


# CardMaster dApp

# 크립토키티 카드도 MetaMask를 통한 인증함

Confirm account details

Your wallet address is **Ox25777...**



**본인 Public Key 가 계정임**

Nickname

Other users will see your nickname instead of your wallet address throughout CryptoKitties.

Email options

We require your email to send you product and account-related updates.

User privacy & rights


We improved a few things, so please review your settings [Terms of Use](#) and [Privacy Policy](#).

☐ I agree to CryptoKitties' Terms of Use.

☐ I agree to CryptoKitties' Privacy Policy.

☐ I want to get marketing updates (optional)

Engage Test Net

 **Account 5**

Ox25777F...

3.979 ETH  
1803.16 USD


SENT

TOKENS

173

June 21 2018 23:17


Ox25777F...

 ETH

172

June 21 2018 23:16


Ox25777F...

 ETH

171

June 21 2018 23:15


Contract Deployment

 ETH

170

June 21 2018 23:13

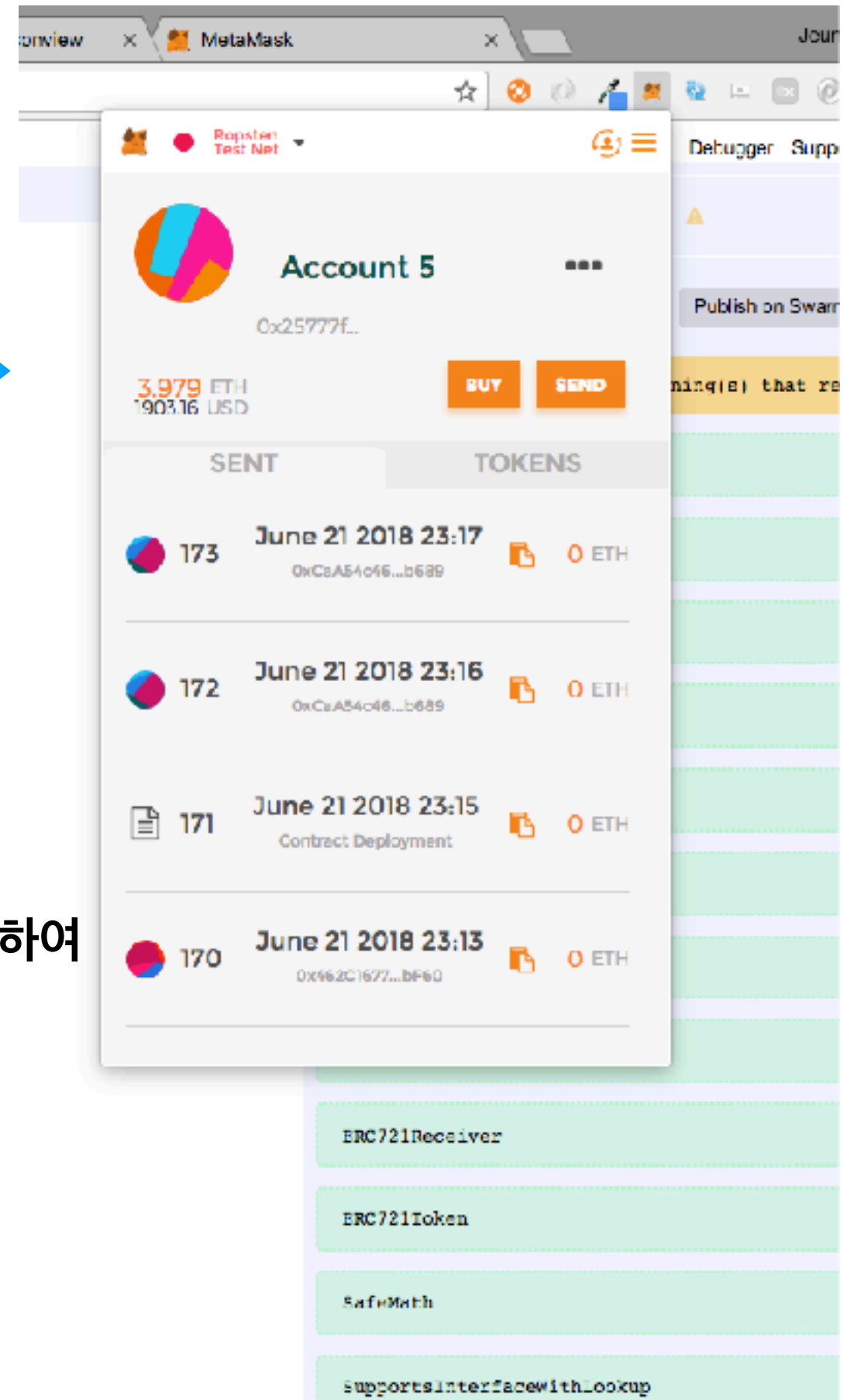
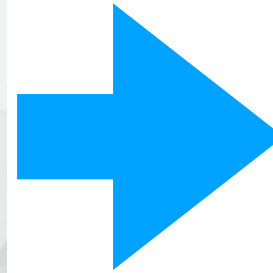
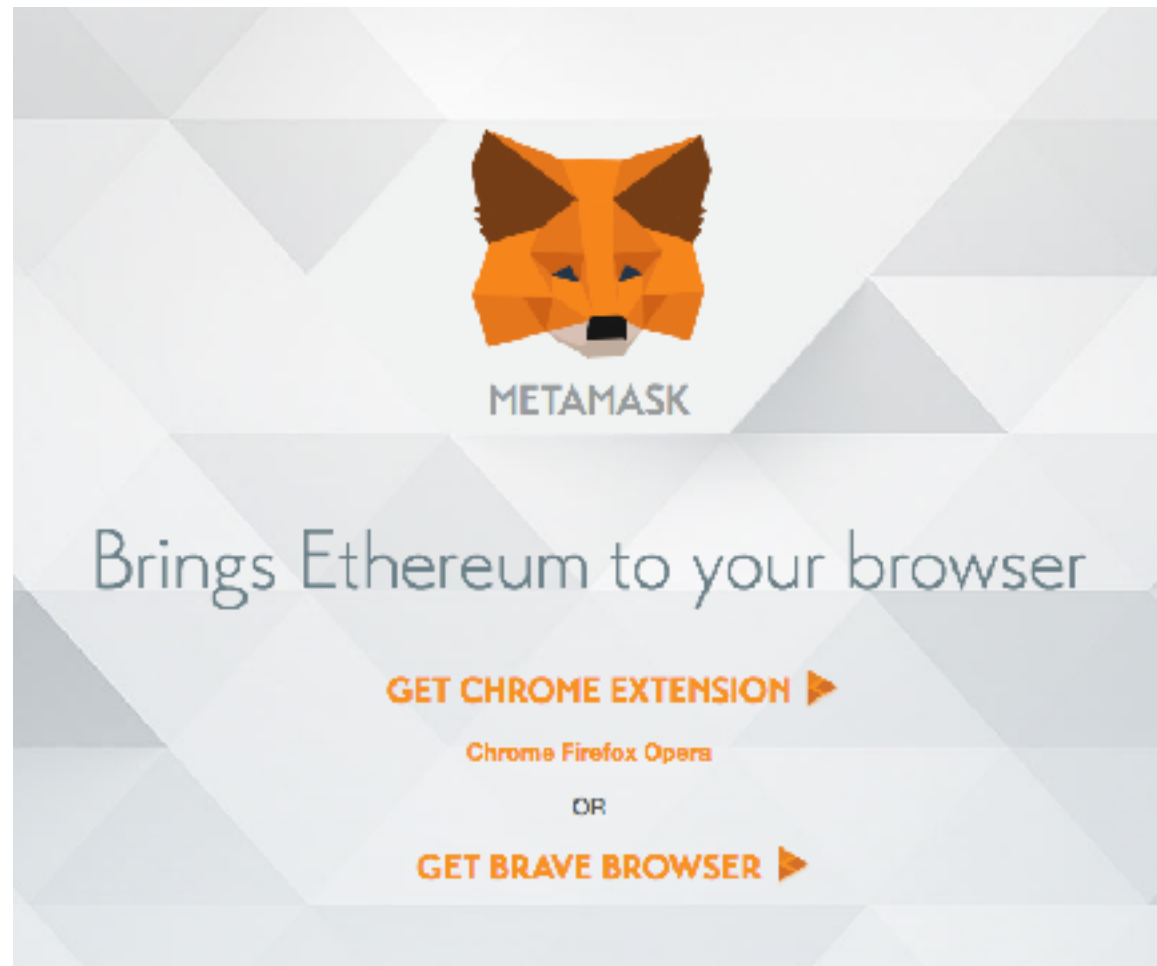
Ox25777F...

 ETH

회원가입시 MetaMask 본인 인증 > 네트워크 선택 > 서비스 이용

# MetaMask를 통한 로그인

<https://metamask.io/>

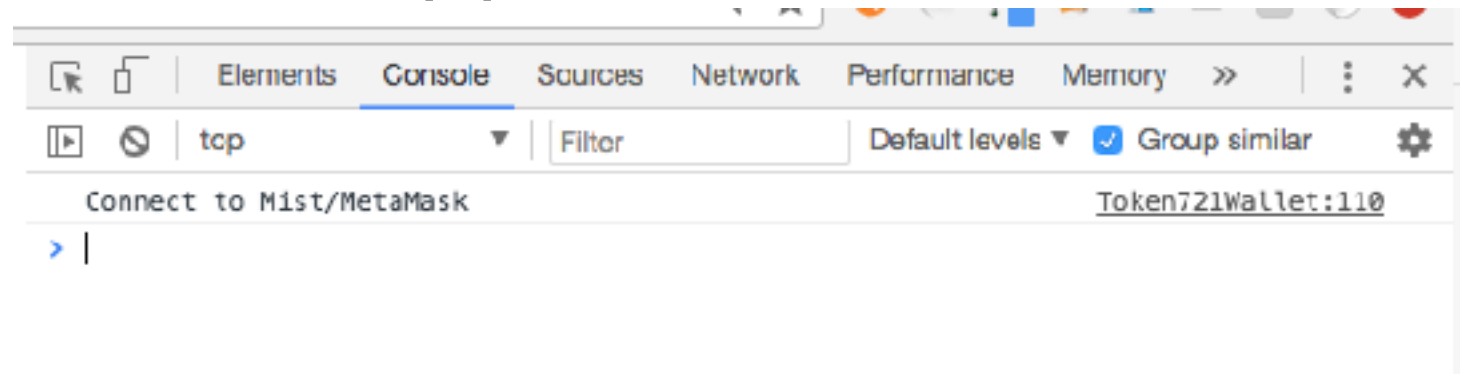


브라우저 Web3js 라이브러리를 통해 MetaMask 와 통신하여 사용자 인증을 할 수 있다.

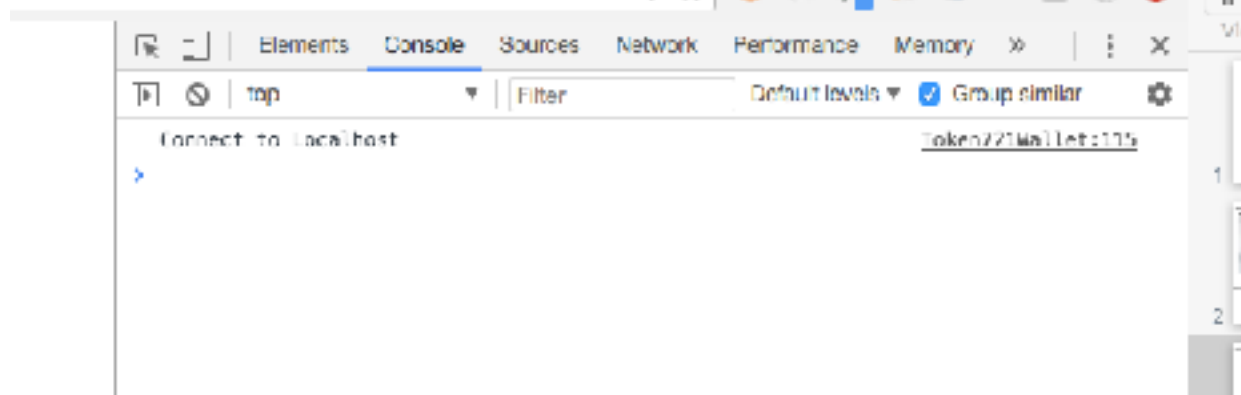
## web3.js 를 통한 접속 - 자바스크립트 코드

```
if (typeof window.web3 !== "undefined") {  
    //MetaMask 접속 시 window 객체 web3.currentProvider 멤버 변수, 함수가 생긴다.  
    web3js = new Web3(window.web3.currentProvider);  
    console.log("Connect to Mist/MetaMask");  
} else {  
    // 직접 JSON-RPC로 해당 Peer에 접속할 수 있는 방법이다.  
    // web3js = new Web3(new Web3.providers.HttpProvider("http://localhost:9545"));  
    // console.log("Connect to Localhost");  
}
```

## MetaMask 클라이언트 connection 된 모습



## localhost 노드에 JSON-RPC에 connection 된 모습



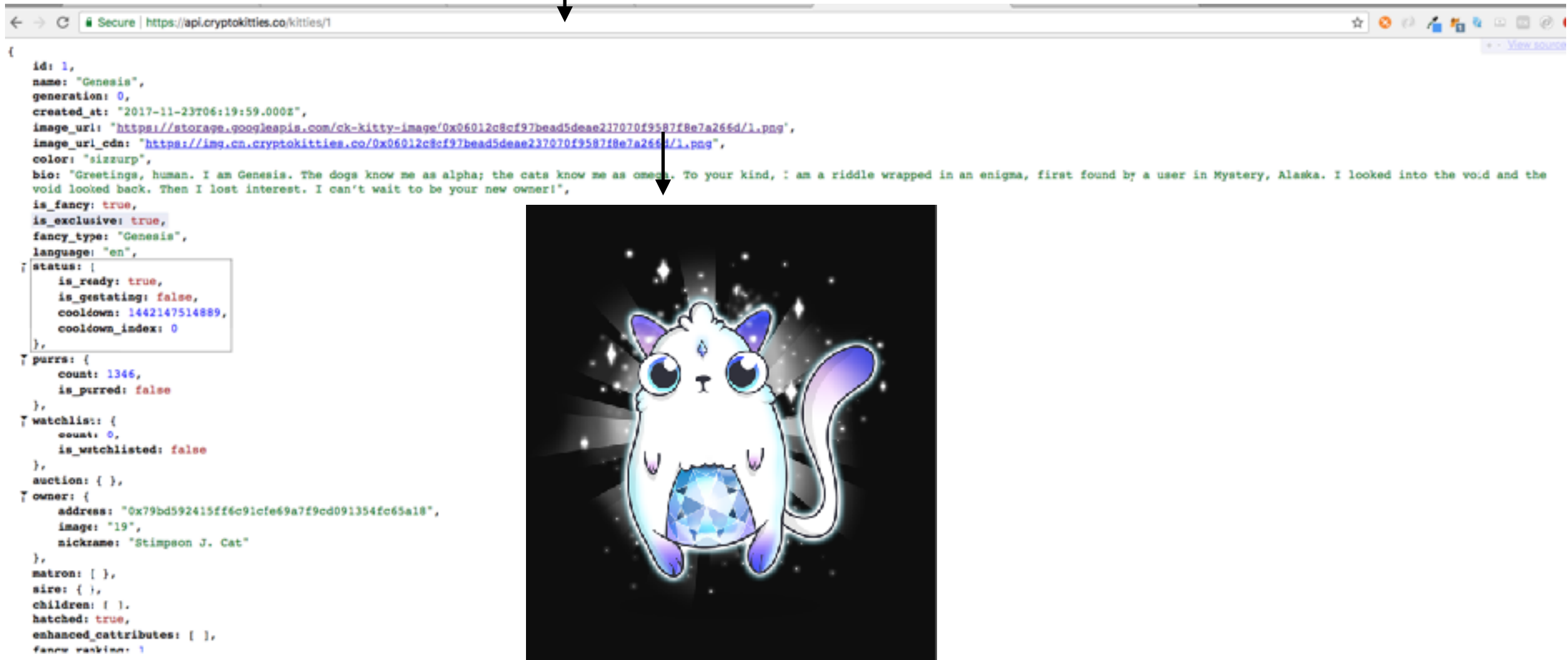
카드 생성 컨트랙트

# CardFactory.sol - 카드 생성 / 카드 메타 정보 셋팅 / 오너쉽 전환

- 새로운 카드를 생성함 ( 카드 ID, 카드 메타 정보 생성 포함 )

**function createCard(string uri) onlyOwner public;**

↓



```
{
  id: 1,
  name: "Genesis",
  generation: 0,
  created_at: "2017-11-23T06:19:59.000Z",
  image_url: "https://storage.googleapis.com/ck-kitty-image/0x06012c8cf97bead5deae237070f9587f8e7a266d/1.png",
  image_url_cdn: "https://img.cn.cryptokitties.co/0x06012c8cf97bead5deae237070f9587f8e7a266d/1.png",
  color: "sizzurp",
  bio: "Greetings, human. I am Genesis. The dogs know me as alpha; the cats know me as omega. To your kind, I am a riddle wrapped in an enigma, first found by a user in Mystery, Alaska. I looked into the void and the void looked back. Then I lost interest. I can't wait to be your new owner!",
  is_fancy: true,
  is_exclusive: true,
  fancy_type: "Genesis",
  language: "en",
  status: {
    is_ready: true,
    is_gestating: false,
    cooldown: 1442147514889,
    cooldown_index: 0
  },
  purrs: {
    count: 1346,
    is_purred: false
  },
  watchlist: {
    count: 0,
    is_watched: false
  },
  auction: {},
  owner: {
    address: "0x79bd592415ff6c91cfe69a7f9cd091354fc65a18",
    image: "19",
    nickname: "Stimpson J. Cat"
  },
  matron: [],
  sire: {},
  children: [],
  hatched: true,
  enhanced_attributes: [],
  fancy_ranking: 1
}
```

우선 CryptoKitty API를 활용하여 해당 ERC721 규약으로 발행된 토큰의 메타 정보를 제공하는 토큰 API URL을 저장함

## CardFactory.sol - 카드 생성 / 카드 메타 정보 셋팅 / 오너쉽 전환

- 카드 소멸 ( 메타정보도 함께 삭제 )

```
function burn(uint _tokenId) onlyOwner(_tokenId) public;
```

- 소유권 이전 위임과 소유권 이전

```
function approve(address _to, uint _tokenId) public;
```

```
function transferFrom(address _from, address _to, uint _tokenId) public;
```

- 본인 소유 카드 리스트 및 디테일 불러오기

```
function balanceOf(address _owner) public view returns (uint256);
```

↓ 갯수 ( 인덱스 )

```
function tokenByIndex(uint _index) public view returns (uint256);
```

for 문을 통해 리스트 출력

- 토큰의 메타정보 가져오기

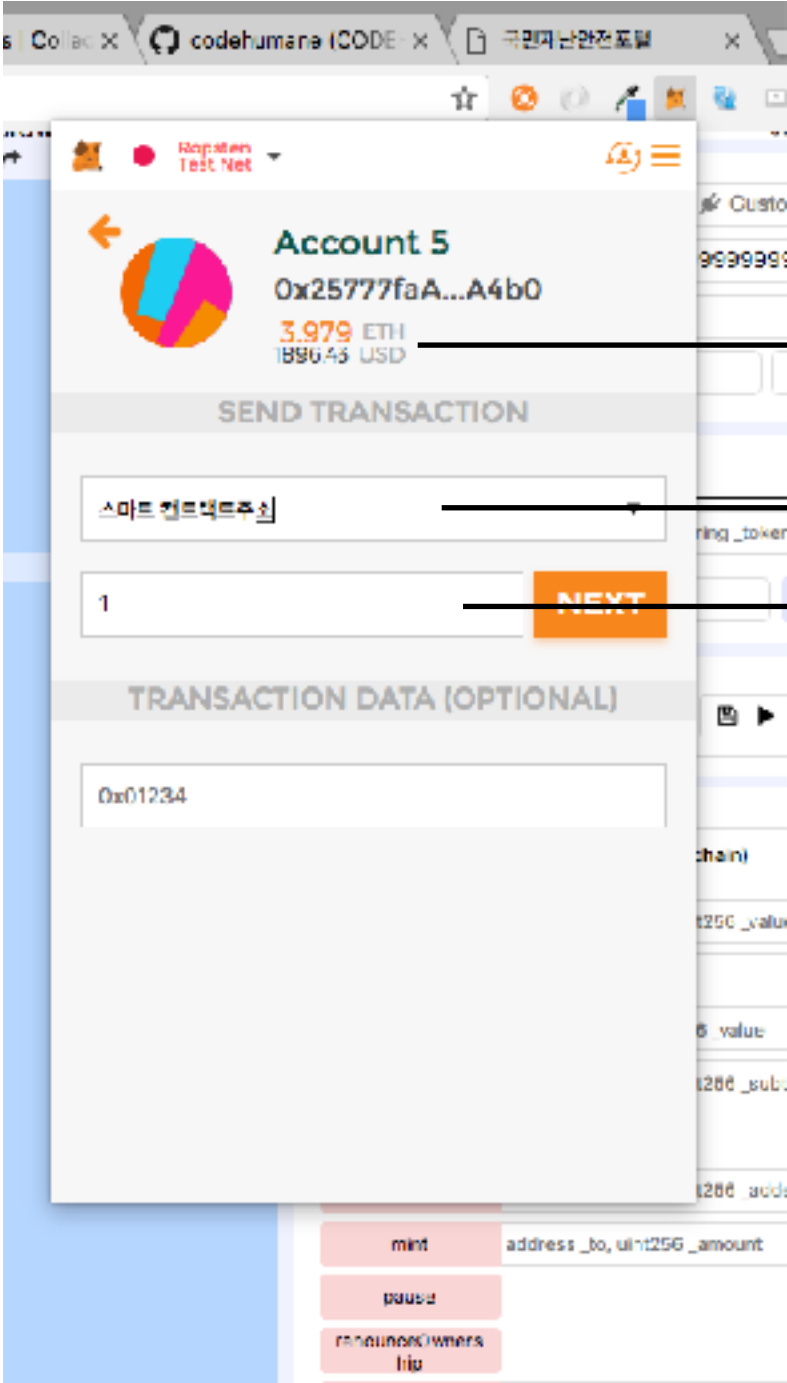
```
function tokenURI(uint _tokenId) public view returns (string);
```

토큰 구매



# 토큰 구매 TokenSale.sol

사용자는 카드마스터의 카드를 사기위해선 자체 발행한 토큰을 사야함. 토큰을 구매하기 위해선 이더를 거래소에서 Biat Money로 구입해야 함 (1:10000(토큰)의 비율로 판매)



Biat Money로 거래소에서 구매함

TokenSale 에 대한 SmartContract 주소

1Eth => 10000 Card Token 을 구매가능함

function buyTokens(address \_beneficiary) public payable ;

해당 계좌의 이더 잔고를 확인 후 보낸 이더의 비율 만  
큼 해당 계좌에 Card Token 을 전송함

카드 구매

# 카드 구매 CardTrading.sol

카드는 일정 시간마다 Provider에서 생성되고 해당 초기 구매 가격은 랜덤하게 설정함. 카드를 사기 위해선 보유 ERC20 토큰이 있어야 하며 해당 토큰은 이더리움으로 구매할 수 있다.



Price 10000 MyToken  
Owner : esak248



마켓 플레이스에 등록된 카드들

구매하기



```
function buyCard(address _purchaser, uint _tokenId, uint _price) public ;
```



성공 시

- 구매자의 토큰은 -10000, 판매자의 토큰은 +10000
- 카드의 소유주는 구매자로 바뀜

\* 해당 가격 체결에 대해선 Event로 남기며 각 카드의 Chart 를 기간별로 보여줄 수 있음

나의 지갑

# 나의 지갑 현황 Token.sol, CardFactory.sol

## 나의 보유 이더와 토큰

닉네임 : esak248

이메일 : [candoublej@gmail.com](mailto:candoublej@gmail.com)

지갑주소 : 0x627306090abab3a6e1400e9345bc60c78a8bef57

이더 보유현황 : 30.3 ether

카드토큰 보유현황 : 1,300,000,000 CardToken [\[입출력 현황 보러가기\]](#)

**function buyCard(address \_owner) public view returns (uint) ;**

## 나의 보유 카드 리스트



**제티**

구매가격 : 10,000 CT  
판매상태 : Not Sale



**케리**

구매가격 : 10,000 CT  
판매가격 : 15,000 CT  
판매상태 : Sale



**홍리**

구매가격 : 12,000 CT  
판매가격 : 300,000 CT  
판매상태 : Sale

# 카드 배틀

이후에 작성해야 함