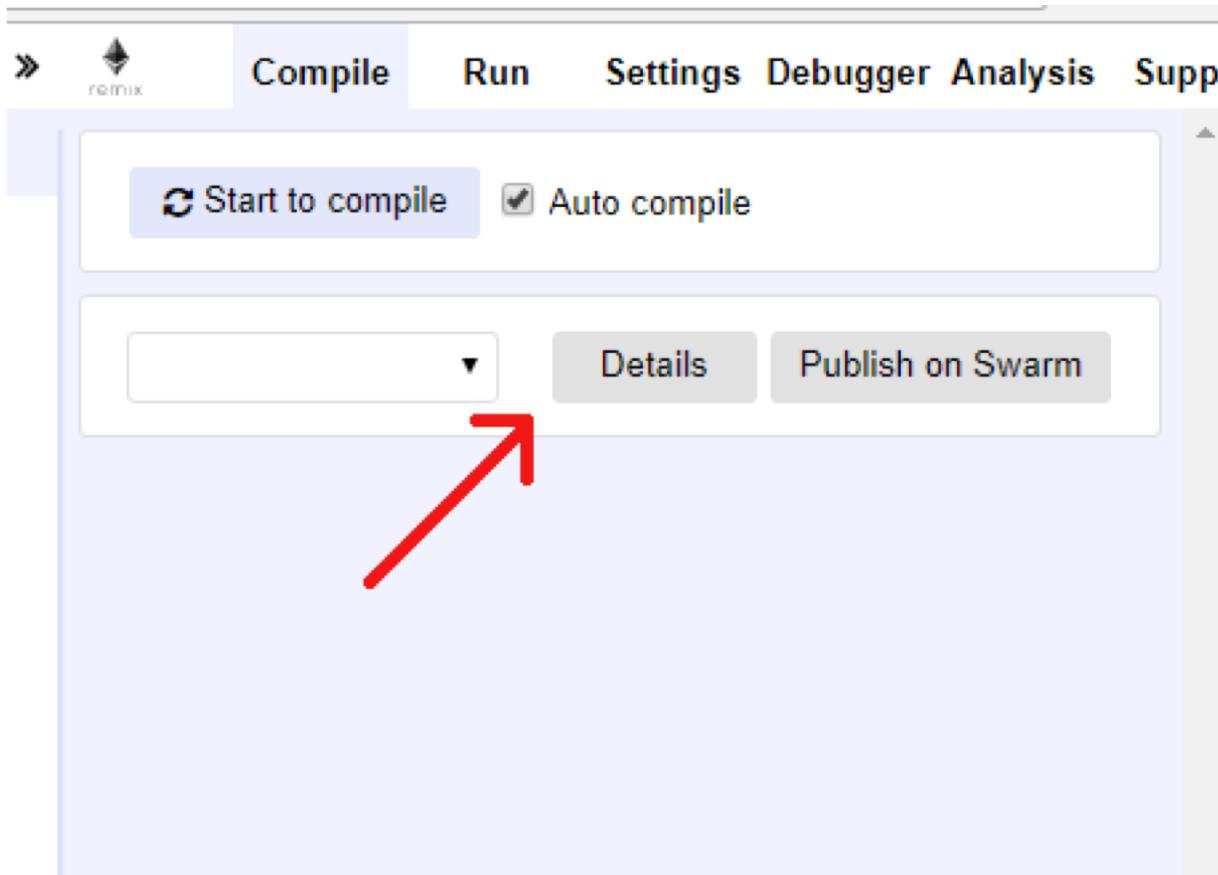
Remix



Remix

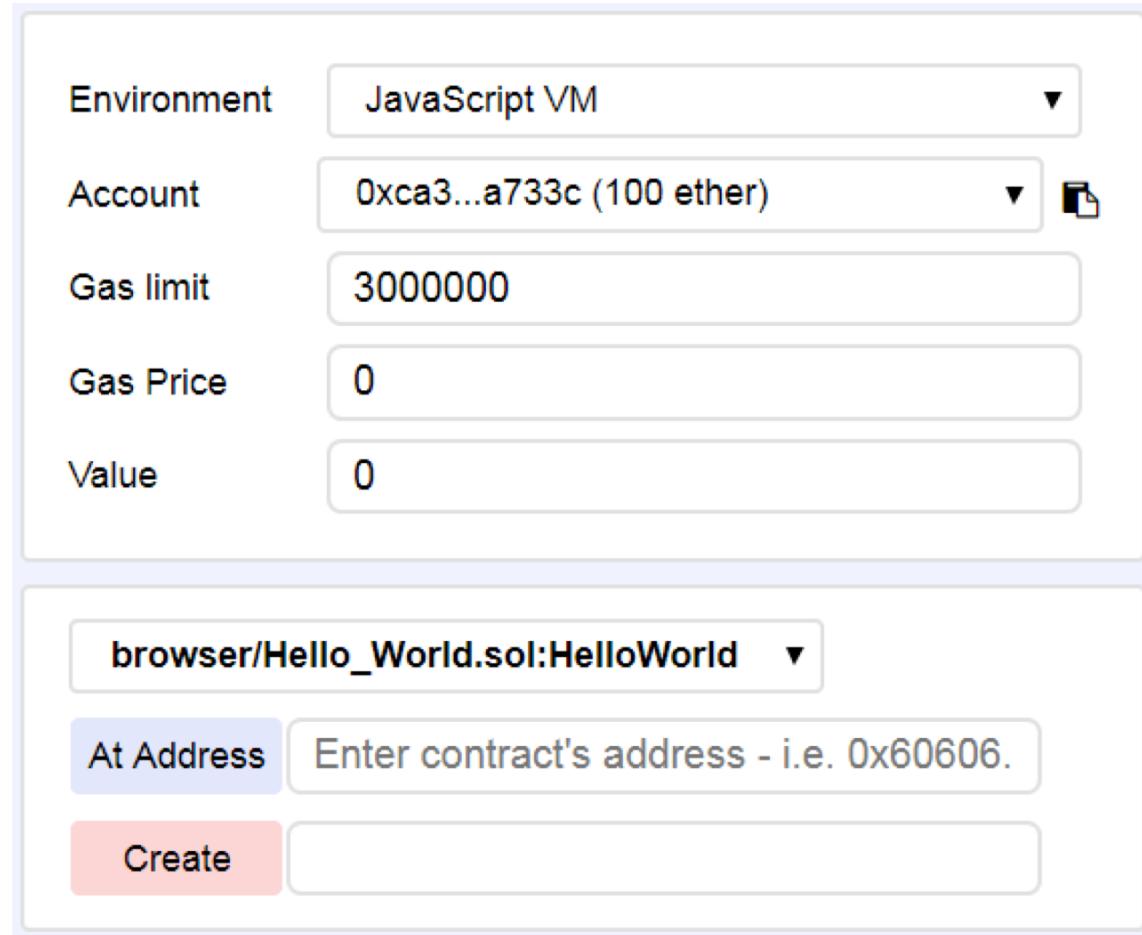
- 소스 코드 브라우져
- 소스 코드 에디터
- 스마트 컨트랙트의 컴파일 및 배포, 디버깅, 분석
 - 환경
 - Javascript VM : geth 연결 없이 모든 개발이 Remix상에서 이뤄짐
 - Injected Web3 : Mist나 Metamask와 같이 공급자에 의해 제공되는 환경
 - Web3 provider : 로컬에서 구동되는 geth 노드에 연결

Remix



- 스마트 컨트랙트 정보 확인
 - ABI
 - Web3 배포 코드
 - 메타데이터
 - 바이트 코드

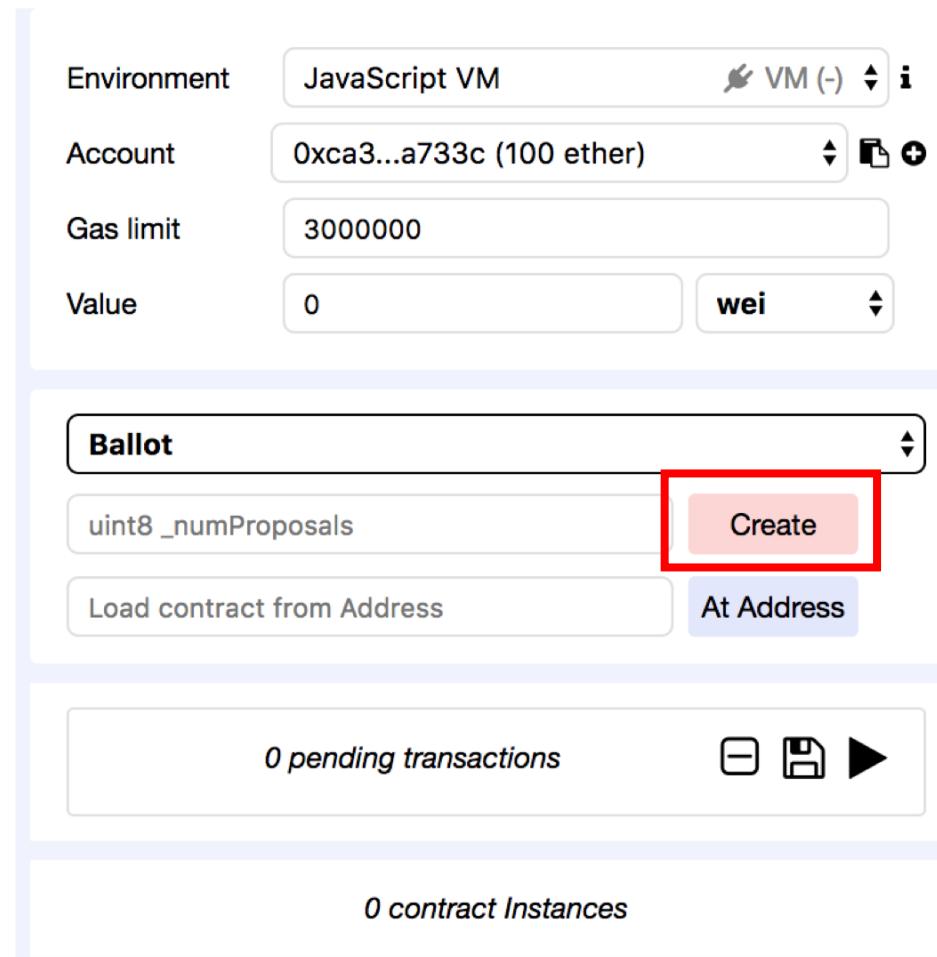
Remix



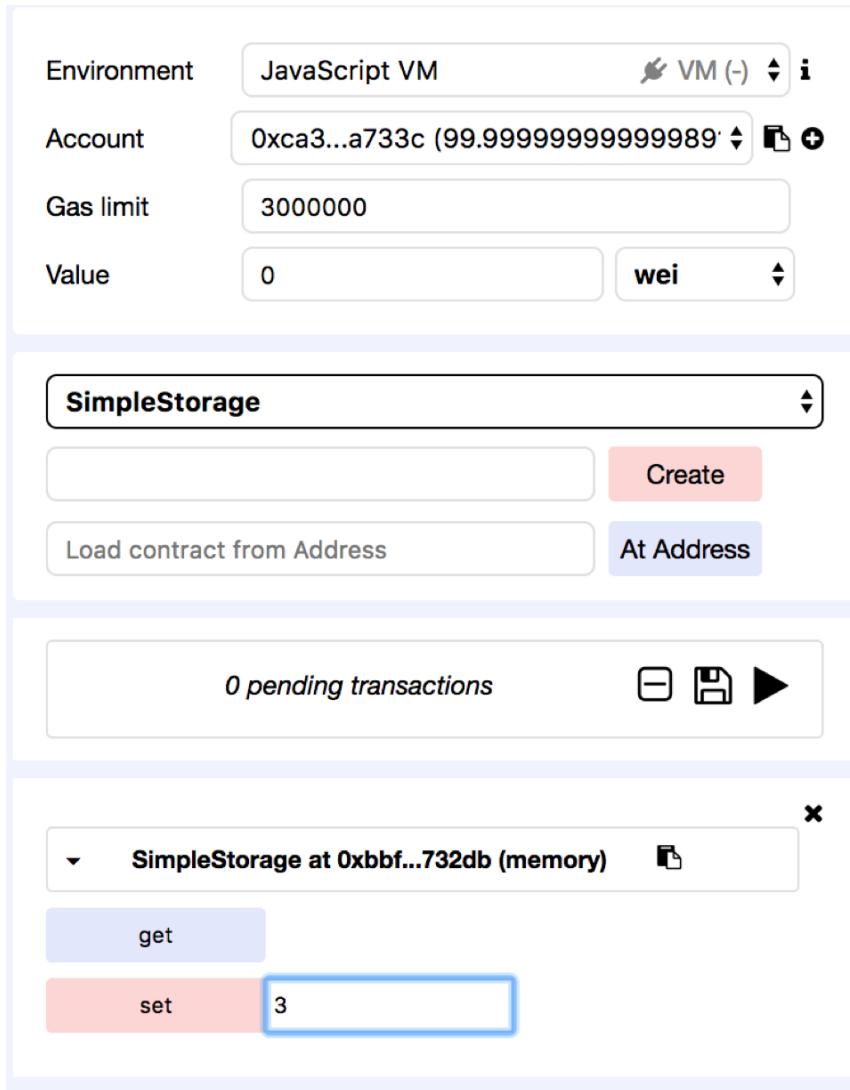
- Environment: JavaScript VM 을 이용해서 브라우저 안에서 실행
- Account: 100 ether가 있는 테스트 계정
- Gas limit: gas (사용료) 제한 설정
- Gas Price: 필요 gas 설정
- Value : 다음에 생성된 트랜잭션의 값 (트랜잭션 실행 후엔 0으로 초기화)
- Gas : Ethereum 트랜잭션을 실행시키기 위한 수수료
 - Ether는 가상화폐이기 때문에 변동성이 있으나 Gas는 변동성이 거의 없음

Hello World.sol

- <https://gist.github.com/shwarz89/495d65e31cdf655aff660e4ddb0a4e6d>
- Copy & Paste
- “Create”



Hello World.sol



- 임의의 값 설정

Hello World.sol

```
creation of SimpleStorage pending...
```

```
[vm] from:0xca3...a733c, to:SimpleStorage.(constructor), value:0 wei, data:  
0x606...d0029, 0 logs, hash:0x85c...39126
```

[Details](#)[Debug](#)

```
transact to SimpleStorage.set pending ...
```

```
[vm] from:0xca3...a733c, to:SimpleStorage.set(uint256) 0xbbf...732db, val  
ue:0 wei, data:0x60f...00003, 0 logs, hash:0x6d2...d0532
```

[Details](#)[Debug](#)

```
call to SimpleStorage.get
```

```
[call] from:0xca35b7d915458ef540ade6068dfe2f44e8fa733c, to:SimpleStorage.ge  
t(), data:6d4ce...ce63c, return:
```

```
{  
    "0": "uint256: 3"  
}
```

[Details](#)[Debug](#)

Mist에 스마트 컨트랙트 배포

- 스마트 컨트랙트를 사용하기 전 ether 필요

- geth --datadir=./chaindata/ console 명령어로 geth 에 접근
- miner.start(); 로 채굴 시작

```
INFO [03-07|08:41:34] ⚡block reached canonical chain
INFO [03-07|08:41:34] ↪ mined potential block
INFO [03-07|08:41:34] Commit new mining work
INFO [03-07|08:41:34] Successfully sealed new block
INFO [03-07|08:41:34] ⚡block reached canonical chain
INFO [03-07|08:41:34] ↪ mined potential block
INFO [03-07|08:41:34] Commit new mining work
number=57 hash=1e4b72...100cb0
number=62 hash=285f5b...1c84a9
number=63 txs=0 uncles=0 elapsed=118.134µs
number=63 hash=8fdd79...0b167d
number=58 hash=d9f203...14861b
number=63 hash=8fdd79...0b167d
number=64 txs=0 uncles=0 elapsed=123.479µs

true
> INFO [03-07|08:41:34] Generating DAG in progress
INFO [03-07|08:41:34] Generating DAG in progress
INFO [03-07|08:41:35] Generating DAG in progress
INFO [03-07|08:41:35] Generating DAG in progress
INFO [03-07|08:41:36] Generating DAG in progress
INFO [03-07|08:41:36] Generated ethash verification cache
epoch=1 percentage=95 elapsed=1m7.025s
epoch=1 percentage=96 elapsed=1m7.507s
epoch=1 percentage=97 elapsed=1m7.983s
epoch=1 percentage=98 elapsed=1m8.462s
epoch=1 percentage=99 elapsed=1m9.540s
epoch=1 elapsed=1m9.543s
```

Mist에 스마트 컨트랙트 배포

- miner.stop(); 로 채굴 정지

Ethereum Wallet <https://wallet.ethereum.org> ▶ account ▶ 0xa7b5038635a6142c471fc5dd9ea055f44709cb57

지갑 보내기 컨트랙트 잔고 315,00 ETHER*

Main account (Etherbase)
Oxa7B5038635A6142C471Fc5DD9ea055F44709cB57
315,00 ETHER*

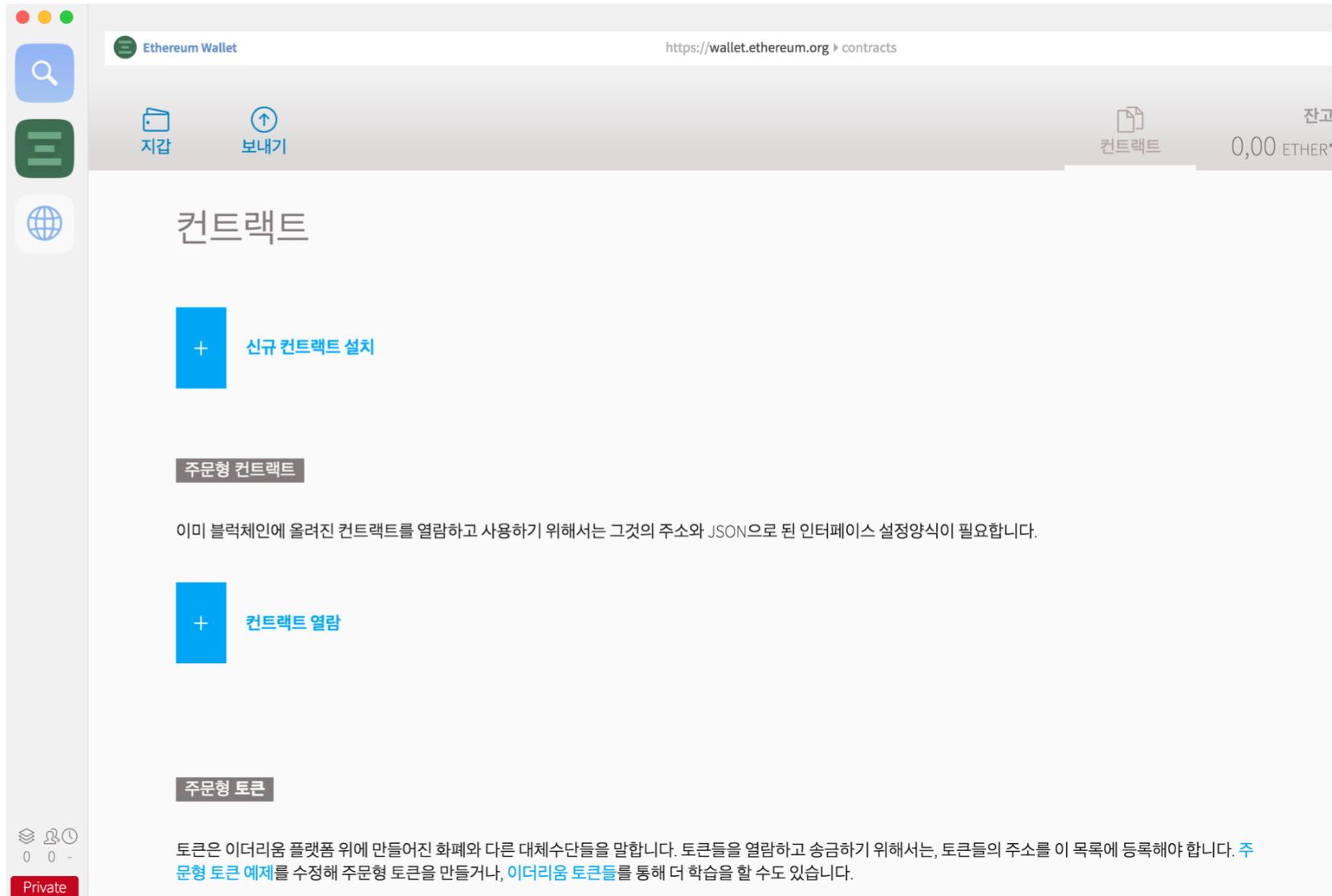
이더 입금 주소 복사 QR 코드 보기

메모

계정들은 이더를 보관하고 전송할 수 있지만 개별 입금되는 트랜잭션들을 보여줄 수 없습니다. 개별 입금 트랜잭션들을 보려면 [지갑 컨트랙트 설치하기](#)를 해야합니다.

If your balance doesn't seem updated, make sure that you are in sync with the network.

Mist에 스마트 컨트랙트 배포



Mist에 스마트 컨트랙트 배포

솔리더티 컨트랙트 소스 코드

```
1 pragma solidity ^0.4.18;
2
3 contract SimpleStorage {
4     uint storedData;
5
6     function set(uint x) public {
7         storedData = x;
8     }
9
10    function get() constant public returns (uint) {
11        return storedData;
12    }
13 }
```

컨트랙트 바이트 코드

설치할 컨트랙트를 선택하세요

Simple Storage

수수료 선택

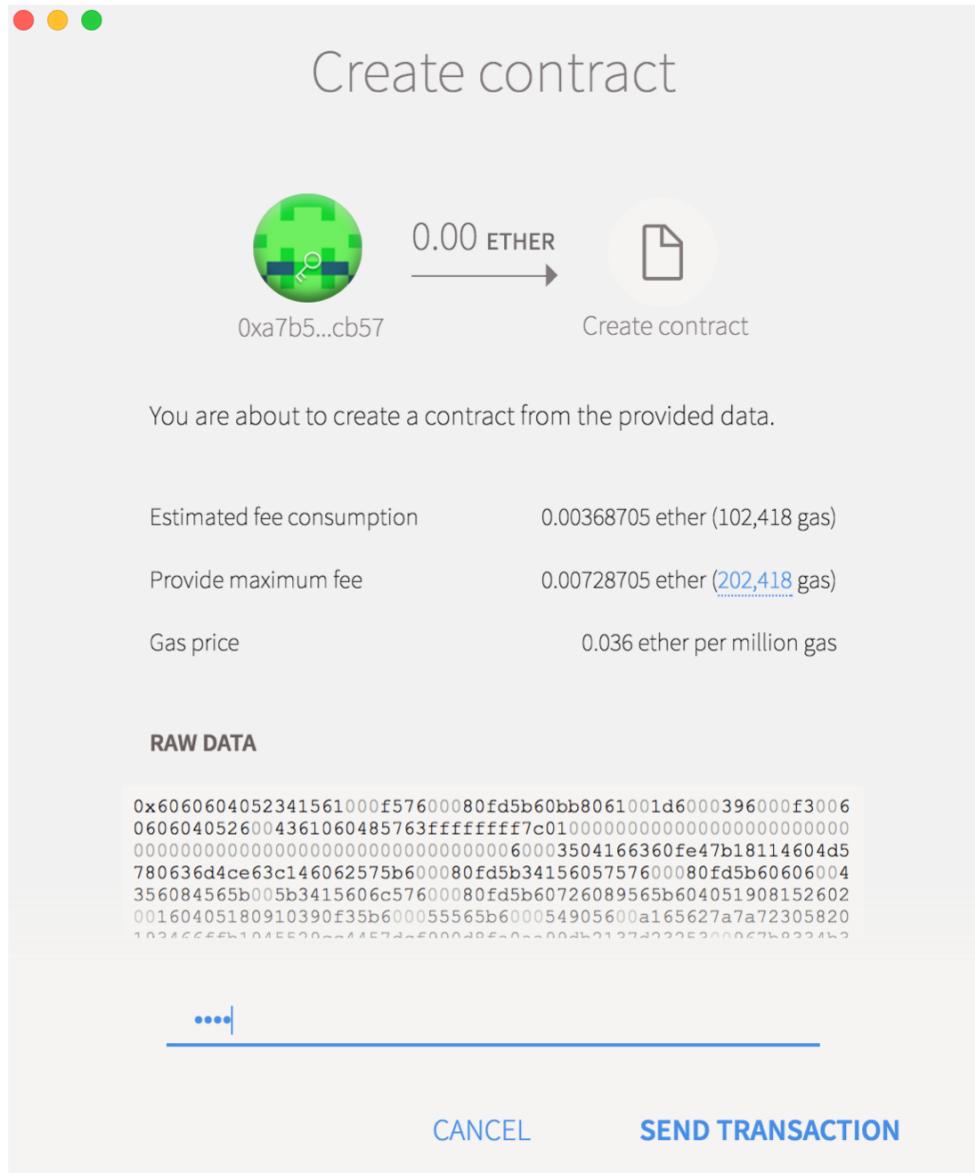
0,003687048 ETHER

낮음(느린 전송)

높음(빠른 전송)

이것은 이 트랜잭션을 처리하기 위해 사용될 최대 금액입니다. 당신의 트랜잭션은 **대략 30초 이내에**에 블록체인에 등록될 것입니다.

Mist에 스마트 컨트랙트 배포



- 계정 초기에 설정한 비밀번호
 - 초기에 비밀번호를 설정하지 않은 경우 새 계정을 만든 후 채굴 과정부터 다시 시작

Mist에 스마트 컨트랙트 배포

트랜잭션 필터링						
3월	생성된 컨트랙트			0 의 12 승인	-0,00 ETHER	
7	Account 2 → 컨트랙트 생성					

Mist에 스마트 컨트랙트 배포

트랜잭션 필터링					
3월	생성된 컨트랙트				
7	 Account 2 →  컨트랙트 생성 	0 의 12 승인	-0,00 ETHER		

↓
채굴 시작 후

트랜잭션 필터링					
3월	생성된 컨트랙트				
7	 Account 2 →  생성시간  : Simple Storage 7f27	9 의 12 승인	-0,00 ETHER		

Mist에 스마트 컨트랙트 배포

- miner.stop(); 로 채굴 정지

Ethereum Wallet <https://wallet.ethereum.org> ▶ account ▶ 0xa7b5038635a6142c471fc5dd9ea055f44709cb57

지갑 보내기 컨트랙트 잔고 315,00 ETHER*

Main account (Etherbase)
Oxa7B5038635A6142C471Fc5DD9ea055F44709cB57
315,00 ETHER*

이더 입금 주소 복사 QR 코드 보기

메모

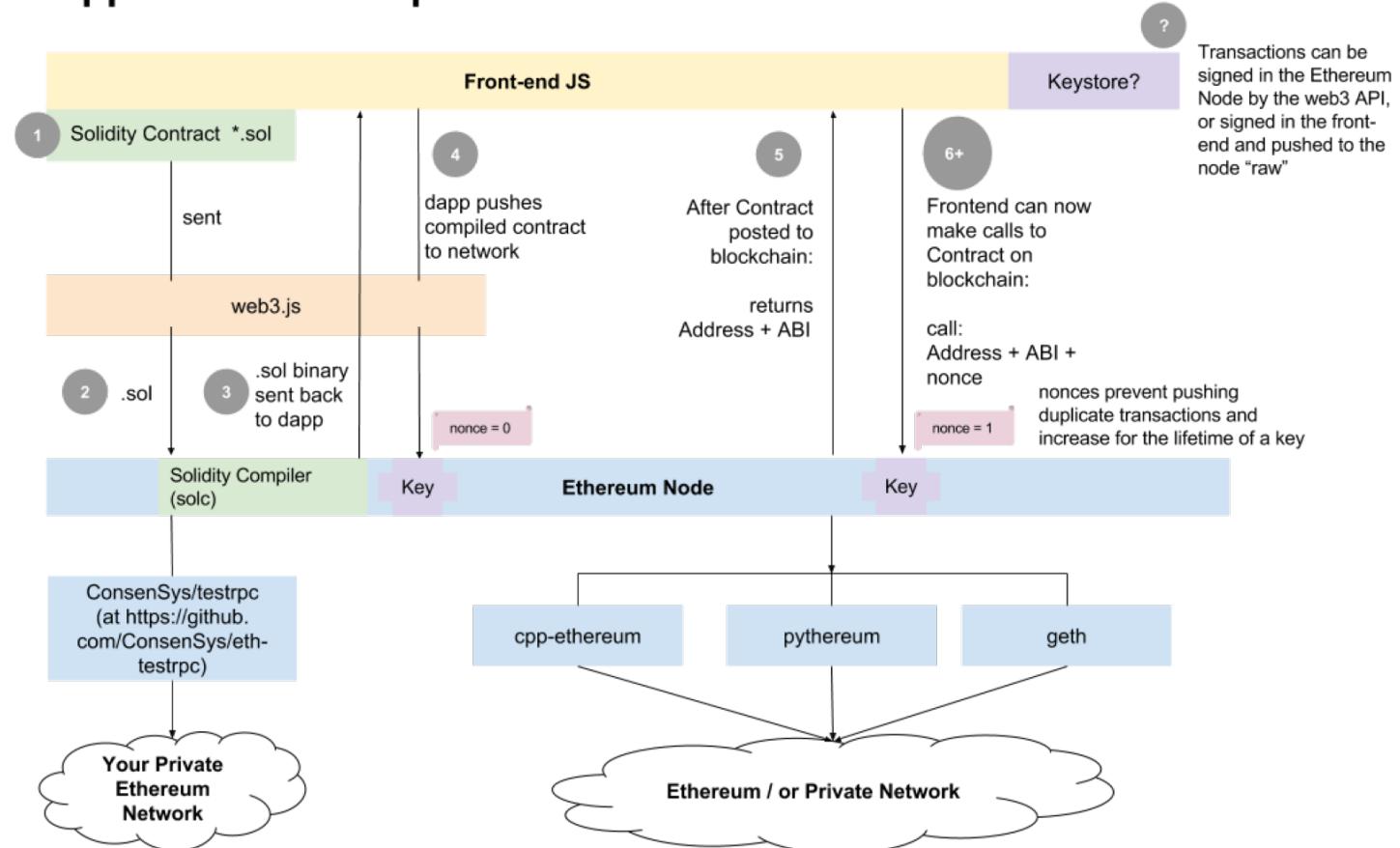
계정들은 이더를 보관하고 전송할 수 있지만 개별 입금되는 트랜잭션들을 보여줄 수 없습니다. 개별 입금 트랜잭션들을 보려면 [지갑 컨트랙트 설치하기](#)를 해야합니다.

If your balance doesn't seem updated, make sure that you are in sync with the network.

Solidity

Solidity 구조

dApp Front-end Steps



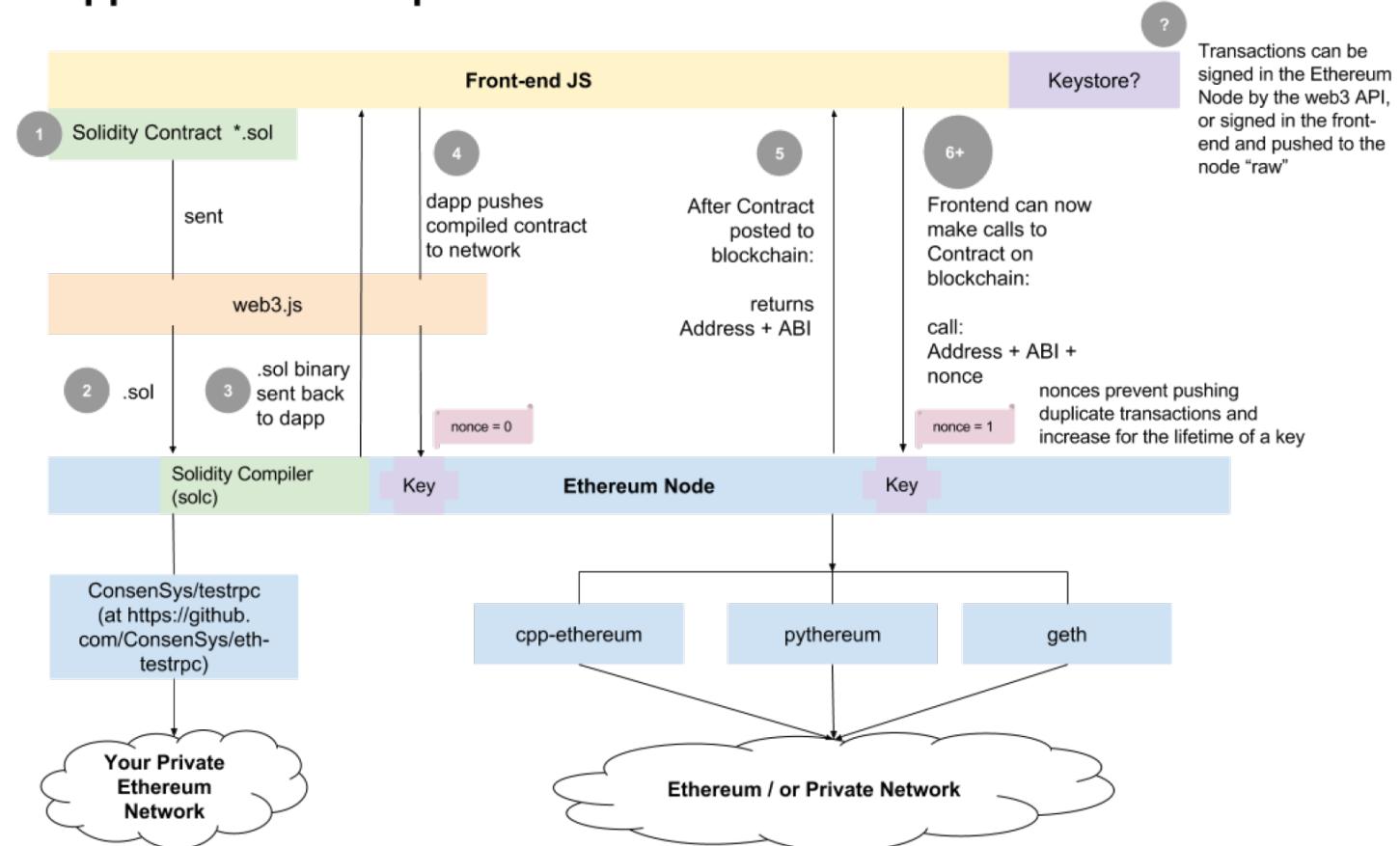
A **Contract Creation Transaction** is shown in steps 1-5 at above.

An **Ether Transfer or Function Call Transaction** is assumed in step 6.

Solidity 구조

dApp Front-end Steps

1. 스마트 컨트랙트 작성



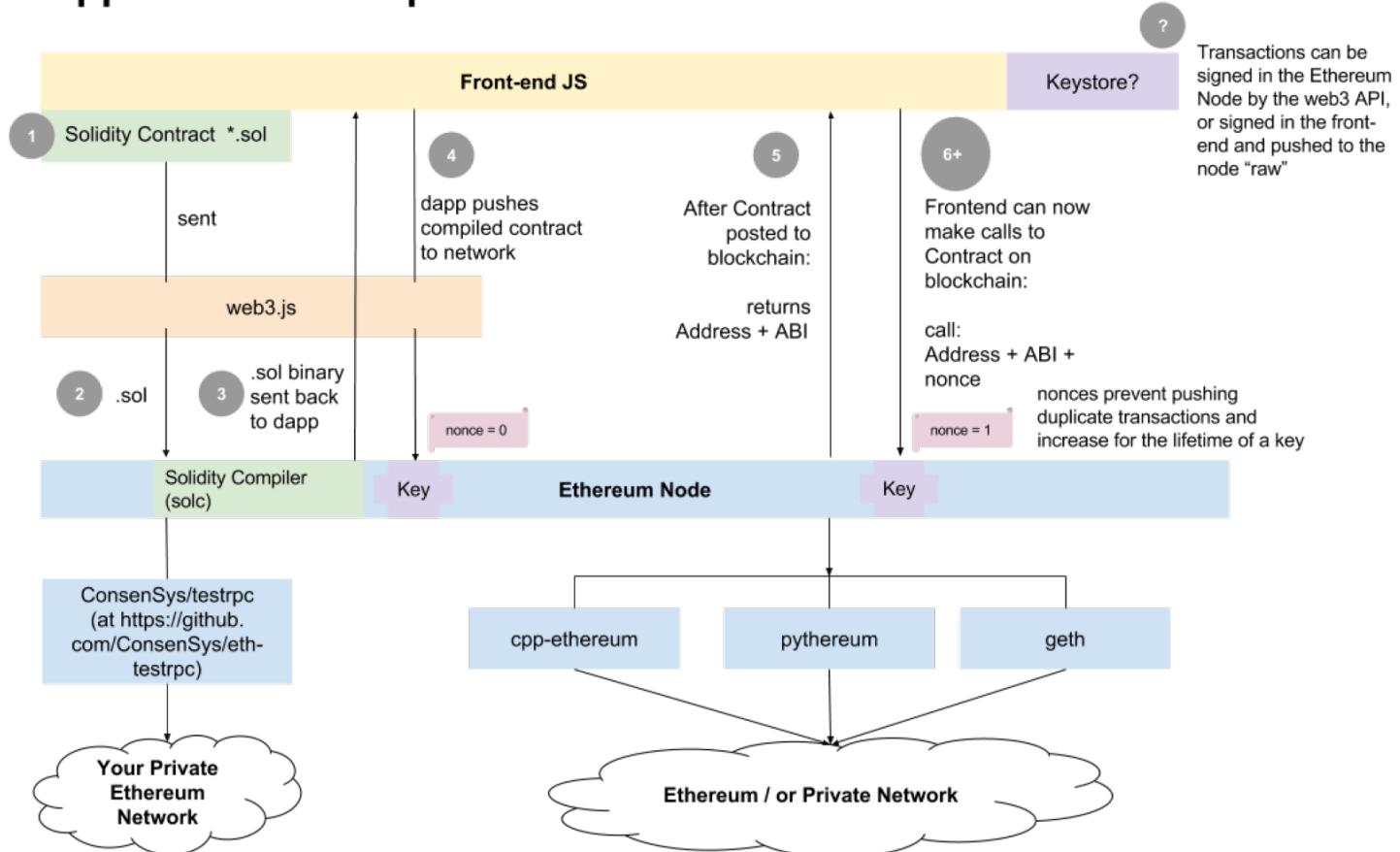
A **Contract Creation Transaction** is shown in steps 1-5 at above.

An **Ether Transfer or Function Call Transaction** is assumed in step 6.

Solidity 구조

dApp Front-end Steps

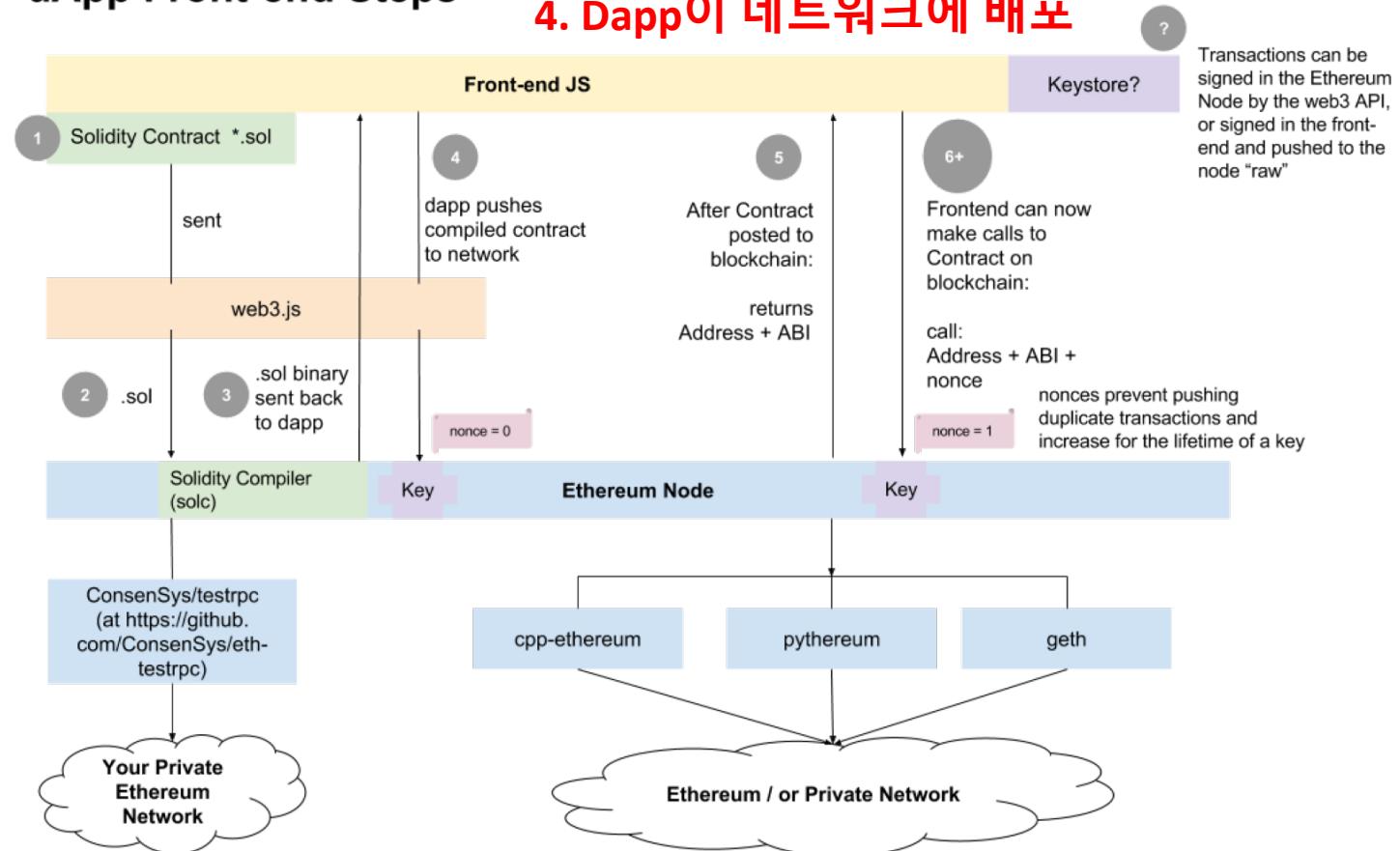
2 & 3. 컴파일



Solidity 구조

dApp Front-end Steps

4. Dapp이 네트워크에 배포

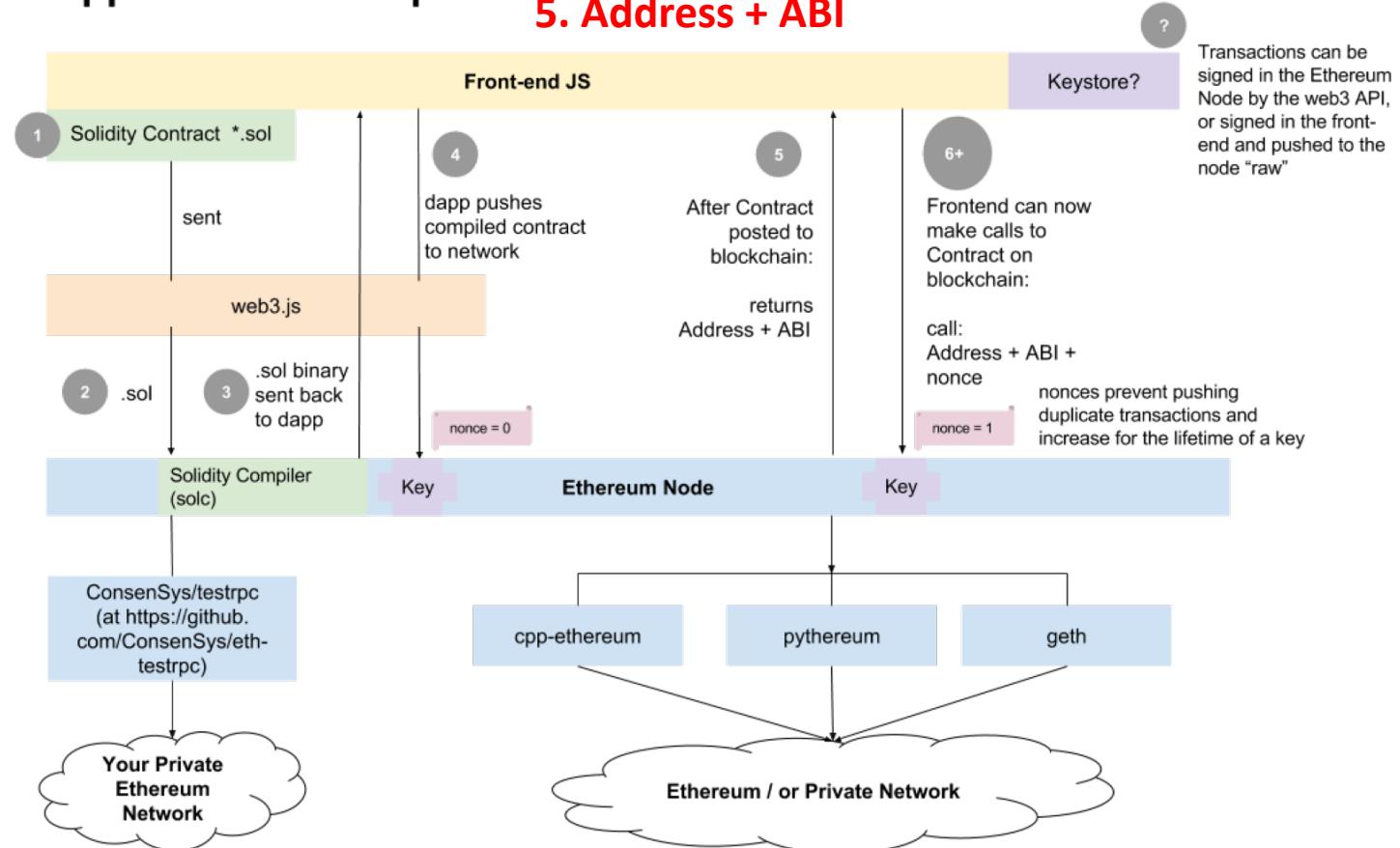


https://cdn-images-1.medium.com/max/1600/0*B_K9HNktRm_DrEZT.png

Solidity 구조

dApp Front-end Steps

5. Address + ABI



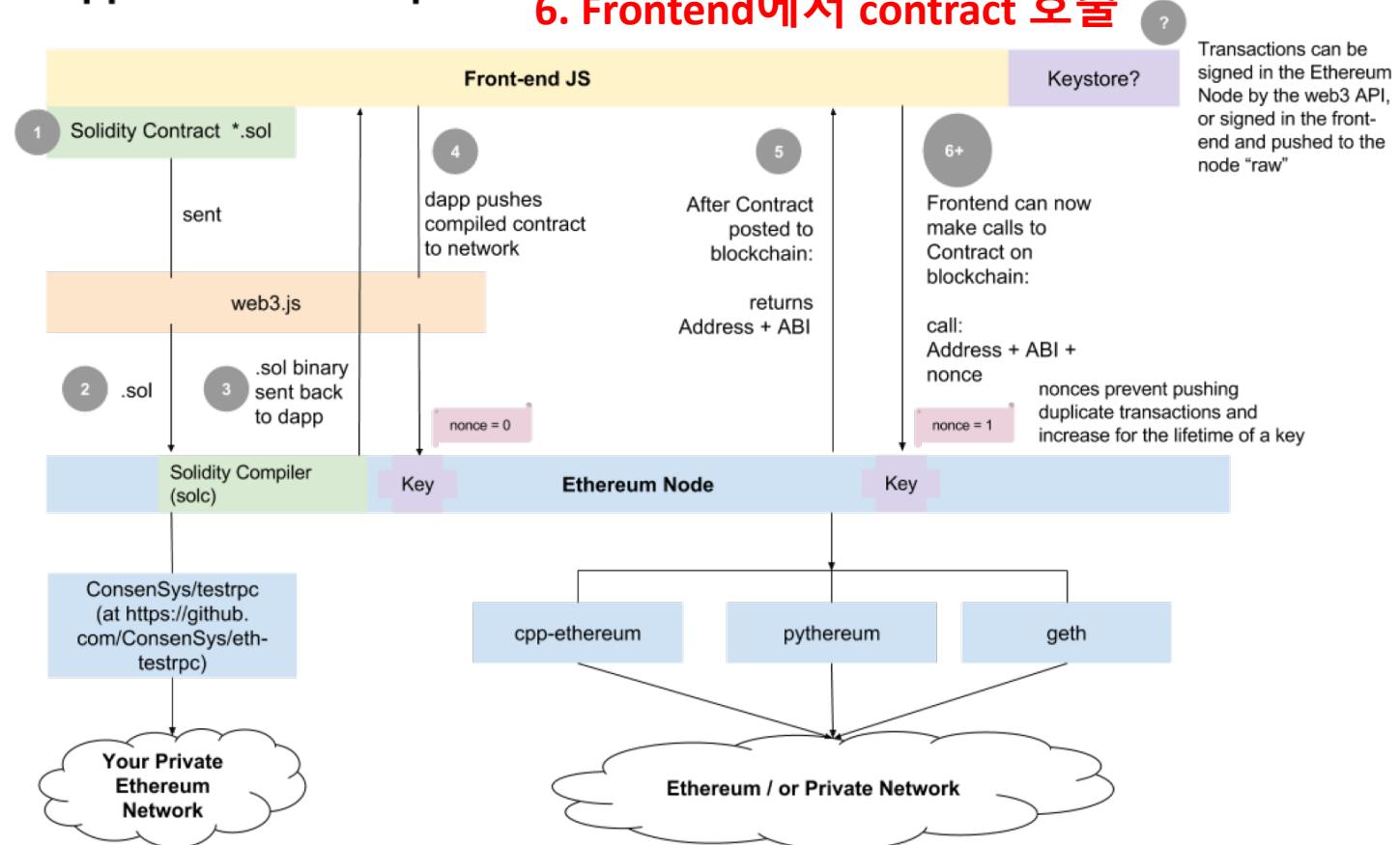
A **Contract Creation Transaction** is shown in steps 1-5 at above.

An **Ether Transfer or Function Call Transaction** is assumed in step 6.

Solidity 구조

dApp Front-end Steps

6. Frontend에서 contract 호출



A **Contract Creation Transaction** is shown in steps 1-5 at above.

An **Ether Transfer or Function Call Transaction** is assumed in step 6.

Solidity 구문

- 자료형

- uint – unsigned int
- address – Ethereum address
- bool, int(1~32)
- String
- No float in Solidity
- Solidity는 소수(float)를 지원하지 않음
 - 소수는 부정확 하기 때문

Solidity 구문

- State variables

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

- Function

```
pragma solidity ^0.4.0;  
contract SimpleAuction {  
    function bid() public payable { // Function // ...  
    }  
}
```

Solidity 구문

- Function modifier : 구문을 명확히 하는 데 사용

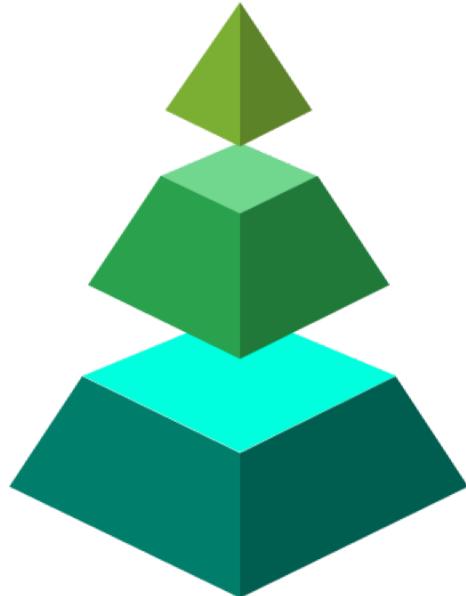
```
pragma solidity ^0.4.11;
contract Purchase {
    address public seller;
    modifier onlySeller() { //Modifier
        require(msg.sender == seller);
    }
    function abort() public onlySeller { // Modifier usage // ... }
}
```

Solidity 구문

- Events : EVM 로깅 기능과 연결되는 구문

```
pragma solidity ^0.4.0;
contract SimpleAuction {
    event HighestBidIncreased(address bidder, uint amount); // Event
    function bid() public payable { // ...
        HighestBidIncreased(msg.sender, msg.value); // Triggering event
    }
}
```

Solidity 자료형



- 1 Structs
- 2 Arrays
- 3 Mappings

- Mapping : 해쉬테이블
- Array : 컴파일 시 정적으로 크기가 결정되거나 동적 사이즈 설정 가능
- Struct : 다양한 타입을 속성으로 가지는 데이터 타입
 - 16개의 멤버 제한

Solidity 구문

- <https://github.com/shwarzes89/solidity101>

예제

ERC20 Specification

- Token Methods:
 - totalSupply
 - balanceOf
 - transfer
 - transferFrom
 - approve
 - allowance
- https://theethereum.wiki/w/index.php/ERC20_Token_Standard

Web3.js를 이용한 스마트 컨트랙트와 UI 연동

- Node.js 설치 (<https://nodejs.org/ko>)



Node.js®는 [Chrome V8 JavaScript 엔진](#)으로 빌드된 JavaScript 런타임입니다. Node.js는 이벤트 기반, 논 블로킹 I/O 모델을 사용해 가볍고 효율적입니다. Node.js의 패키지 생태계인 [npm](#)은 세계에서 가장 큰 오픈 소스 라이브러리 생태계이기도 합니다.

Spectre and Meltdown in the context of Node.js.

다운로드 macOS (x64)

8.10.0 LTS
안정적, 신뢰도 높음

9.8.0 현재 버전
최신 기능

[다른 운영 체제](#) | [변경사항](#) | [API 문서](#)

[다른 운영 체제](#) | [변경사항](#) | [API 문서](#)

LTS 일정은 [여기서 확인하세요](#).

Web3.js를 이용한 스마트 컨트랙트와 UI 연동

- 정상적으로 설치되었는지 확인
 - node -v
 - npm -v
- Ganache cli 설치
 - npm install ganache-cli

Web3.js를 이용한 스마트 컨트랙트와 UI 연동

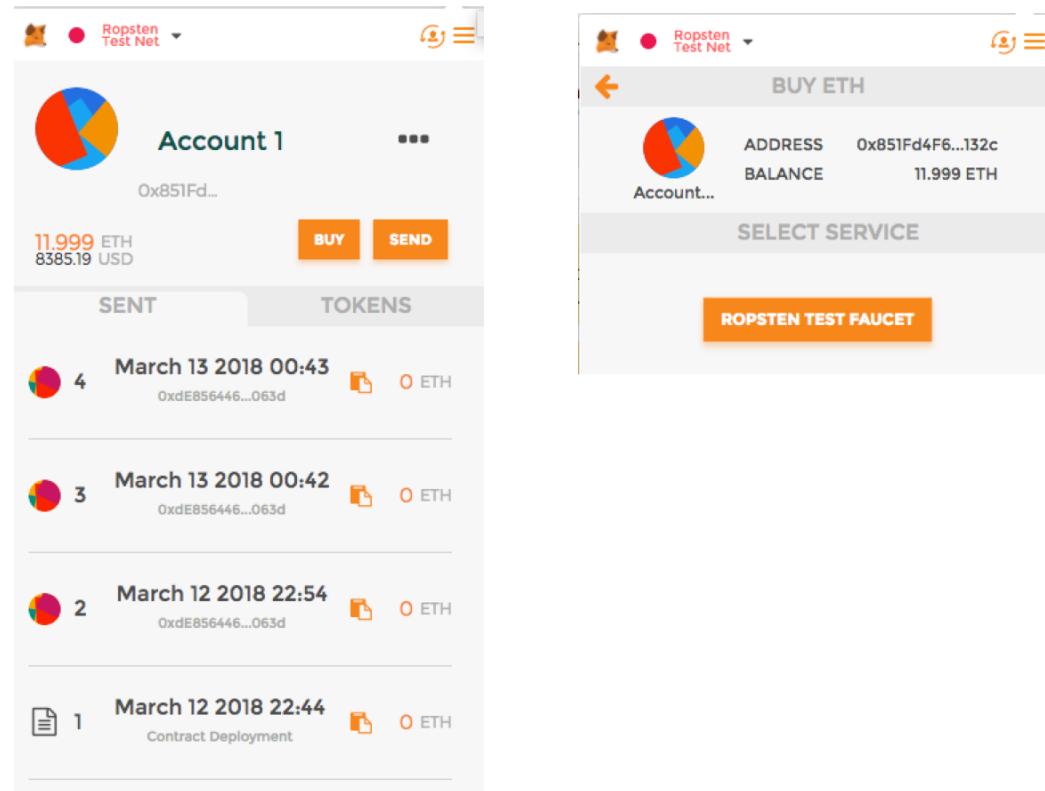
- Web3.js 설치
 - npm install ethereum/web3.js --save
- Ganache cli 설치
 - npm install ganache-cli

Web3.js를 이용한 스마트 컨트랙트와 UI 연동

- 샘플 스마트 컨트랙트 작성
 - <https://gist.github.com/shwarz89/0a275daefb2a190ce6a78a87db9563a4>
 - TODO 부분 실습

Web3.js를 이용한 스마트 컨트랙트와 UI 연동

1. MetaMask 실행
2. Ropsten network 연결
3. 'Buy' 클릭
4. 'Ropsten test faucet' 클릭



Web3.js를 이용한 스마트 컨트랙트와 UI 연동

5. 'Request 1 ether from faucet' - Ether 요청

The image displays three vertically stacked user interface components, each enclosed in a light gray border:

- Faucet Component:** Labeled "faucet" at the top. It shows the address 0x81b7e08f65bdf5648606c89998a9cc8164397647 and balance 9169234.49 ether. A green button labeled "request 1 ether from faucet" is present.
- User Component:** Labeled "user" at the top. It shows the address 0x851fd4f68dfc10027a81cb524e1bd482ae32132c and balance 12.00 ether. Below it is a "donate to faucet:" section with three orange buttons labeled "1 ether", "10 ether", and "100 ether".
- Transactions Component:** Labeled "transactions" at the top. This component is currently empty, showing a single row of three columns.

Web3.js를 이용한 스마트 컨트랙트와 UI 연동

- 스마트 컨트랙트 생성

The screenshot shows a user interface for interacting with a blockchain environment. At the top, there are configuration fields for the environment (set to 'Injected Web3' connected to 'Ropsten (3)'), account selection, gas limit (3000000), and value (0 wei). Below these are sections for creating a new contract ('Coursetro') and loading an existing one from an address. A summary at the bottom indicates 0 pending transactions and 0 contract instances.

Environment: Injected Web3 (Ropsten (3))

Account:

Gas limit: 3000000

Value: 0 wei

Coursetro

Create

Load contract from Address

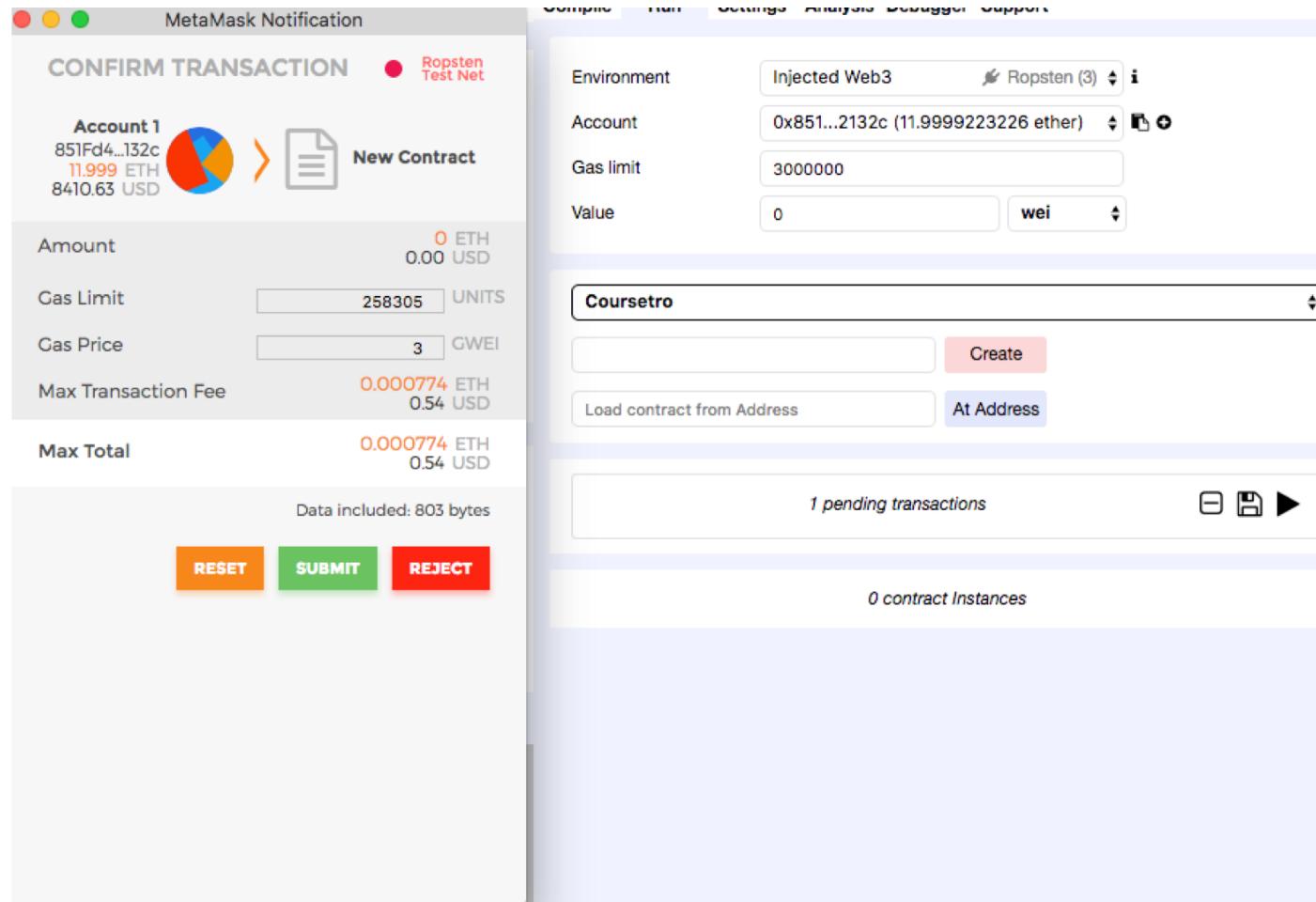
At Address

0 pending transactions

0 contract Instances

Web3.js를 이용한 스마트 컨트랙트와 UI 연동

- 스마트 컨트랙트 생성



Web3.js를 이용한 스마트 컨트랙트와 UI 연동

- Remix의 Compile 탭 – ‘Details’
- ABI 복사



Web3.js를 이용한 스마트 컨트랙트와 UI 연동

- Web page 작성
 - <https://gist.github.com/shwarzes89/469bfbf53356c03da04c24ccfb764b97>
 - TODO 부분 실습

Web3.js를 이용한 스마트 컨트랙트와 UI 연동2

- 실습
 - <https://gist.github.com/shwarz89/499abac6c86673efe5cd9ec1be54b92c>

Greeter

<https://gist.github.com/shwarzes89/fe25c9756b8506906906d45b6c4ecbe9>

안전거래 서비스

<https://gist.github.com/shwarz89/0526d0302e044496c618ebcbd9b493ec>

- value = 물건 가격
- seller = 판매자
- Buyer = 구매자
- State = 생성, 잠금, 물건 배송, 완료

ERC20 Specification

- <https://gist.github.com/shwarzes89/ea754daef5dd805aaa11bd98a308c8b>

Coin example

- 이더리움으로 살 수 있는 코인
- 1eth = 20 MyCoin
- 구매
- 환불

My Coin

<https://gist.github.com/shwarzes89/e8bcfffc8bd0d1f43535a99fe0aab94a>

My Coin

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

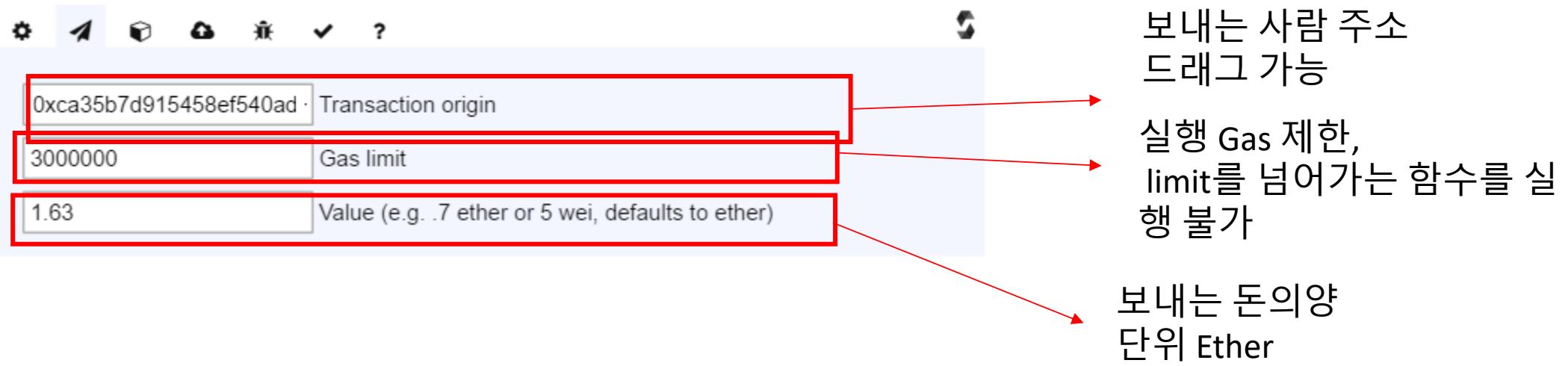
- 잔액 저장

```
uint COIN_WEI = 500000000000000000;
```

- 1Eth는 너무 크기 때문에 Ethereum에선 wei로 바꿔서 사용한다.
- 1 eth = 1000000000000000000 wei
- 1 eth = 20 coin
- 1 coin = 500000000000000000 wei

My Coin

트랜잭션 세팅하기



My Coin



컨트렉트 생성
같은 소스코드로 여러
컨트렉트를 생성 가능

A screenshot of a blockchain interface showing a list of functions for a specific contract instance. The title is "▼ sogangCoin at 0x610033b6dd5a08004e46f2097ca09b693d744118 (memory)". Below the title is a red button labeled "(fallback)". There are two tabs: "balances" (selected) and "address". Underneath are four red buttons labeled "buyToken", "confirmToken", and "sellToken", followed by the number "2". A blue icon with a question mark is next to the "balances" tab. A red arrow points from the "balances" tab to the Korean text "아이콘을 누르면 함수 실행, 파라미터는 옆에 기재".

아이콘을 누르면 함수 실행,
파라미터는 옆에 기재

안전 거래 서비스

- 스마트 컨트렉트는 돈을 보관한다.
- 물건이 확인되면 돈을 보낸다.

경매

<https://gist.github.com/shwarzes89/0c089125246ea814b19e98b93d0c4e96>

참조

- Solidity 문서
 - <https://solidity.readthedocs.io/en/v0.4.20>
- Solidity 샘플
 - <https://github.com/fivedigit/solidity-baby-steps>
 - <https://github.com/bellaj/smart-contract>