
10. Ethereum Token

이더리움 코어개발 2기

2019.03.14



- 일반적으로 사적으로 발행 된 전용 토큰, 세탁 토큰 및 아케이드 게임 토큰과 같이 중요하지 않은 본질적인 가치가있는 특수 용도의 동전과 같은 항목을 가리키는 데 사용
- 블록 체인에서 관리되는 "토큰"은 소유 할 수 있고 자산, 통화 또는 액세스 권한을 나타내는 블록 체인 기반 추상화를 의미하는 단어
- 이번에 다루게 될 내용 목록
 - 토큰의 다양한 용도
 - 토큰이 어떻게 생성되는지
 - 토큰의 속성

토큰 사용방법

- 일반적인 토큰은 디지털 개인 통화의 기능을 담당함.
- 이러한 토큰은 종종 서로 겹치는 다양한 기능을 제공하도록 프로그래밍이 가능하고 예를 들어, 토큰은 투표권, 액세스 권한 및 자원 소유권을 동시에 전달할 수 있습니다.
 - 통화 : 거래를 통해 결정되는 가치와 함께 통화로 사용 가능
 - 의지 : 경제 또는 자원 공유 환경에서 얻거나 생산 된 자원을 표현(cpu, disk...)
 - 유산 : 내재적 또는 외적, 유형 또는 무형 자산의 소유권을 나타냄(부동산,금...)
 - 액세스 : 액세스 권한을 표현할 수 있음(독점웹사이트, 포럼, 호텔, 렌터카...)
 - 지분 : 디지털 조직이나 법인의 주주 지분을 표현함
 - 투표 : 디지털 법률 시스템에서의 투표권을 나타냄
 - 소장품 : 디지털 수집품 혹은 물리적 수집품(그림)을 나타냄
 - 정체 : 디지털 신원(아바타)이나 법적 신원(국가id)을 나타냄
 - 증명 : 기관이나 분산 시스템에서의 사실 증명(결혼기록, 학위)등을 나타냄
 - 유용 : 서비스에 액세스하거나 지불하는데 사용이 가능함

토큰 및 대체 가능성

- “경제학에서 대체 성이란 개별 단위가 본질적으로 서로 교환 할 수 있는 상품 또는 상품의 재산입니다.” - Wikipedia
- 대체 가능한 토큰
 - 기본적으로 토큰의 단일 단위를 값이나 기능의 차이없이 다른 토큰으로 대체 할 수있는 경우 대체 할 수 있다.
- 대체 할 수 없는 토큰
 - 고유 한 유형 또는 무형의 항목을 나타내는 토큰은 상호 교환 할 수 없다.
 - 예를 들어, 특정 Van Gogh 그림의 소유권을 나타내는 토큰은 동일한 "예술 소유권 토큰"시스템의 일부분 일지라도 Picasso를 나타내는 다른 토큰과 동일하지 않음.

Utility or Equity

- 유틸리티 토큰
 - 토큰을 사용하여 서비스나 응용 프로그램 또는 자원에 액세스를 해야하는 곳에 사용이 됨
 - 가장 혁신적인 비즈니스 아이디어 중 일부는 실제로 암호화 영역에서 발생한다. 규제 기관이 법률을 채택하고 새로운 비즈니스 모델을 지원하기에 충분히 빠르지 않은 경우, 기업가 및 관련 인재는 보다 암호화에 민감한 다른 관할 구역에서 사업을 추구 할 수 있고 이것은 이미 일어나고 있다.
- 주식 토큰
 - 무언가의 통제 또는 소유권에 대한 지분을 나타내는 토큰이며
 - securities (equity) 토큰은 대부분의 지역에서 규제에 대상임.

Tokens on Ethereum



- 어떤면에서 첫 번째 블록 체인 통화 Bitcoin은 토큰 자체이다.
- Ethereum 전에 많은 토큰 플랫폼이 Bitcoin과 다른 암호화폐를 위해 개발됨. 그러나 Ethereum에 첫 번째 토큰 표준이 도입됨에 따라 토큰이 폭발적으로 증가함.
- Vitalik Buterin은 Ethereum과 같은 범용 프로그래머블 블록 체인에서 가장 명확하고 유용한 애플리케이션 중 하나로 토큰을 제안함.
 - 토큰은 이더리움과 다르다. 왜냐하면 이더리움 프로토콜은 토큰에 대해서 고려하지 않기 때문이다.
 - 이더를 전송하는 행위는 동작을 수행하지만 이는 토큰을 보내거나 소유하지 않는다
 - 따라서 이러한 이더 관련 수행은 프로토콜 단에서 수행이 되지만, 토큰은 스마트컨트랙트 단에서 수행이 된다.
 - 그러므로 Ethereum에서 새 토큰을 만들려면 새로운 스마트 계약을 만들어야 한다. 배포된 스마트 계약은 소유권, 이전 및 액세스 권한을 포함한 모든 것을 처리함.
 - 토큰 표준을 잘 따르자.

ERC20 토큰 표준



- 첫 번째 표준은 2015 년 11 월 Fabian Vogelsteller가 Ethereum Request for Comments (ERC)로 발표했습니다.
- GitHub 발행 번호 20이 자동으로 할당되어 "ERC20 토큰"이라는 이름이되었습니다.(Ethereum Improvement Proposal 20 (EIP-20))
- ERC20 표준 은 토큰을 구현하는 계약에 대한 공통 인터페이스를 정의하므로 모든 호환 가능한 토큰에 동일한 방식으로 액세스하고 사용할 수 있도록 함

ERC20 필수 기능 및 이벤트



- 필수기능
 - totalSupply : 토큰의 합계 단위를 리턴.
 - balanceOf : address에 대한 token balance를 리턴.
 - transfer : address and amount가 주어지면, 해당 토큰을 이전(전송) 처리함.
 - transferFrom : sender, recipient, and amount가 주어지면, 한 계정에서 다른 계정으로 토큰을 전송함. 승인과 함께 사용.
 - approve : recipient address and amount가 주어지면, 그 주소가 승인을 한 계정에서 최대 금액까지 여러번 송금하도록 승인함.
 - allowance : owner address and a spender address가 주어지면, 소비자가 소유자로부터 철회하도록 승인 된 남은 금액을 반환함.
- 이벤트
 - Transfer : 전송이 성공하면 해당 이벤트가 트리거됨.
 - Approval : 승인에 성공한 호출시 이벤트가 기록됨.
- ERC20 optional functions
 - name : 토큰의 이름 (e.g., "US Dollars")
 - symbol : 토큰의 심볼 (e.g., "USD")
 - decimals : 토큰 양을 나눌 수 있는 소수 자릿수를 반환합니다. 예를 들어, 소수가 2이면 토큰 양을 100으로 나눠 토큰양을 결정함.

The ERC20 interface defined in Solidity



```
contract ERC20 {  
    function totalSupply() constant returns (uint theTotalSupply);  
    function balanceOf(address _owner) constant returns (uint balance);  
    function transfer(address _to, uint _value) returns (bool success);  
    function transferFrom(address _from, address _to, uint _value) returns  
        (bool success);  
    function approve(address _spender, uint _value) returns (bool success);  
    function allowance(address _owner, address _spender) constant returns  
        (uint remaining);  
    event Transfer(address indexed _from, address indexed _to, uint _value);  
    event Approval(address indexed _owner, address indexed _spender, uint _value);  
}
```

ERC20 data structures



- internal table of token balances

```
mapping(address => uint256) balances;
```

- data mapping of allowances

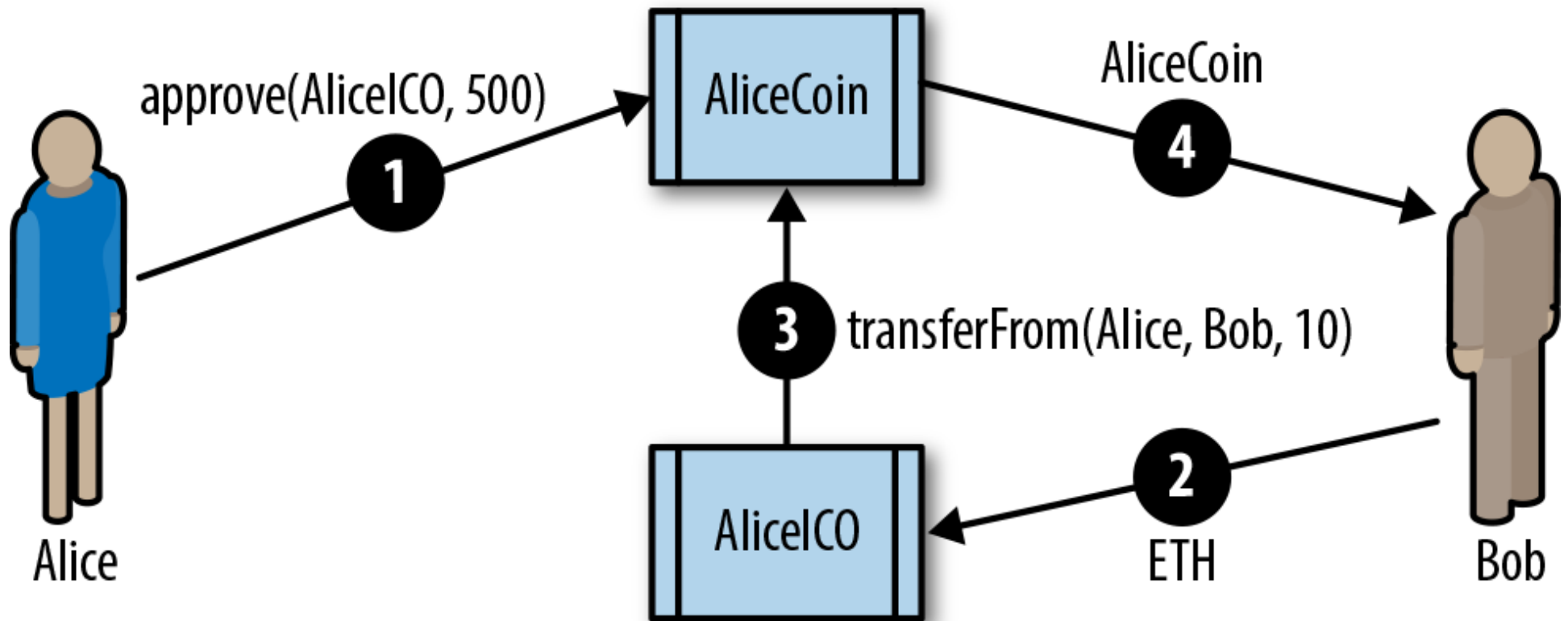
```
mapping (address => mapping (address => uint256)) public allowed;
```

ERC20 workflows: "transfer" and "approve & transferFrom"



- 두가지 표준 전송방식이 있다.
- single-transaction
 - If Alice wants to send 10 tokens to Bob
 - her wallet sends a transaction to the token contract's address
 - calling the transfer function with Bob's address and 10 as the arguments
 - The token contract adjusts Alice's balance (-10) and Bob's balance (+10) and issues a Transfer event
- two-transaction workflow
 - a token owner to delegate their control to another address
 - It is most often used to delegate control to a contract for distribution of tokens

ICO workflow



DEV Example



- <https://github.com/ethereumbook/ethereumbook/tree/develop/code/truffle/METoken>

ethereumbook / ethereumbook

Watch

435

★ Star

4,932

🍴 Fork

1,137

<> Code

🔔 Issues 4

🔗 Pull requests 6

📊 Insights

Branch: develop ▾

ethereumbook / code / truffle / METoken /

Create new file

Find file

History

 nadamsoreilly zeppelin to openzeppelin

Latest commit b8c3297 on 10 Oct

..

contracts	METoken.sol	4 months ago
migrations	Reorganized code/ directory	9 months ago
test	Unit tests for METoken	9 months ago
package-lock.json	zeppelin to openzeppelin	2 months ago
package.json	Added build directories and package.json for all code samples	4 months ago
truffle-config.js	Reorganized code/ directory	9 months ago

DEV Example



- Truffle, OpenZeppelin, ganache

- 소스 다운로드 및 실행

```
$ mkdir METoken
$ cd METoken
METoken $ truffle init
METoken $ npm init
```

- 라이브러리 설치

```
$ npm install openzeppelin-solidity@1.12.0
```

```
+ openzeppelin-solidity@1.12.0
added 1 package from 1 contributor and audited 2381 packages in 4.074s
```

- 컴파일

```
$ truffle compile
Compiling ./contracts/METoken.sol...
Compiling ./contracts/Migrations.sol...
```

DEV Example



- ganache console

ACCOUNTS

BLOCKS

TRANSACTIONS

LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK
4

GAS PRICE
100000000000

GAS LIMIT
6721975

NETWORK ID
5777

RPC SERVER
HTTP://127.0.0.1:7545

MINING STATUS
AUTOMINING

TX HASH

0xf36163615f41ef7ed8f4a8f192149a0bf633fe1a2398ce001bf44c43dc7bdda0

CONTRACT CALL

FROM ADDRESS

0x627306090abab3a6e1400e9345bc60c78a8bef57

TO CONTRACT ADDRESS

0x8cdf0cd259887258bc13a92c0a6da92698644c0

GAS USED

26981

VALUE

0

TX HASH

0xbe9290d59678b412e60ed6aefedb17364f4ad2977cfb2076b9b8ad415c5dc9f0

CONTRACT CREATION

FROM ADDRESS

0x627306090abab3a6e1400e9345bc60c78a8bef57

CREATED CONTRACT ADDRESS

0x345ca3e014aaf5dca488057592ee47305d9b3e10

GAS USED

1475948

VALUE

0

TX HASH

0xd7bc86d31bee32fa3988f1c1eabce403a1b5d570340a3a9cdba53a472ee8c956

CONTRACT CALL

FROM ADDRESS

0x627306090abab3a6e1400e9345bc60c78a8bef57

TO CONTRACT ADDRESS

0x8cdf0cd259887258bc13a92c0a6da92698644c0

GAS USED

41981

VALUE

0

TX HASH

0xb2e90a056dc6ad8e654683921fc613c796a03b89df6760ec1db1084ea4a084eb

CONTRACT CREATION

FROM ADDRESS

0x627306090abab3a6e1400e9345bc60c78a8bef57

CREATED CONTRACT ADDRESS

0x8cdf0cd259887258bc13a92c0a6da92698644c0

GAS USED

269607


VALUE


0


DEV Example





- Faucet example


 [ethereumbook](#) / [ethereumbook](#)


 Watch 435


 Star 4,932

 Fork 1,137

 Code


 Issues 4

 Pull requests 6









 Insights

Branch: **develop** ▾ [ethereumbook](#) / [code](#) / [truffle](#) /

Create new fileFind fileHistory

 **nadamsoreilly** zeppelin to openzeppelin Latest commit b8c3297 on 10 Oct

..

 CallExamples	CE edits Ch 7	2 months ago
 Faucet	Added build directories and package.json for all code samples	4 months ago
 FaucetEvents	Added build directories and package.json for all code samples	4 months ago
 FaucetReentryAttack	Added build directories and package.json for all code samples	4 months ago
 METoken	zeppelin to openzeppelin	2 months ago
 METoken_Faucet	zeppelin to openzeppelin	2 months ago
 METoken_METFaucet	zeppelin to openzeppelin	2 months ago
 console	Added build directories and package.json for all code samples	4 months ago

ERC20 토큰 문제



- ICO에서 자금을 모으기 위해 수천 개의 토큰이 출시되며 폭발적으로 성장함.
- 잠재적인 문제가 존재함
 - ether를 주소로 보내는 트랜잭션은 주소의 상태를 변경하지만, 토큰 전송에서 실제로 토큰의 수신자에게 전송된 트랜잭션은 없음. 토큰 계약 자체의 맵에서 관리됨.
 - ERC20 토큰이 많아지면서 사용 가능한 토큰보다 전자 메일 스팸과 유사하게 됨. 그들은 사용자를 유치하기 위해 이더리움 활동이있는 계정에 대한 잔액을 자동으로 생성하기 때문.
 - 이더리움을 보내거나 Ethereum 계약을 사용하려면 가스를 지불하기 위해 이더가 필요함. 토큰을 보내려면 역시 이더가 필요함. 사람들이 혼란스러워..
- 이를 해결하기 위한 다른 인터페이스가 발전함.

ERC223

- ERC223 제안서는 목적지 주소가 계약인지 아닌지 여부를 감지함으로써 실수로 토큰을 계약 (토큰을 지원할 수도 있고 지원하지 않을 수도 있음)으로 전송하는 문제를 해결하려고 시도합니다
- 토큰을 승인하도록 설계된 계약이 tokenFallback이라는 함수를 구현하도록 요구함.
- 전송 대상이 계약이고 계약에 토큰에 대한 지원이없는 경우 (즉, tokenFallback을 구현하지 않은 경우) 전송이 실패함.
- 많이 사용되지 않음.

```
function isContract(address _addr) private view returns (bool is_contract) {  
    uint length;  
    assembly {  
        // retrieve the size of the code on target address; this needs assembly  
        length := extcodesize(_addr)  
    }  
    return (length>0);  
}
```

ERC777

- ERC20 호환 인터페이스 제공
- send 함수를 사용하여 토큰을 전송하려면 ether 전송과 비슷합니다.
- 토큰 계약 등록을 위해 ERC820과 호환 가능
- tokensToSend function을 사용해서 토큰을 보내기전에 계약과 주소를 제어가능
- tokensReceived function을 사용해서 tokens' receipt 알림을 받을 수 있으며 토큰이 해당 계약에 lock될 가능성을 줄여주기 위해 사용됨.
- proxy contracts 을 사용할 수 있음
- contract or EOA로 전송하는 방식이 동일함.
- 토큰을 minting and burning 하는 특별한 이벤트를 제공함.
- userData and operatorData fields 같은 metadata를 제공함.

ERC777

```
interface ERC777Token {
    function name() public constant returns (string);
    function symbol() public constant returns (string);
    function totalSupply() public constant returns (uint256);
    function granularity() public constant returns (uint256);
    function balanceOf(address owner) public constant returns (uint256);

    function send(address to, uint256 amount, bytes userData) public;

    function authorizeOperator(address operator) public;
    function revokeOperator(address operator) public;
    function isOperatorFor(address operator, address tokenHolder)
        public constant returns (bool);
    function operatorSend(address from, address to, uint256 amount,
        bytes userData, bytes operatorData) public;

    event Sent(address indexed operator, address indexed from,
        address indexed to, uint256 amount, bytes userData,
        bytes operatorData);
    event Minted(address indexed operator, address indexed to,
        uint256 amount, bytes operatorData);
    event Burned(address indexed operator, address indexed from,
        uint256 amount, bytes userData, bytes operatorData);
    event AuthorizedOperator(address indexed operator,
        address indexed tokenHolder);
    event RevokedOperator(address indexed operator, address indexed tokenHolder);
}
```

ERC777



- ERC777 hooks

```
interface ERC777TokensSender {  
    function tokensToSend(address operator, address from, address to,  
        uint value, bytes userData, bytes operatorData) public;  
}
```

```
interface ERC777TokensRecipient {  
    function tokensReceived(  
        address operator, address from, address to,  
        uint amount, bytes userData, bytes operatorData  
    ) public;  
}
```

ERC721: Non-fungible Token (Deed) Standard



- 대체 할 수없는 토큰

(증서)*deed*: A legal document that is signed and delivered, especially one regarding the ownership of property or legal rights.

- 증서 ID를 기본키로 사용함.

```
// Mapping from deed ID to owner  
mapping (uint256 => address) private deedOwner;
```

ERC721: Non-fungible Token (Deed) Standard



```
interface ERC721 /* is ERC165 */ {
    event Transfer(address indexed _from, address indexed _to, uint256 _deedId);
    event Approval(address indexed _owner, address indexed _approved,
        uint256 _deedId);
    event ApprovalForAll(address indexed _owner, address indexed _operator,
        bool _approved);

    function balanceOf(address _owner) external view returns (uint256 _balance);
    function ownerOf(uint256 _deedId) external view returns (address _owner);
    function transfer(address _to, uint256 _deedId) external payable;
    function transferFrom(address _from, address _to, uint256 _deedId)
        external payable;
    function approve(address _approved, uint256 _deedId) external payable;
    function setApprovalForAll(address _operator, boolean _approved) payable;
    function supportsInterface(bytes4 interfaceID) external view returns (bool);
}
```

ERC721: Non-fungible Token (Deed) Standard

The ERC721 optional interface for metadata is:

```
interface ERC721Metadata /* is ERC721 */ {  
    function name() external pure returns (string _name);  
    function symbol() external pure returns (string _symbol);  
    function deedUri(uint256 _deedId) external view returns (string _deedUri);  
}
```

The ERC721 optional interface for enumeration is:

```
interface ERC721Enumerable /* is ERC721 */ {  
    function totalSupply() external view returns (uint256 _count);  
    function deedByIndex(uint256 _index) external view returns (uint256 _deedId);  
    function countOfOwners() external view returns (uint256 _count);  
    function ownerByIndex(uint256 _index) external view returns (address _owner);  
    function deedOfOwnerByIndex(address _owner, uint256 _index) external view  
        returns (uint256 _deedId);  
}
```


토큰 인터페이스 표준 확장



- 토큰 표준은 기능이 제한되어 있는 매우 최소한의 인터페이스를 제공함.
- Owner control
- Burning : 토큰 소각
- Minting : 토큰 추가발행
- Crowdfunding : 세일즈를 위한 토큰 제공, 경매
- Caps : 미리 정의 된 불변의 제한을 설정 (minting과 반대)
- Recovery backdoors : recover, reverse을 위한 기능
- Whitelisting : 토큰 전송과 같은 작업을 특정 주소로 제한
- Blacklisting : 특정 주소를 허용하지 않음으로써 토큰 전송을 제한
- 위와 관련된 것은 OpenZeppelin에 많은 참조자료가 있으나 표준은 없음.