

Dapp 실습

박혜영

goblinok@gmail.com

목표

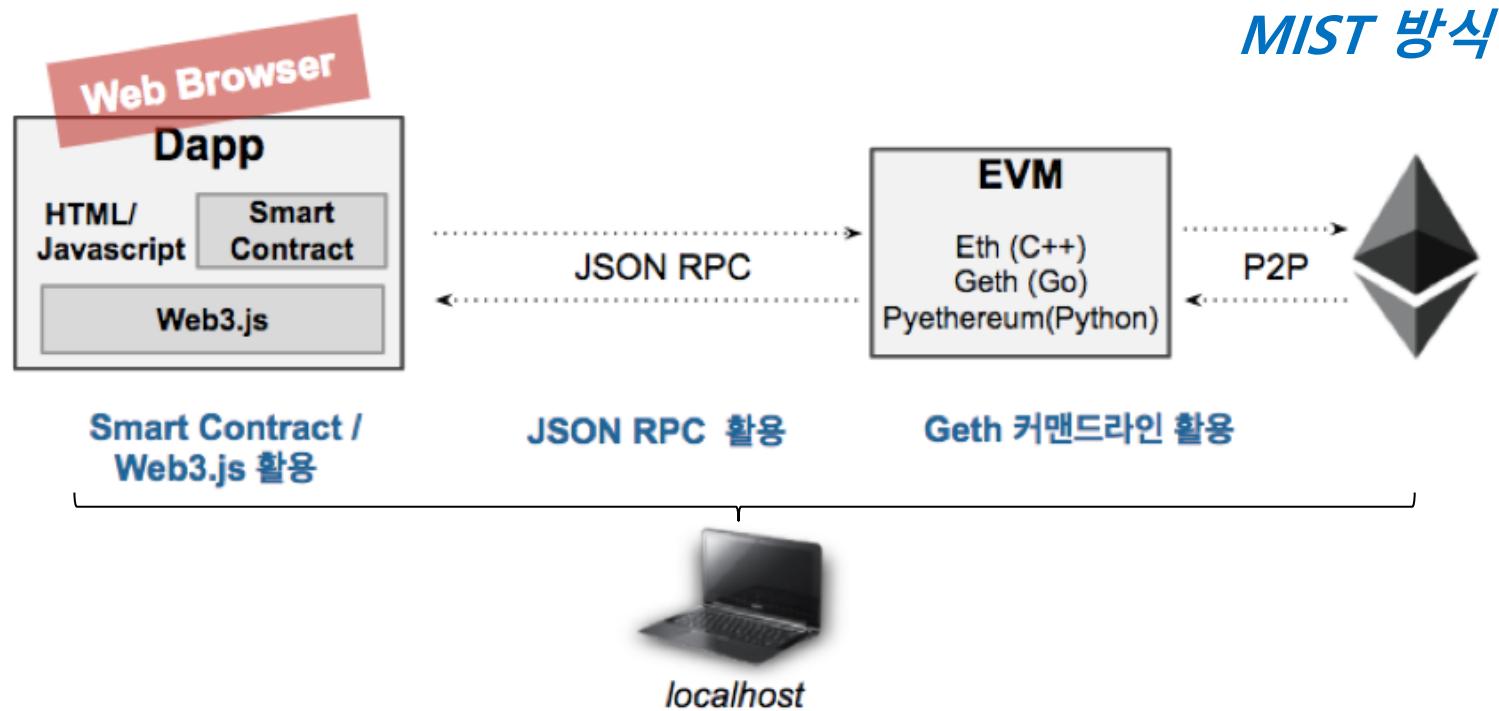
Geth의 실행부터 응용서비스 구현까지 ~

Geth 실습과 Solidity 실습을 통해 얻은 결과물을 바탕으로 실제 응용 서비스 구현

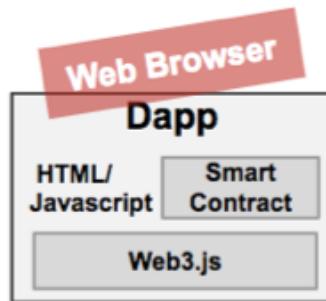


<http://etherstudy.net>

이더리움 응용 – Dapp(1/2)



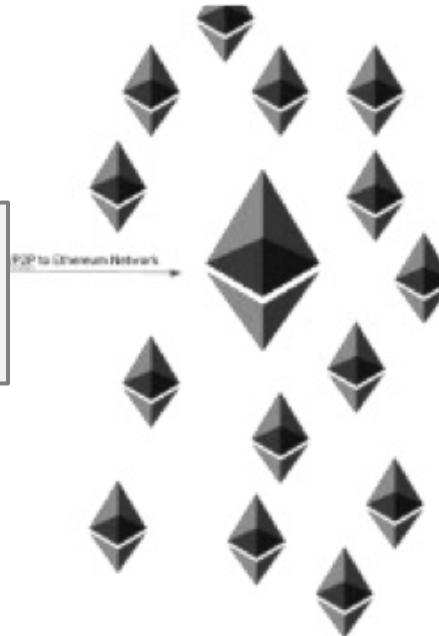
이더리움 응용 – Dapp(2/2)



JSON RPC



MetaMask 방식



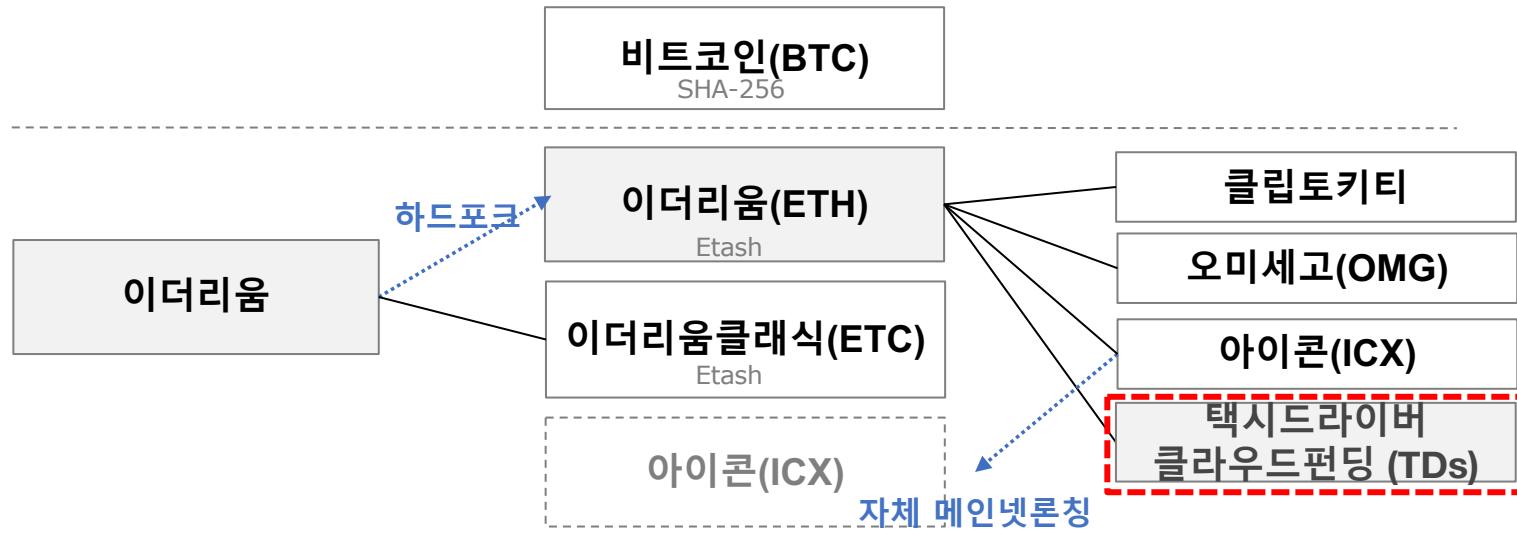


오늘 실습할 Dapp은?

플랫폼

메인넷(코인)

Dapp(토큰)



퀀텀

퀀텀(Qtum)

Proof of Stake

메디블럭(MED)

로빈8(PUT)

Step 1. '클라우드 펀딩' 사이트 예시

【참여자 화면】

<http://127.0.0.1:3000/users>

클라우드펀딩 메인



펀딩 모금액 : 124 ETH

보유 토큰양 : 376 TDs

참여하기



/users/new

클라우드 펀딩 참여하기

사용자 계정

참여 금액

비번

passphase

참여하기



/users/<address>

사용자 상세 정보

- 사용자 계정 : 0x7e684ab6c930e7d77078edf7053368cf3003e

- 이더 보유량 : 68589 061031466 ETH

- 토큰 보유량 : 624 TDs

【제작자 화면】

<http://127.0.0.1:3000/makers>

클라우드펀딩 메인



펀딩 모금액 : 124 ETH

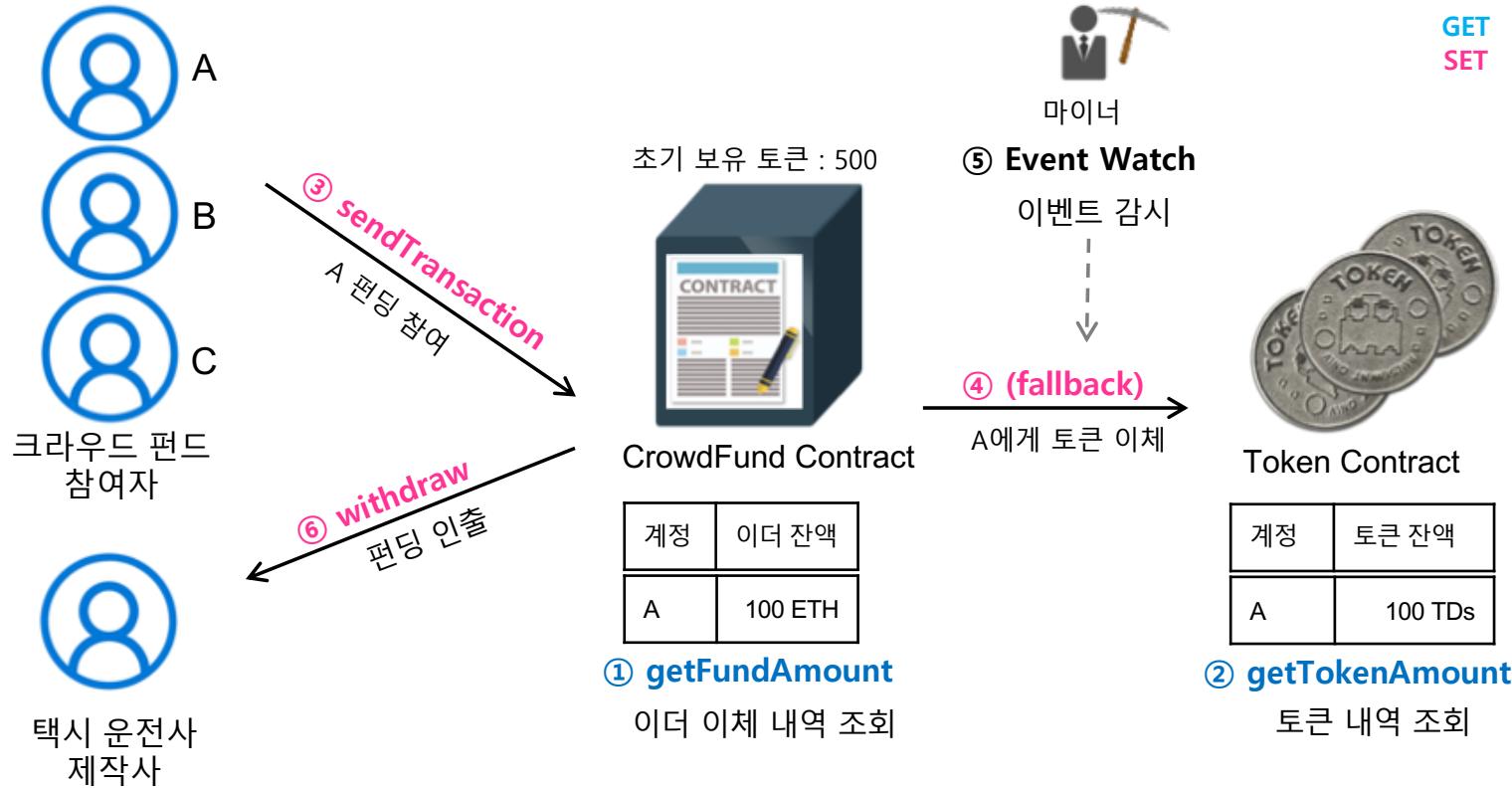


제작자 보유 이더량 : 0 ETH

모금액 인출하기



Step 1. '클라우드 펀딩' 시나리오



[참고] Contract Call 함수

Set	Get
블록체인에 데이터를 기록하는 것	블록체인의 데이터를 가져오는 것
Transaction을 보낸다.	Transaction이 발생 X
Value(ETH), Gas 필요	Value(ETH), Gas 필요하지 않다.
Hex string 값으로 저장한다.	Hex string 값을 읽어온다.
Transaction이 Mining되었다는 callback 함수를 지원하지 않는다.	get 함수를 호출하면 callback 함수로 데이터를 받아올 수 있다.
전송한 Transaction이 Mining되었나를 확인하는 코드 작성 필요한다.	callback 함수로 데이터를 바로 받아온다.
복잡하거나 큰 데이터 등을 저장하기 위해서는 swarm과 같은 p2p 스토리지를 사용하는 것이 좋다.	

Step 2. 실습 환경 준비 – 설치

0 Geth 구동

```
$ geth --datadir "./ether_study" --networkid 15 --port 8080 --rpc --rpccorsdomain "*" --rpcapi "admin,db,eth,miner,net,txpool,personal,web3" console  
> (프롬프트)
```

1 \$ node -v <https://nodejs.org/ko/download/>

2 \$ npm -v

3 \$ npm install express-generator -g

git clone https://github.com/etherstudy/crowdfund_exercise.git

4 \$ express crowdfund

5 \$ cd crowdfund;

6 /crowdfund \$ npm install ethereum/web3.js --save

7 /crowdfund\$ npm install

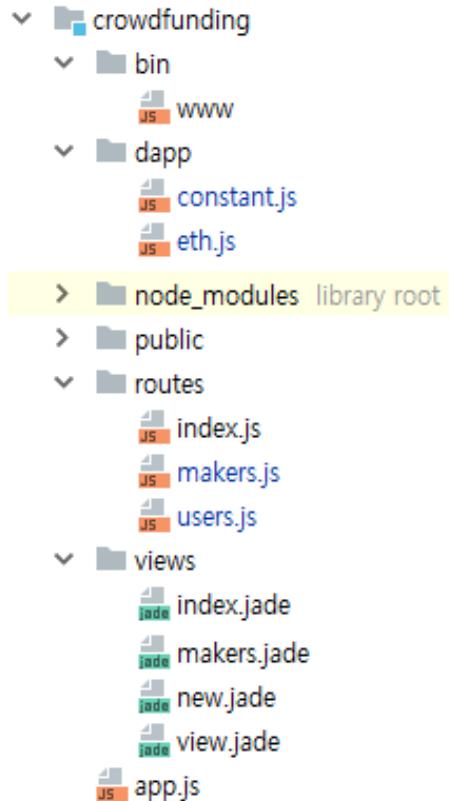
/crowdfund\$ npm start ←구동시작

- node.js : v9.2.0
- npm : 5.6.0
- Express : v4.15.5
- web3.js : 0.19.0
- Solidity 0.4.16
- Geth v1.7.0
- os : mac os / window

정상 동작 확인 <http://127.0.0.1:3000>



[참고] 소스 디렉토리 구조



/bin/www: 서버 구동을 위한 코드가 기록되어 있습니다. 익스프레스 서버설정 코드가 기록된 app.js 파일을 가져와 노드의 HTTP 객체와 연동하는 작업이 이루어진다. 디폴트 포트 3000이 설정되어 있다.

/dapp: web3와 컨트랙트 관련 라이브러리

/node_modules : 필요한 모듈 설치 폴더. package.json 파일을 열고 dependencies 값을 이용해 의존성 모듈을 다운로드하여 node_modules 폴더에 저장

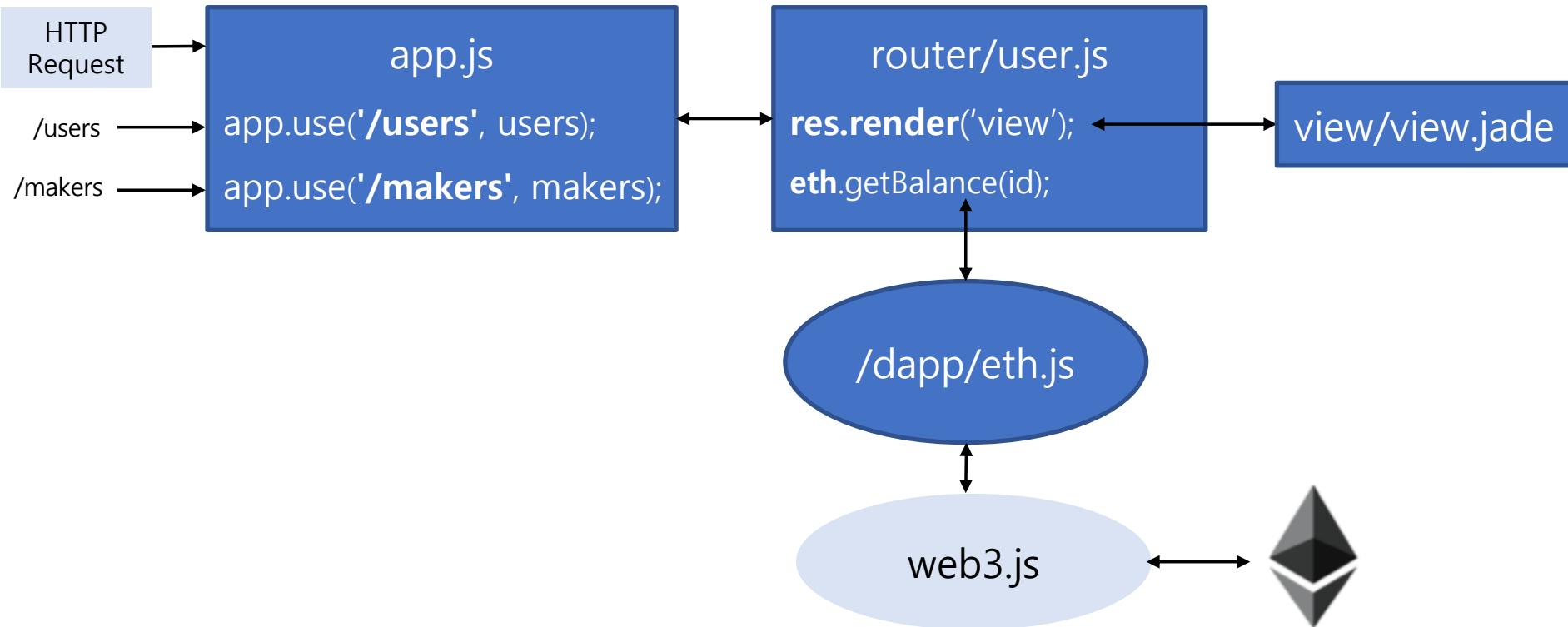
/public: 정적 파일을 위한 폴더로서 자바스크립트 파일, 이미지 파일, 스타일시트 등

/routes: 라우팅을 위한 폴더. 라우팅 리소스 별로 모듈을 만들어 라우팅 로직을 구현

/views: 템플릿 Jade 파일. 라우팅 로직 중 render() 함수에서 사용한다.

app.js: /bin/www 에서 사용되며 익스프레스 설정이 담겨 있는 파일

[참고] 실행 플로우



Step 3. Web3 연동

Web3 설치 및 연동

- web3 모듈 설치 : npm install ethereum/web3 –save
- Localhost에 http로 연결

```
//-----
// Web3 연결
//-----
var Web3 = require('web3');
var web3 = new Web3(new Web3.providers.HttpProvider("http://127.0.0.1:8545"));
```

Step 3. Contract 연동

Contract 연동

- 필요한 사항 : ABI, Contract Address
- ABI 와 Contract Address 로 contract를 호출한다.

```
//-----
// 스마트 컨트랙트 연결
//-----
// crowdFund Contract
var walletTokenAbi = [ <ABI 입력>];
var TokenContract = web3.eth.contract(walletTokenAbi).at(<TokenAddress 입력>);

// crowdFund Contract
var crowdFundAbi = [ <ABI 입력>];
var CrowdFundContract = web3.eth.contract(crowdFundAbi).at(< CrowdFundAddress 입력>)
```

[참고] 토큰 컨트랙트 생성 방법

The screenshot shows the Remix IDE interface with the following components:

- Code Editor:** On the left, the file `browser/WalletCompatibleToken.sol` contains Solidity code for a token contract. A red arrow points from the `Create` button in the center to the `balanceOf` function in the code.
- Environment:** On the right, the environment settings show a Web3 Provider set to "Custom (15)", an Account set to `0xf3b...9a812 (100.1 ether)`, a Gas limit of 3000000, and a Value of 0 wei.
- Contract List:** Below the environment, the `WalletCompatibleToken` contract is listed with its constructor parameters: `"Crowd Fund Token", "CFTKs", 0, 1000`. A red box highlights this row, and a red arrow points from it to the `CA` button at the bottom.
- ABI:** A red box highlights the `ABI` section, which lists the four functions: `0:`, `1:`, `2:`, and `3:`.
- Logs:** At the bottom, it shows "0 pending transactions".
- Bottom Buttons:** The `CA` button is highlighted with a red box, and the text "token at 0x" followed by a partial address is displayed next to it.

The bottom status bar displays the text: `" TaxiDriver Token ", " TDs", 0, 1000`.

[참고] 클라우드펀딩 컨트랙트 생성 방법

The screenshot shows the Remix IDE interface with the following details:

- Left Panel (Code View):** Displays the Solidity code for the `CrowdFund` contract. The code defines a constructor that initializes variables like `beneficiary`, `fundingGoal`, `deadline`, `price`, and `tokenReward`. It also includes a payable function that updates the balance of the sender.
- Top Bar:** Shows tabs for `browser/WalletCompatibleToken.sol` and `browser/CrowdFund.sol`. The `CrowdFund.sol` tab is active.
- Right Panel (Toolbars and Settings):**
 - Environment:** Set to `Web3 Provider` with `Custom (15)`.
 - Account:** Set to `0xf3b...9a812 (190.1 ether)`.
 - Gas limit:** Set to `3000000`.
 - Value:** Set to `wei`.
- Middle Panel (Contract Interaction):**
 - CrowdFund:** A dropdown menu showing the address `0xef77e0895435f4324cc724cb454fc14d06bc392e`. A red box highlights this address.
 - Create:** A pink button next to the address.
 - Load contract from Address:** A field with the placeholder `At Address`.
- Bottom Panel (Transactions):**
 - 2 pending transactions** (indicated by a red box).
 - CA CrowdFund at 0x550...9f326 (blockchain)** (also indicated by a red box).

"0xef77e0895435f4324cc724cb454fc14d06bc392e", 200, 10, 1, "0xfeecab0f15c2a769b3f1658e2b956e6236d4600b"



생성된 Contract는 블록에서 어떻게 확인할까?

>eth.getTransaction(txhash)

>eth.getTransactionReceipt(txhash)

>eth.getBlock(14)

txhash:0xfe0ab5a80f015c865f9a47b332172fe46cf218cdcf8d2c4261e9012b57d0642



생성된 Contract는 블록에서 어떻게 확인할까?

Geth Console

```
> admin.peersINFO [01-29|17:01:14] Submitted contract creation fullhash=0xfe0ab5a80f  
015c865f9a47b332172fe46cfa218cdcf8d2c4261e9012b57d0642 contract=0x1179d5Ef2233f428c12A1a2247b267B  
305aAD19e
```

txhash
contractAddress

1. eth.getTransaction(txhash)

```
> eth.getTransaction("0xfe0ab5a80f015c865f9a47b332  
{  
  blockHash: "0x308aef74c11d1bae9f64859de35b2c823d  
  blockNumber: 14,  
  from: "0xf3bef488f92f2774d8b50c41efe9a61a51e9a81  
  gas: 838457,  
  gasPrice: 18000000000,  
  hash: "0xfe0ab5a80f015c865f9a47b332172fe46cfa218  
  input: "0x60606040526000600760006101000a81548160
```

blockNumber

2. eth.getTransactionReceipt(txhash)

```
> eth.getTransactionReceipt("0xfe0ab5a80f015c865  
{  
  blockHash: "0x308aef74c11d1bae9f64859de35b2c82  
  blockNumber: 14,  
  contractAddress: "0x1179d5Ef2233f428c12A1a2247  
  cumulativeGasUsed: 838457,  
  from: "0xf3bef488f92f2774d8b50c41efe9a61a51e9a
```

contractAddress

3. eth.getBlock(14)

```
> eth.getBlock(14)  
{  
  difficulty: 182119,  
  extraData: "0xd88301080084676574  
  gasLimit: 3142635,  
  gasUsed: 838457,  
  hash: "0x308aef74c11d1bae9f64859de35b2c823d
```

Step 4. Get

Contract Get

- <컨트랙트>.<함수명> 으로 컨트랙트의 Public 함수를 호출할 수 있다.
- 컨트랙트에서 Public으로 선언된 balanceOf를 읽어온다

```
//-----  
// 토큰 장부 조회 (balanceOf)  
//-----  
exports.getTokenAmount = function (address) {  
    return TokenContract.balanceOf(address);  
};
```

```
//-----  
// 펀딩 장부 조회 (balanceOf)  
//-----  
exports.getFundAmount = function (address) {  
    return CrowdFundContract.balanceOf(address);  
};
```

<WalletCompatible.sol>

```
contract WalletCompatibleToken {  
    string public name;  
    string public symbol;  
    uint8 public decimals;  
  
    mapping (address => uint256) public balanceOf;
```

<CrowdFunding.sol>

```
contract CrowdFund {  
    token public tokenReward;  
    uint public price;  
    mapping(address => uint256) public balanceOf;
```

Step 4. SET

SendTransaction

<https://web3js.readthedocs.io/en/1.0/web3-eth.html#sendtransaction>

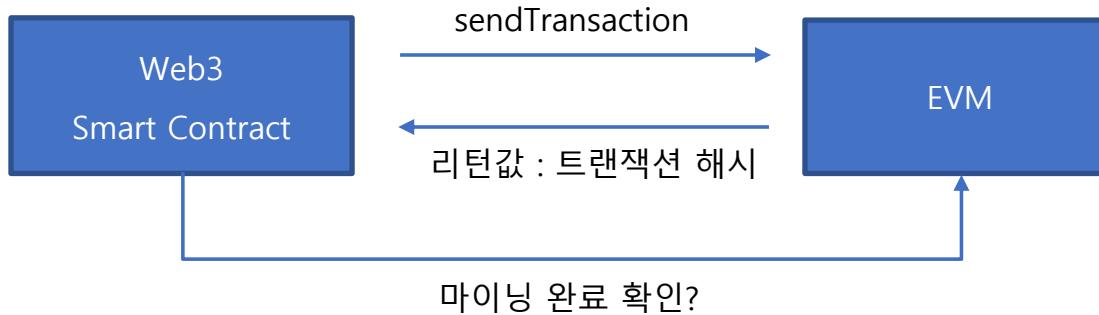
함수명 **web3.eth.sendTransaction(Transaction Object [, callback])**

from	보내는 wallet의 주소
to	받는 wallet의 주소
value	보낼 이더의 양(hex)
gas	지불할 가스의 양(hex)

리턴 32Byte의 hex값, Transaction의 Id

```
//-----  
// 이더 트랜잭션 수행  
//-----  
exports.sendTransaction =  
function(from,to,value,gas,callback) {  
  
    web3.eth.sendTransaction({  
        to: to,  
        from: from,  
        value: web3.toWei(value,'ether'),  
        gas: 100000}, function (err, hash) {  
        if (err) {  
            return callback(err, "");  
        } else {  
            return callback(null, hash);  
        }  
    });  
};
```

[참고] Set 완료 확인



- ① getTransaction 결과를 주기적으로 풀링하여 마이닝 완료를 체크
- ② Contract Event Watch 로 마이닝 완료를 감지

[참고] Set 완료 확인

getTransaction 폴링

<https://web3js.readthedocs.io/en/1.0/web3-eth.html#gettransaction>

- sendTransaction의 리턴값인 Transaction Id로 getTransaction 함수를 호출한다.
- getTransaction 결과, 블록넘버가 null 이면 해당 트랜잭션이 아직 마이닝이 완료되지 않다는 뜻이다.
- Timer를 실행하여 주기적으로 트랜잭션이 마이닝이 되었는지 확인한다.

함수명 web3.eth.getTransaction(transactionId [, callback])

주요
파라미터

Transaction ID sendTransaction으로 받은 리턴값

리턴 트랜잭션의 정보를 담은 객체

```
exports.checkTransaction = function (id, callback) {  
  if (web3.eth.getTransaction(id) !== null) {  
    console.log('mining done');  
    return callback(null, id);  
  }  
  
  setTimeout(function(){  
    if (web3.eth.getTransaction(id) !== null) {  
      clearTimeout(timer);  
      return callback(id);  
    }  
  }, 3000);  
  
  this.checkTransaction(id,callback);  
};
```

Step 5. Event Watch

Event Watch

- Contract에서 event 형으로 선언한 함수가 호출되면 EVM에서 Event가 발생한다.
- event 형으로 선언한 함수는 전부 호출을 감지할 수 있다. 이를 콜백처럼 활용 가능하다.

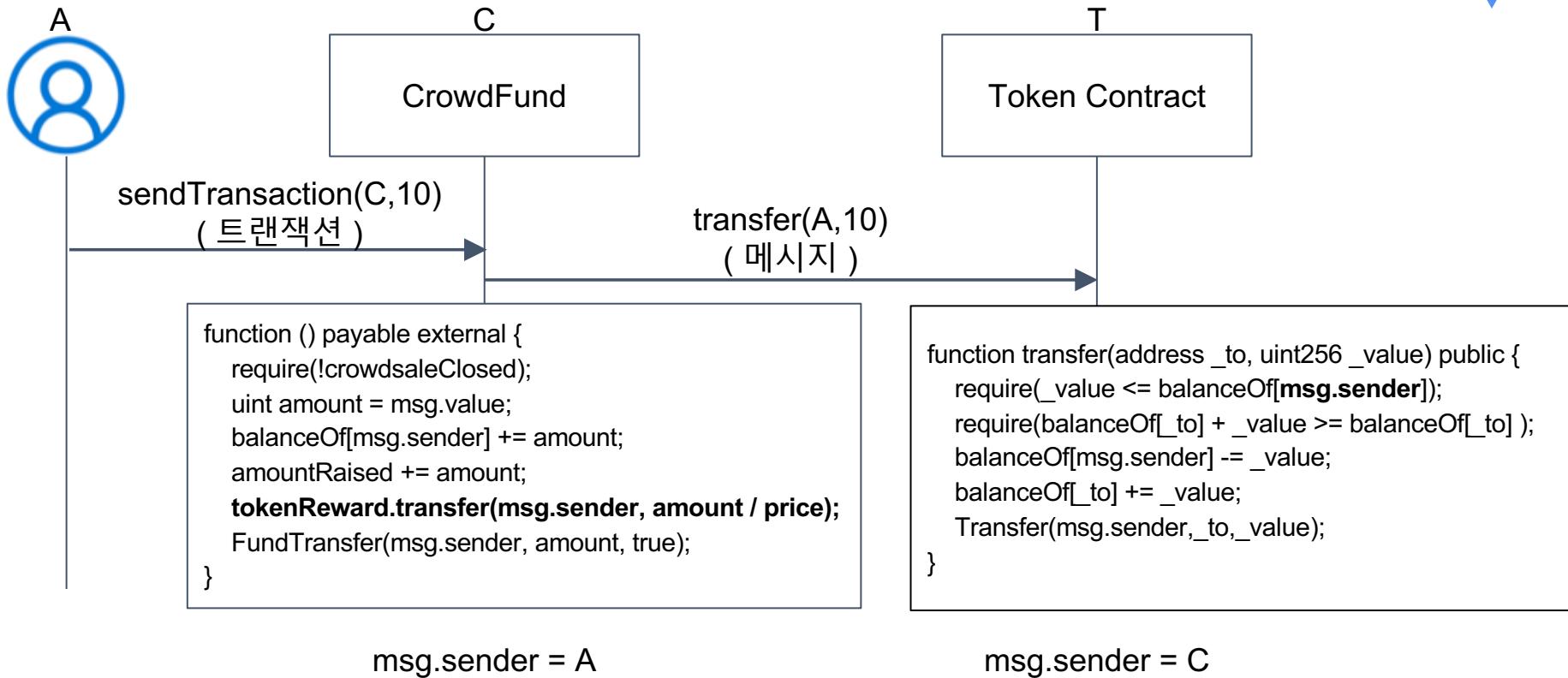
```
//-----  
// 이벤트 모니터링  
//-----
```

```
exports.fundTransferEvent = function( callback ) {  
    CrowdFundContract.FundTransfer().watch(function(error, res){  
        if (error) {  
            return callback(err, "");  
        } else {  
            return callback(null, res);  
        }  
    });  
};
```

<CrowdFunding.sol>

```
event FundTransfer(address backer, uint amount, bool  
function () payable {  
    uint amount = msg.value;  
    balanceOf[msg.sender] += amount;  
    tokenReward.transfer(msg.sender, amount / price);  
    FundTransfer(msg.sender, amount, true);  
}
```

[참고] CrowdFund 로 송금



Q&A

