이론부터 배포까지

단계별로 익히는 이더리움 ĐApp 개발

박지수(jisupark@sooho.io)

sooho.

0 OUTLINE

발표 내용 소개

1

BLOCKCHAIN FUNDAMENTALS

- 비트코인
- 스마트 컨트랙트
- 이더리움
- EVM

2

SOLIDITY

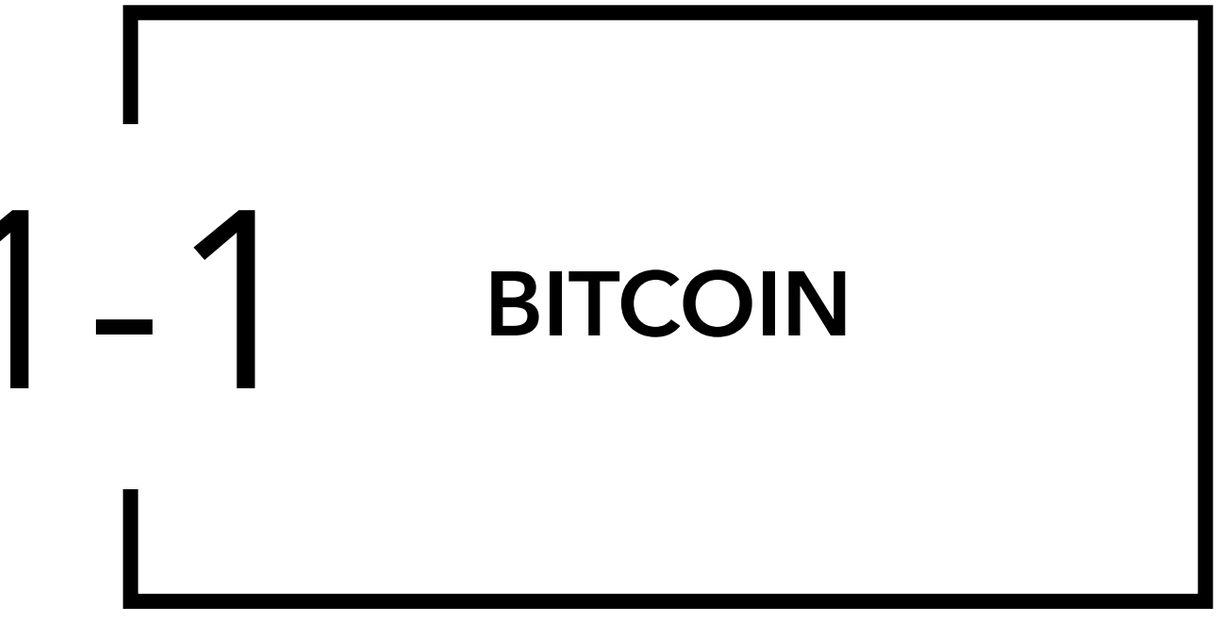
- 컨트랙트 구조
- 상속과 접근 제어
- 데이터 저장소
- 배열, 구조체, 매핑

3

DEVELOP

- 개발 환경 구성
- 개발 실습

1 BLOCKCHAIN FUNDAMENTALS

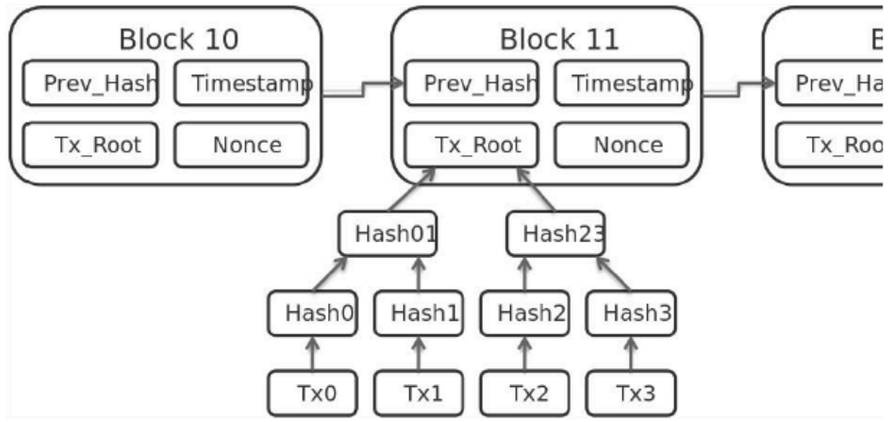

1-1
BITCOIN

1 BITCOIN
비트코인

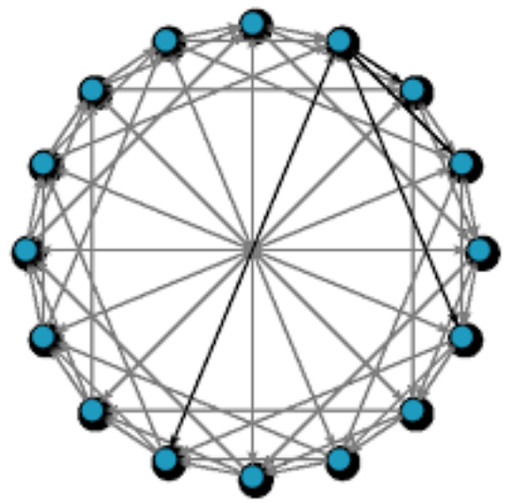
"Then, gentlemen, it is the consensus of this meeting that we say nothing, do nothing, and hope it all blows over before our next meeting."

컨센서스 알고리즘

암호화 알고리즘



블록체인



분산 시스템

1 BITCOIN
비트코인

✓ Peer-to-Peer 기반 기술

✓ 거래의 익명성과 투명성 보장

✓ 이중 거래, 거래 부정 방지

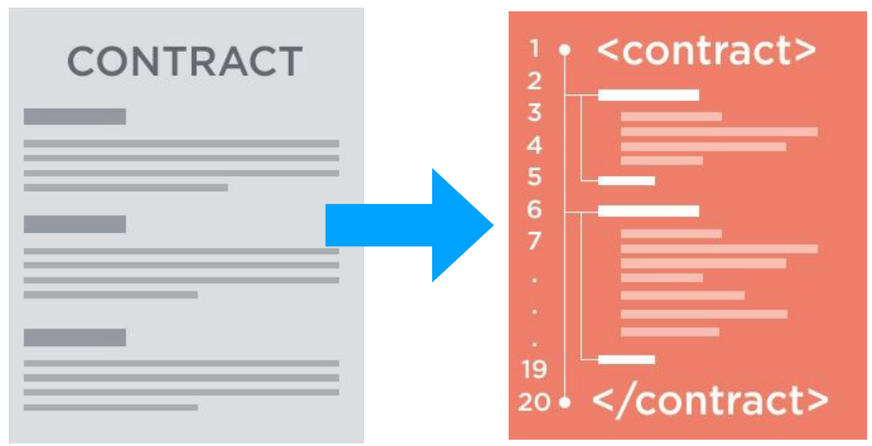
✓ 중앙통제 기구로부터 자유

1-2 SMART CONTRACT

2 SMART CONTRACT
스마트 컨트랙트

1. 개념 제안
- Szabo, Nick. "Smart Contracts" (1994).
- Szabo, Nick. "Formalizing and securing relationships on public networks." First Monday 2.9 (1997).

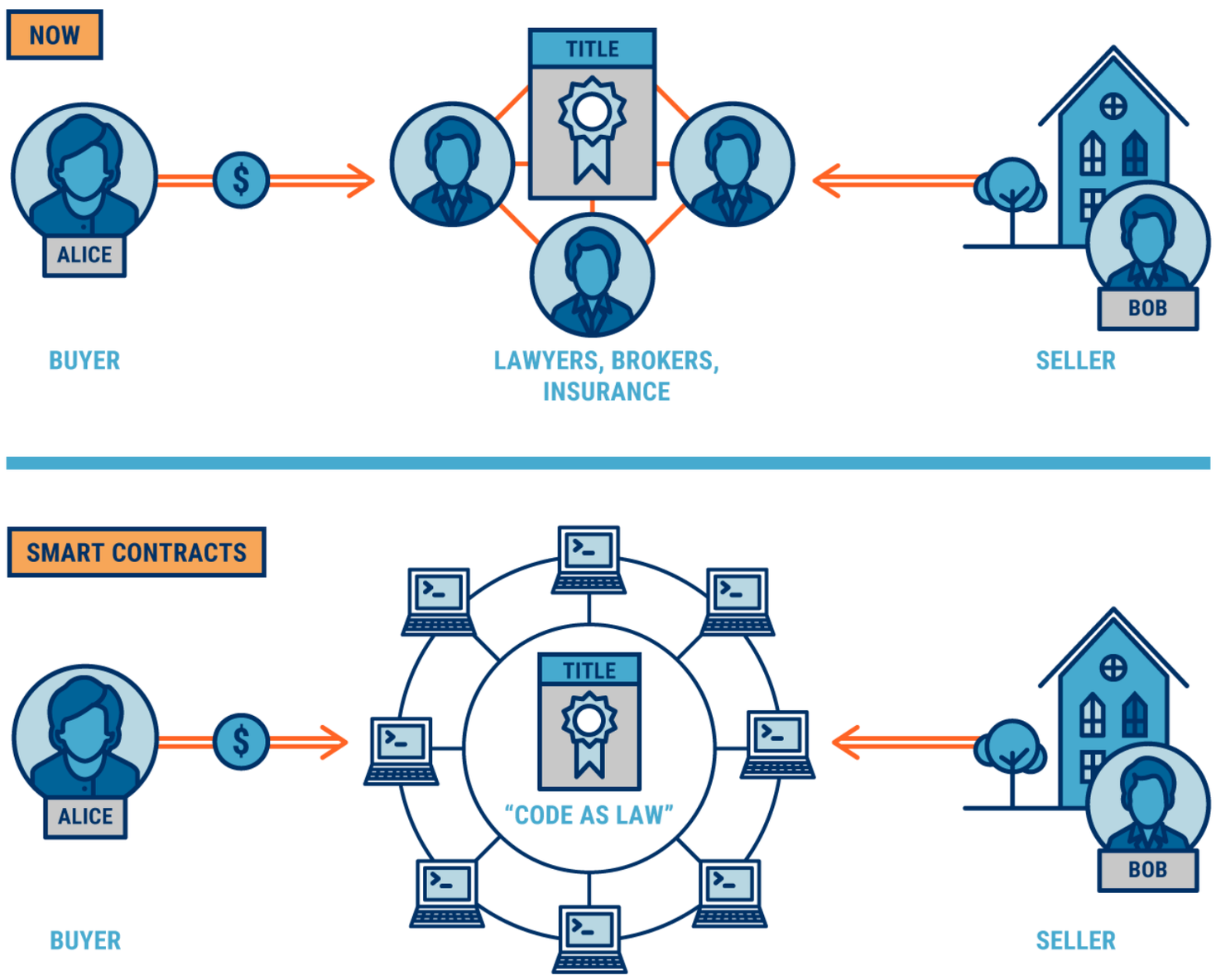
2. 핵심 아이디어
- **디지털 방식**으로 계약이 가능하고, 검증과 수행할 수 있는 프로토콜
- 제 3자 개입없이 **믿을 수 있는 거래**가 가능한 프로토콜
- 거래는 **트래킹 가능**하며 **돌이킬 수 없는** 프로토콜



3. 구현체
- 비트코인: 일정 부분 제한된 스마트 컨트랙 플랫폼
- 이더리움: 스마트 컨트랙 플랫폼

2 SMART CONTRACT
스마트 컨트랙트

Ex) 이더리움 상에서의 집 구매



1-3 ETHEREUM

3 ETHEREUM
이더리움

1. 스마트 컨트랙트를 실행시키기 위해 설계된 탈중앙화 플랫폼

- Account 기반 블록체인
- 분산된 State Machine
  - Transaction이 블록체인 네트워크의 전체 상태를 변경한다
  - 즉, transaction은 일종의 전이 함수

2. 이더라는 고유 화폐가 존재한다

- 이더리움 생태계를 지탱하는 화폐
- 생태계 참여자, 채굴자들의 보상을 위해 존재

3 ETHEREUM
이더리움

BITCOIN

- 화폐를 위한 블록체인
- 재화: 비트코인
  - Bitcoin 블록 체인의 주요 목적
- 단순하지만 견고함
- 튜링 완전하지 않은 원시 스크립트 언어
- UTXO 기반
- Proof-of-Work

ETHEREUM

- 스마트 컨트랙 블록체인 플랫폼
- 재화: 이더
  - 인센티브에 사용되는 보조 목적
- 복잡하고 기능이 풍부한
- 튜링 완전한 스크립트 언어
- Account 기반
- Proof-of-Stake로 전환 예정

3 ETHEREUM
이더리움

BITCOIN
UTXO

3 BTC => A

5 BTC => A

1 BTC => A

ETHEREUM
ACCOUNT

Address: “0xcebd”
Balance: 40 ETH
Code: c := a + b

Address: “0xf3bd”
Balance: 3 ETH
Code: c := a + b

3 ETHEREUM
이더리움

BITCOIN
UTXO

3 BTC => A

5 BTC => A

1 BTC => A

ETHEREUM
ACCOUNT

Address: “0xcebd”
Balance: 40 ETH
Code: c := a + b

Address: “0xf3bd”
Balance: 3 ETH
Code: c := a + b

트랜잭션 생성 용이, 이중지불 방지 등

UTXO를 모두 저장하지 않아도 됨, 개발자 친화적

3 ETHEREUM ACCOUNTS

이더리움 account 종류

Externally Owned Accounts

- 단체나 사람, 기관 등이 소유
- 이더를 전송하거나 컨트랙을 실행하는 트랜잭션 생성 가능
- 주소와 이더를 가지고 있음

Contract Accounts

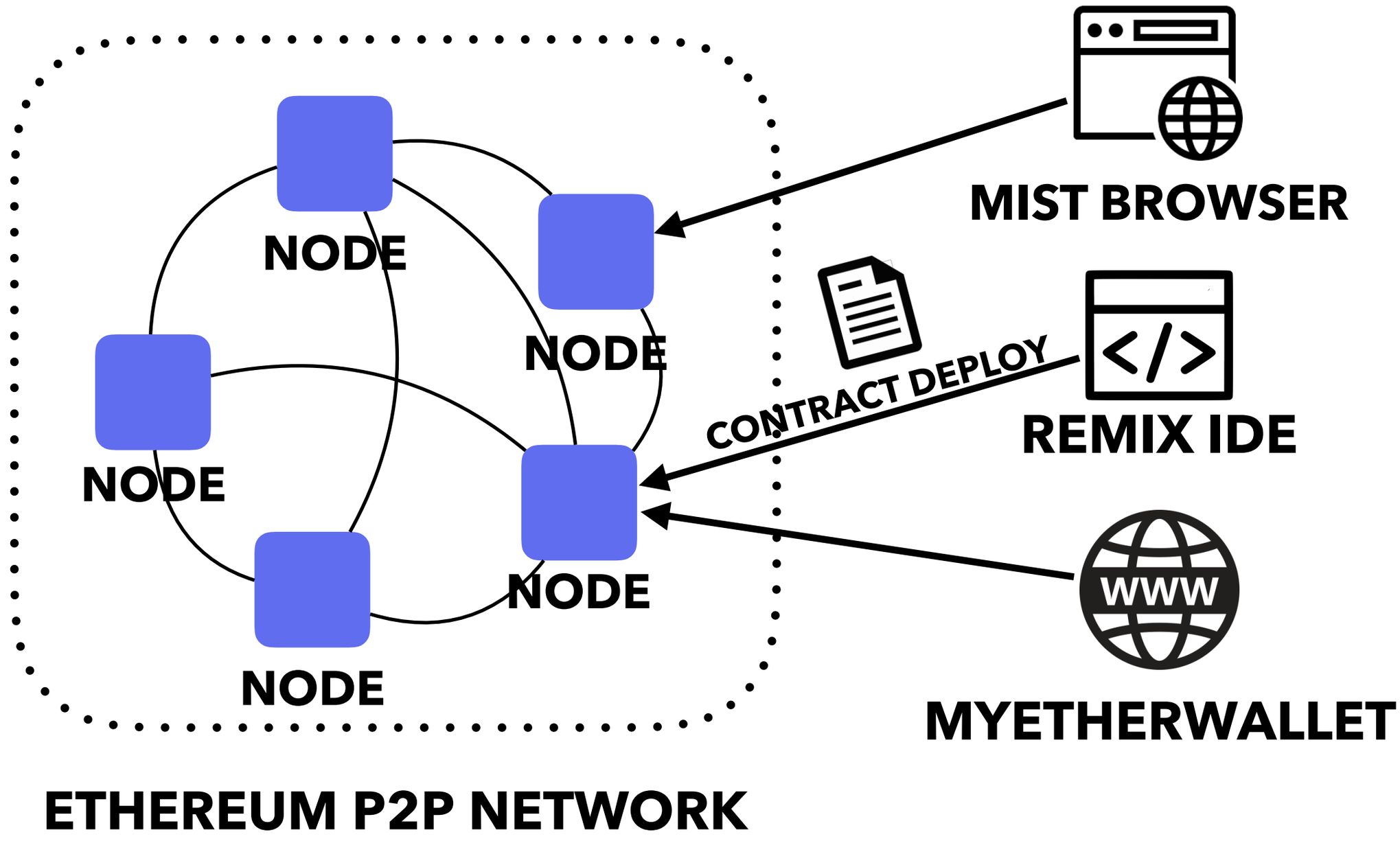
- 스마트 컨트랙이 소유
- 트랜잭션 혹은 함수에 의해 코드를 실행
- 주소와 이더, 코드를 가지고 있음

3 ETHEREUM SMART CONTRACt
이더리움 스마트 컨트랙트

이더리움에서의 스마트 컨트랙트는 블록체인 속 자동화된 봇

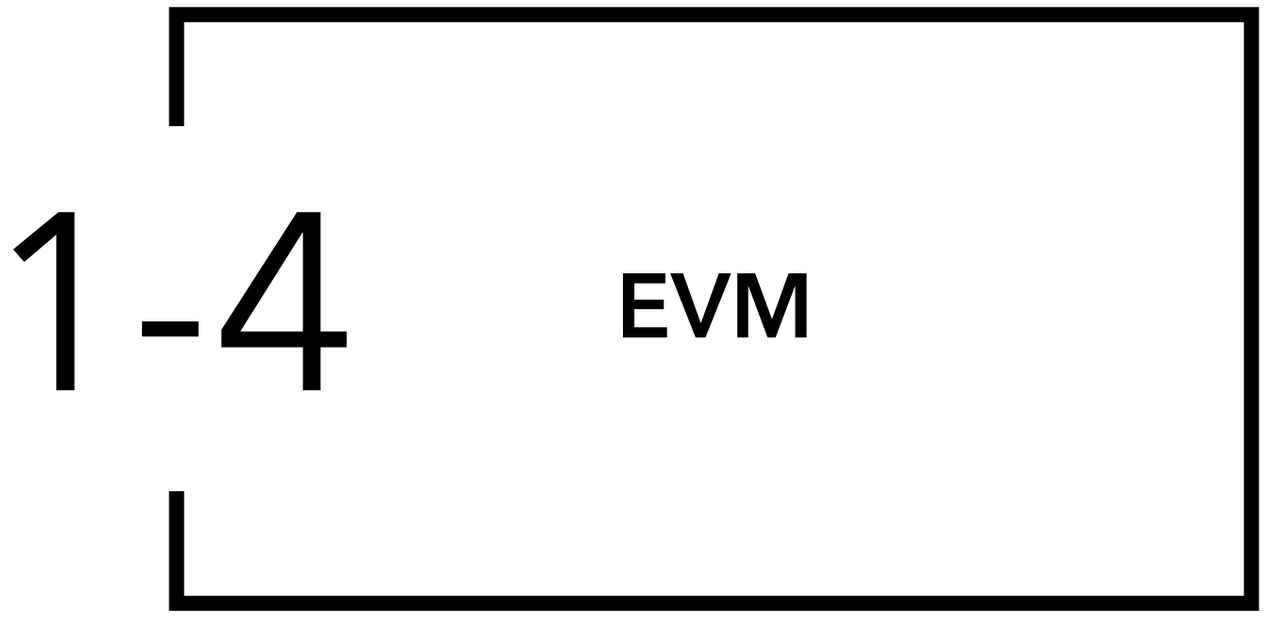
- 트랜잭션 발생(함수 실행)에 반응해 동작함
- 컨트랙을 통한 제어
  - 컨트랙 내부 이더
  - 컨트랙 내부 상태
  - 데이터의 영구적 저장

3 ETHEREUM
이더리움



3 ETHEREUM
이더리움

```
contract ERC20 is IERC20 {
  using SafeMath for uint256;

  mapping (address => uint256) private balances_;

  mapping (address => mapping (address => uint256)) private allowed_;

  uint256 private totalSupply_;

/**
* @dev Total number of tokens in existence
*/
function totalSupply() public view returns (uint256) {
  return totalSupply_;
}

/**
* @dev Gets the balance of the specified address.
* @param _owner The address to query the the balance of.
* @return An uint256 representing the amount owned by the passed address.
*/
function balanceOf(address _owner) public view returns (uint256) {
  return balances_[_owner];
}
```


1-4
EVM

4 ETHEREUM VIRTUAL MACHINE

EVM

SOLIDITY

```
contract ERC {
    /* ... */
}
```

Solidity compiler

VYPER

```
def register(key, value):
    /* ... */
```

Vyper compiler

EVM CODE

```
PUSH1 0x40
PUSH1 0x50
....
JUMP
```

4 ETHEREUM VIRTUAL MACHINE
EVM

- 모든 이더리움 노드는 블록 검증 과정에서 EVM을 실행시킨다
- 컨센서스 알고리즘은 3자 개입 없이 신뢰를 제공한다
    - 컨트랙의 실행 또한 컨센서스 알고리즘을 따라간다
- 컨트랙은 EVM 상에서 실행된다
- 실행되는 코드는 특정 언어가 아닌 EVM 바이트 코드

4 GAS
EVM의 가스

- 만약 코드가 무한 루프를 돈다면?

4 GAS
EVM의 가스

- 만약 코드가 무한 루프를 돈다면?
  - 모든 이더리움 노드가 먹통이 되는 문제가 발생한다.

4 GAS
EVM의 가스

- 해결책?
  - 컨트랙 실행에 일종의 수수료인 Gas를 **지불**
  - EVM의 op code 별로 실행에 필요한 가스가 존재
  - 트랜잭션 생성 시 명시한다
    - startgas: 사용 할 가스의 최대량
    - gasprice: 단위 가스당 지불할 이더양

Value	Mnemonic	Gas Used	Subset	Removed from stack	Added to stack	Notes
0x00	STOP	0	zero	0	0	Halts execution.
0x01	ADD	3	verylow	2	1	Addition operation
0x02	MUL	5	low	2	1	Multiplication oper
0x03	SUB	3	verylow	2	1	Subtraction operat
0x04	DIV	5	low	2	1	Integer division op
0x05	SDIV	5	low	2	1	Signed integer divi
0x06	MOD	5	low	2	1	Modulo remainder

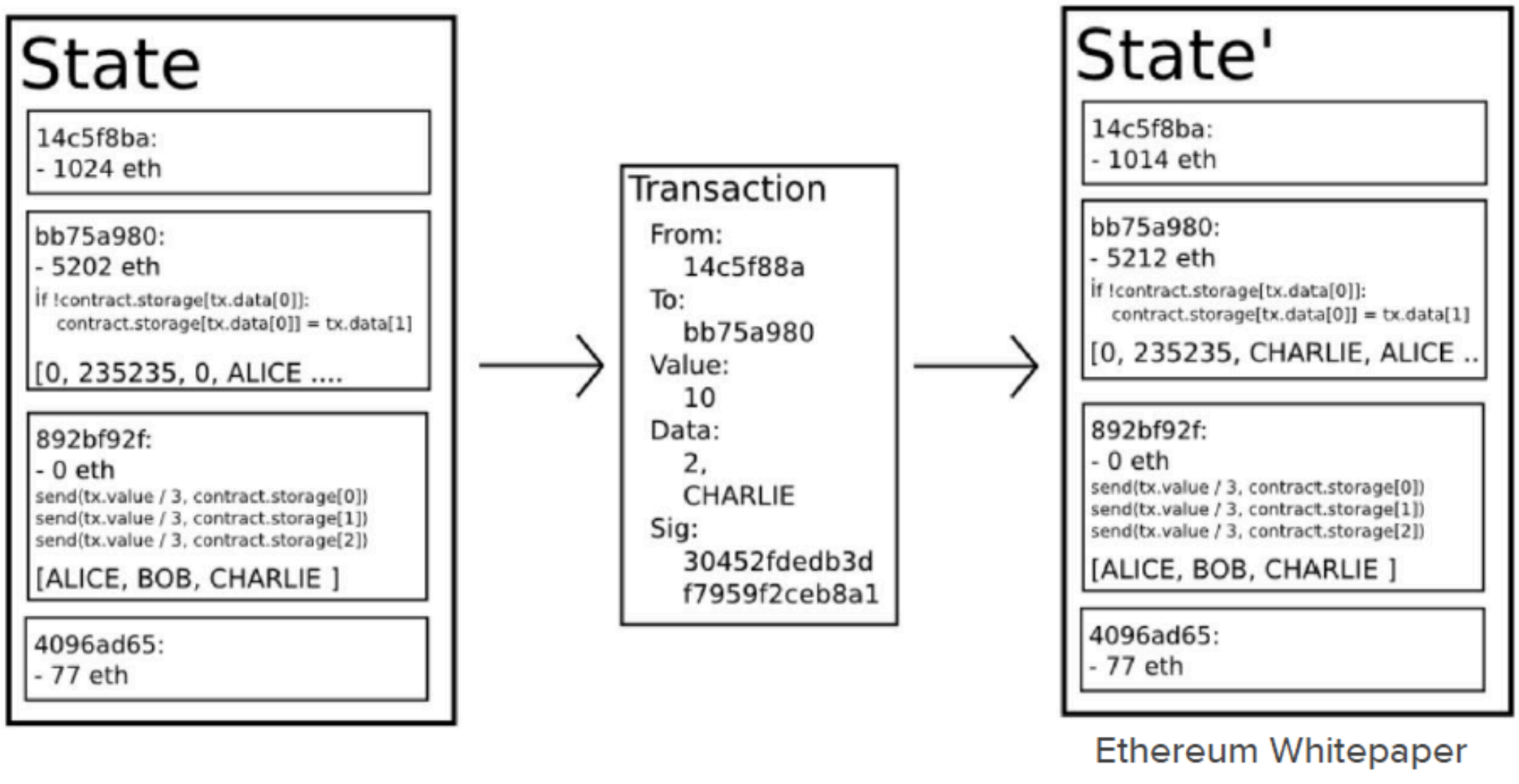
4 GAS
EVM의 가스

- 트랜잭션 생성 시
  - startgas * pricegas만큼의 이더가 트랜잭션 생성 유저의 계정에서 차감
- 트랜잭션 결과
  - 성공 시
    - 실행 후 남겨진 gas는 모두 트랜잭션 생성 유저에게 반환
  - 가스 부족으로 인한 실패
    - 실행 전 상태로 돌아감
    - 트랜잭션 생성에 소요된 startgas * pricegas는 반환되지 않음

=> 이더리움 네트워크의 컴퓨팅 파워를 사용하는 데 드는 수수료

4 ETHEREUM NETWORK
이더리움 네트워크

- 일종의 상태 머신



Ethereum Whitepaper

Q&A

2 SOLIDITY

1 LEARNING SOLIDITY

예제를 통해 배우는 솔리디티

- 은행 컨트랙트
  1. 입금
  2. 출금
  3. 예금 조회

```
pragma solidity ^0.4.19;

contract SimpleBank {
    mapping (address => uint) private balances;

    address public owner;
    event LogDepositMade(address accountAddress, uint amount);

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

    function deposit() public payable returns (uint) {
      require((balances[msg.sender] + msg.value) >= balances[msg.sender]);
      balances[msg.sender] += msg.value;
      emit LogDepositMade(msg.sender, msg.value);
      return balances[msg.sender];
    }

    function withdraw(uint withdrawAmount) public returns (uint remainingBal) {
      require(withdrawAmount <= balances[msg.sender]);
      balances[msg.sender] -= withdrawAmount;
      msg.sender.transfer(withdrawAmount);
      return balances[msg.sender];
    }

    function balance() constant public returns (uint) {
        return balances[msg.sender];
    }
}
```

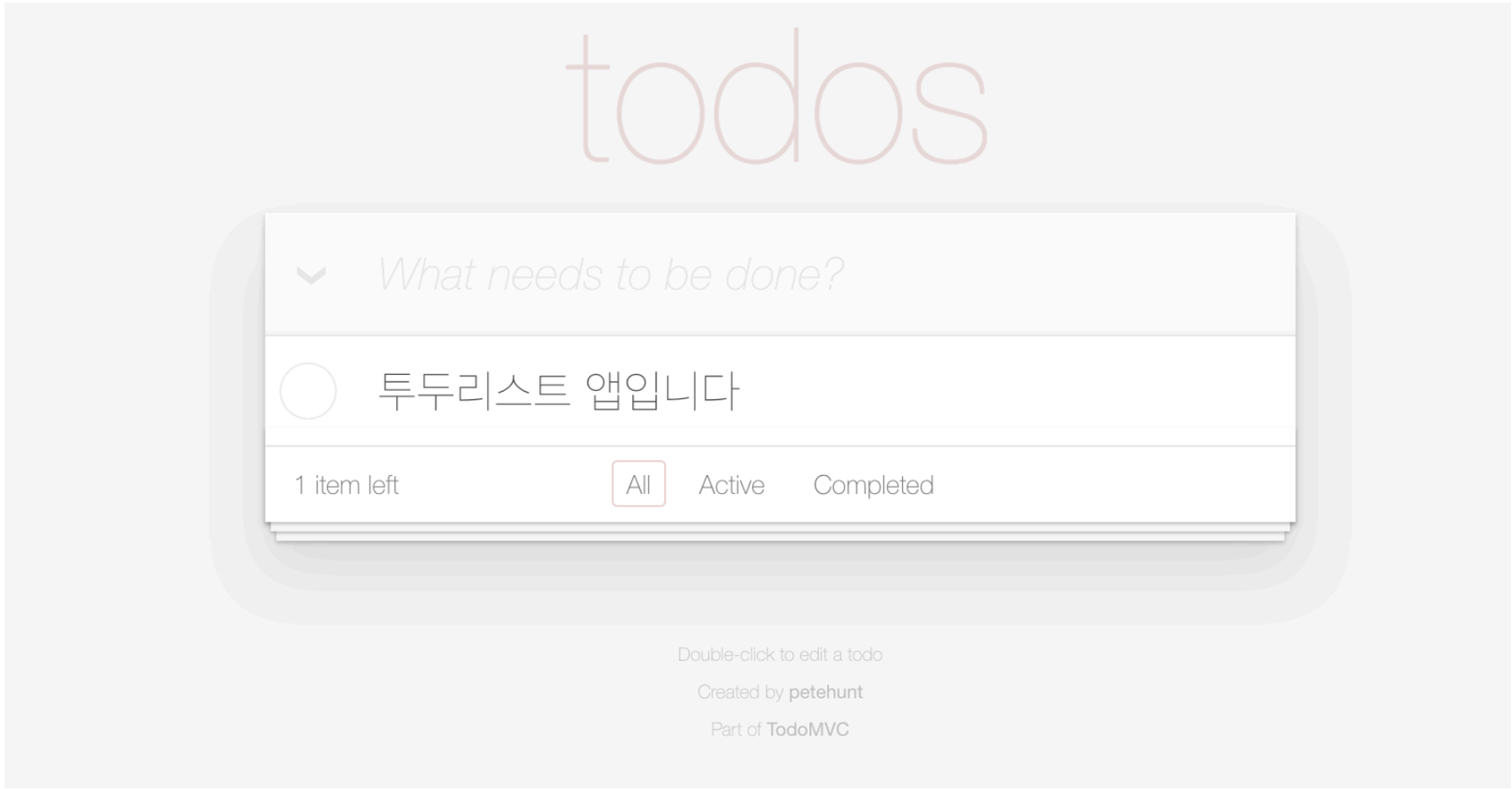
Q&A

3

DEVELOP

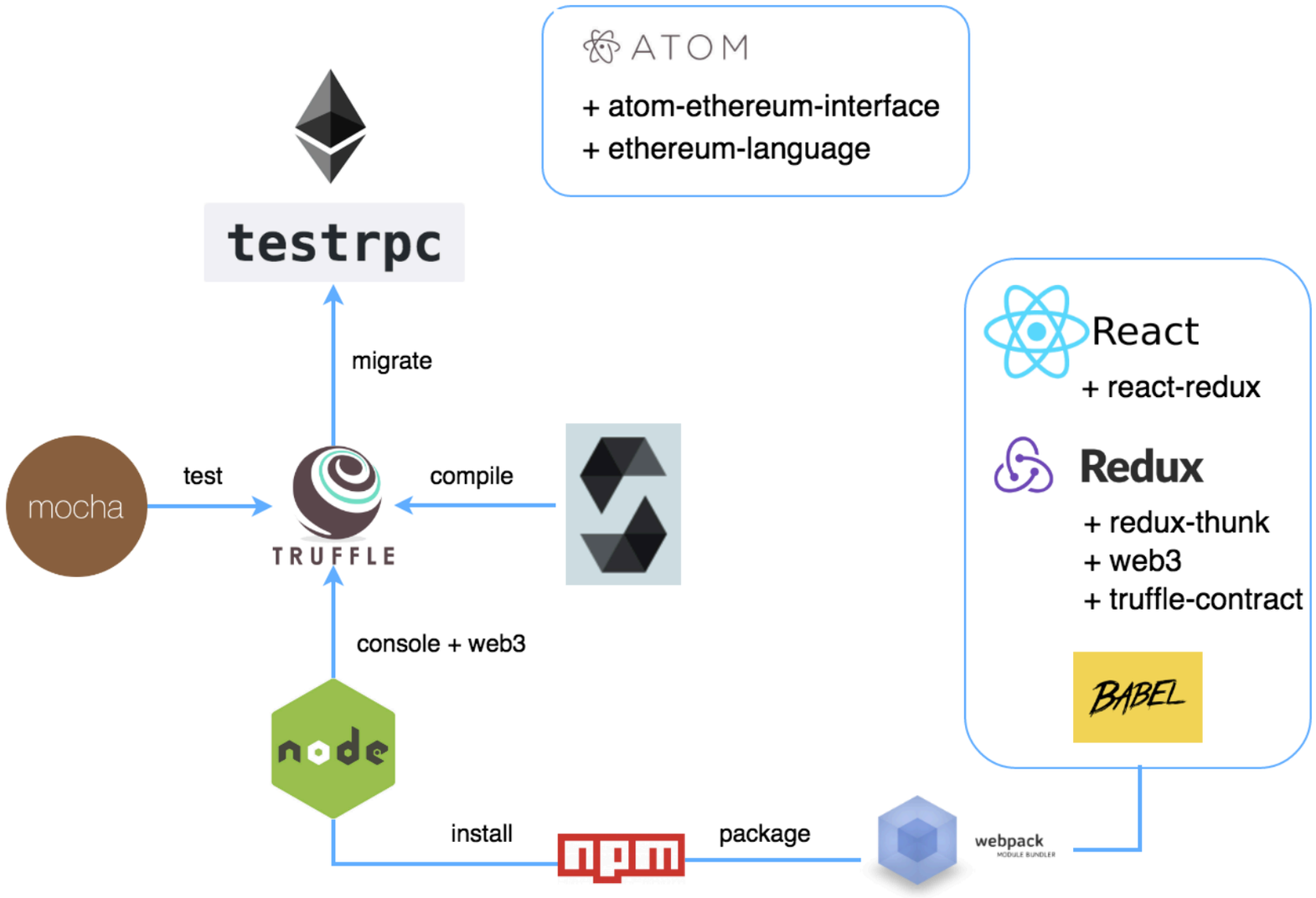
1 TODO MVC
투두리스트 웹앱

- 완성품



1 TODO MVC
투두리스트 웹앱

- 프로젝트 구조



Q&A