

---

Hands-on Part

# *Metamask*のインストーラー

株式会社Neo Breakthrough  
山科 優希

---

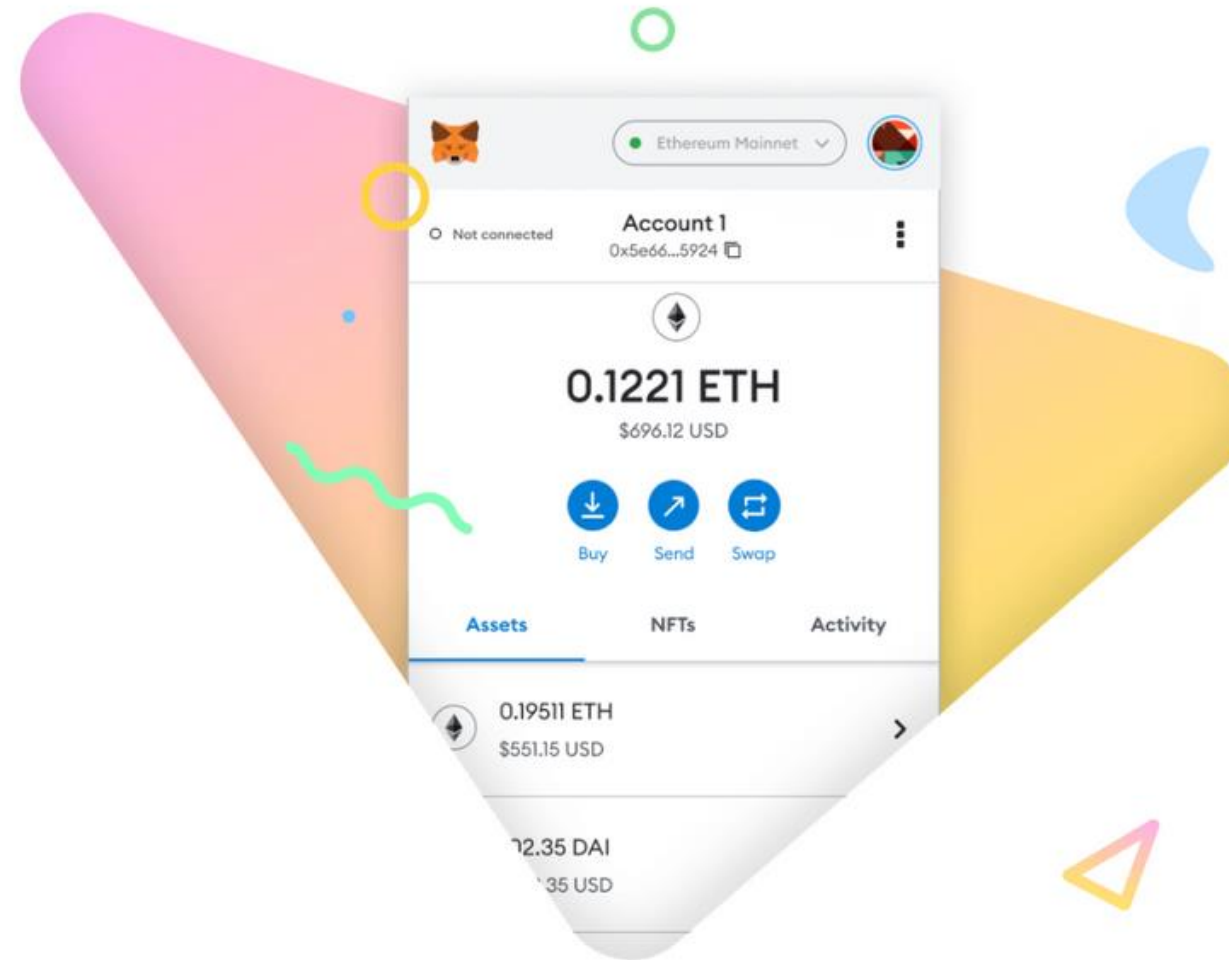
# *Metamask*とは

MetamaskはEthereum系ブロックチェーンのFT/NFTを一元管理できるソフトウェアで、米Consensysによって2016年に公開された。モバイルアプリとブラウザ拡張の両方がある。

# A crypto wallet & gateway to blockchain apps

Start exploring blockchain applications in seconds. Trusted by over 30 million users worldwide.

Download for 



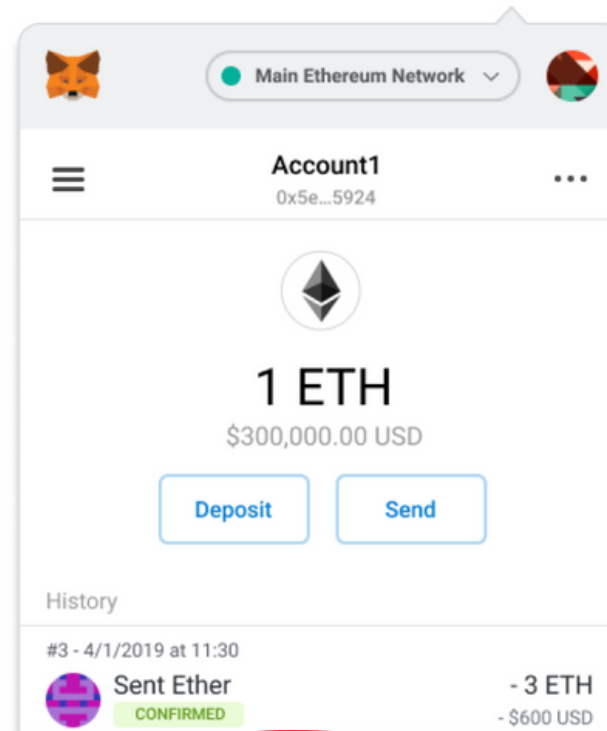


Chrome

iOS

Android

## Install MetaMask for your browser



Install MetaMask for Chrome



ホーム > 拡張機能 > MetaMask



# MetaMask

metamask.io

★★★★★ 2,889 ⓘ | 仕事効率化 | ユーザー数: 10,000,000+ 人

Chrome に追加

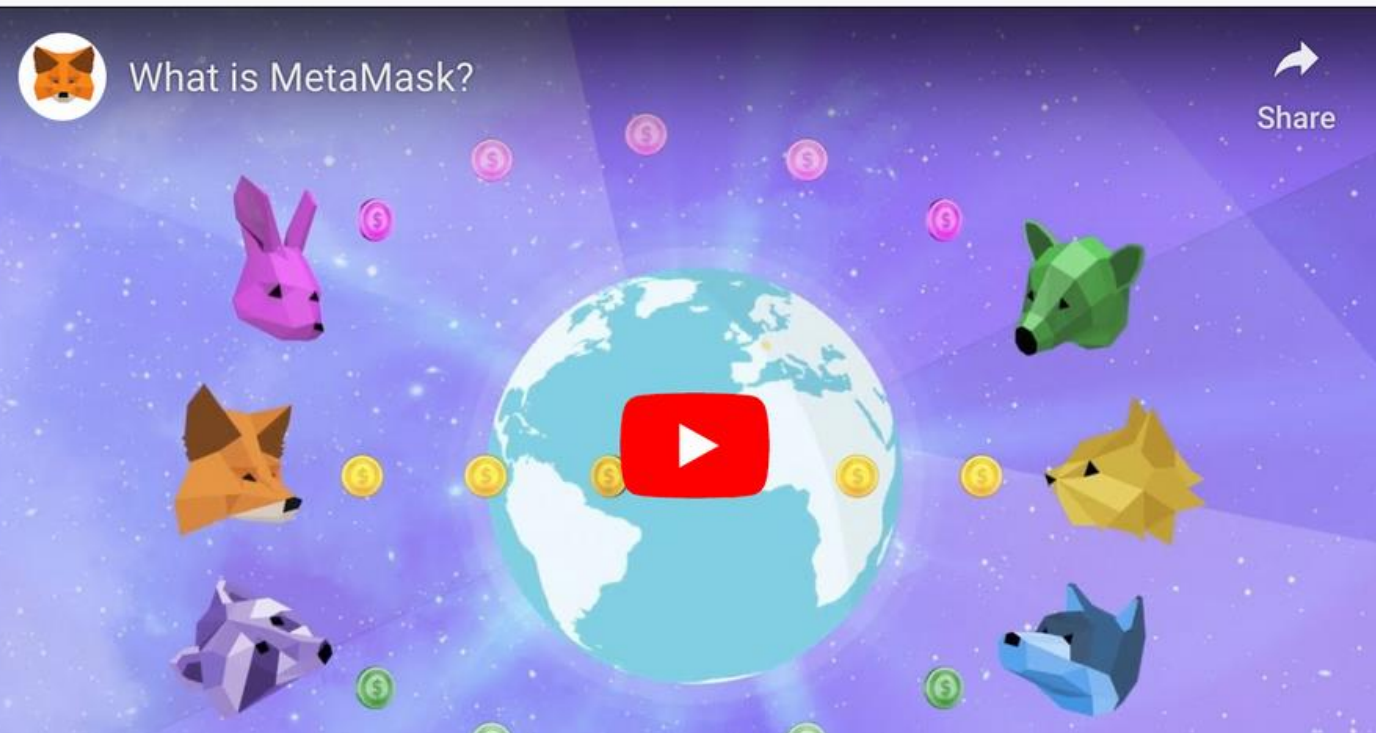
概要

プライバシーへの取り組み

レビュー

サポート

関連アイテム



## Let's get started

Trusted by millions, MetaMask is a secure wallet making the world of web3 accessible to all.



Create a new wallet

Import an existing wallet



## Help us improve MetaMask

MetaMask would like to gather usage data to better understand how our users interact with MetaMask. This data will be used to provide the service, which includes improving the service based on your use.

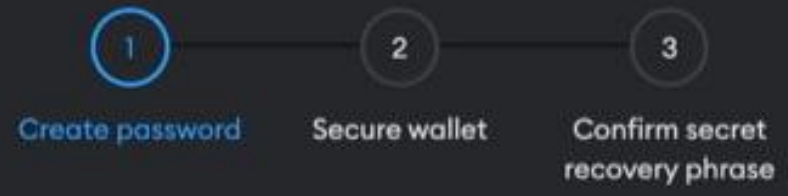
MetaMask will...

- ✓ Always allow you to opt-out via Settings
- ✓ Send anonymized click and pageview events
- ✗ **Never** collect information we don't need to provide the service (such as keys, addresses, transaction hashes, or balances)
- ✗ **Never** collect your full IP address\*
- ✗ **Never** sell data. Ever!

This data is aggregated and is therefore anonymous for the purposes of General Data Protection Regulation (EU) 2016/679.

\* When you use Infura as your default RPC provider in MetaMask, Infura will collect your IP address and your Ethereum wallet address when you send a transaction. We don't store this information in a way that allows our systems to associate those two pieces of data. For more information on how MetaMask and Infura interact from a data collection perspective, see our update [here](#). For more information on our privacy practices in general, see our [Privacy Policy here](#).

I agree



## Create password

This password will unlock your MetaMask wallet only on this device. MetaMask can not recover this password.

New password (8 characters min) [Show](#)

Confirm password

I understand that MetaMask can not recover this password for me. [Learn more](#)

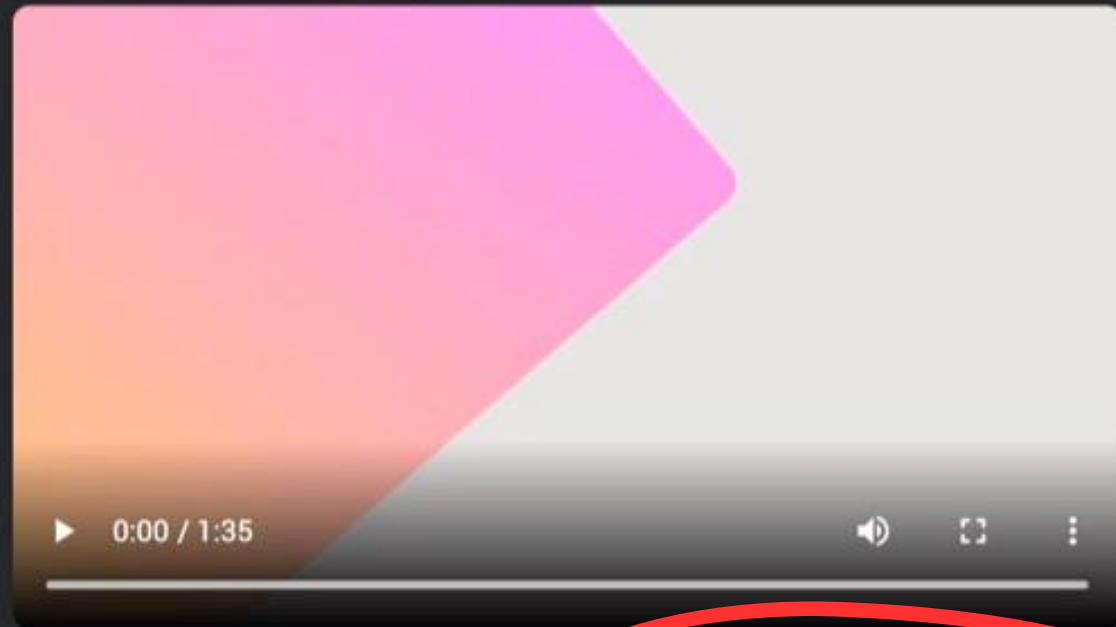
Create a new wallet





## Secure your wallet

Before getting started, watch this short video to learn about your Secret Recovery Phrase and how to keep your wallet safe.



Remind me later (not recommended)

Secure my wallet (recommended)

---

### **What is a Secret Recovery Phrase?**

Your Secret Recovery Phrase is a 12-word phrase that is the “master key” to your wallet and your funds

### **How do I save my Secret Recovery Phrase?**

- Save in a password manager
- Store in a safe deposit box
- Write down and store in multiple secret places

### **Should I share my Secret Recovery Phrase?**

Never, ever share your Secret Recovery Phrase, not even with MetaMask!

If someone asks for your recovery phrase they are likely trying to scam you and steal your wallet funds.

### **What is a Secret Recovery Phrase?**

Your Secret Recovery Phrase is a 12-word phrase that is the “master key” to your wallet and your funds

### **How do I save my Secret Recovery Phrase?**

- Save in a password manager
- Store in a safe deposit box
- Write down and store in multiple secret places

### **Should I share my Secret Recovery Phrase?**

Never, ever share your Secret Recovery Phrase, not even with MetaMask!

If someone asks for your recovery phrase they are likely trying to scam you and steal your wallet funds.

## Secret Recovery Phrase とは何か？

あなたのSRPは12単語で、これはウォレットと資産のMaster Keyです。

## どのようにSRPを保管するか？

- パスワードマネージャーで保管
- 安全な金庫に入れる
- 紙に書いて複数の秘密の場所に保管

## SRPを他の人と共有することはあるか？

Metamaskといえども、決してSRPを共有することはありません。

誰かがあなたのSRPを求めてきたら、彼らはあなたを騙して資産を盗もうとする可能性が高いです。

1

Create password

2

Secure wallet

3

Confirm secret  
recovery phrase

## Write down your Secret Recovery Phrase

Write down this 12-word Secret Recovery Phrase and save it in a place that you trust and only you can access.

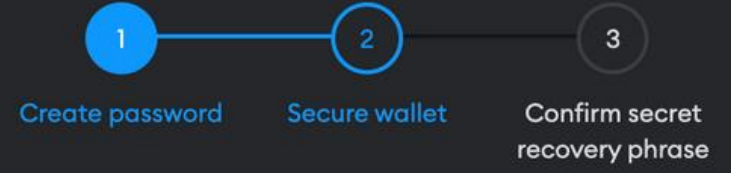
### Tips:

- Save in a password manager
- Store in a safe deposit box
- Write down and store in multiple secret places



Make sure no one is watching your screen

Reveal Secret Recovery Phrase



## Write down your Secret Recovery Phrase

Write down this 12-word Secret Recovery Phrase and save it in a place that you trust and only you can access.

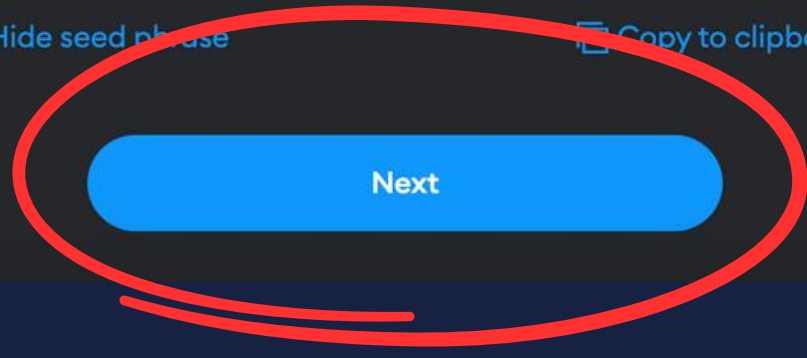
**Tips:**

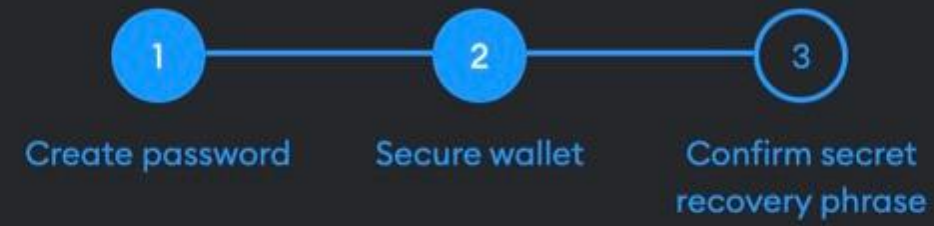
- Save in a password manager
- Store in a safe deposit box
- Write down and store in multiple secret places

1. <input type="text"/>	2. <input type="text"/>	3. <input type="text"/>
4. <input type="text"/>	5. <input type="text"/>	6. <input type="text"/>
7. <input type="text"/>	8. <input type="text"/>	9. <input type="text"/>
10. <input type="text"/>	11. <input type="text"/>	12. <input type="text"/>

Hide seed phrase

Copy to clipboard





# Confirm Secret Recovery Phrase

Confirm Secret Recovery Phrase

1. <input type="text"/>	2. <input type="text"/>	3. <input type="text"/>
4. <input type="text"/>	5. <input type="text"/>	6. <input type="text"/>
7. <input type="text"/>	8. <input type="text"/>	9. <input type="text"/>
10. <input type="text"/>	11. <input type="text"/>	12. <input type="text"/>

Confirm



## Wallet creation successful

You've successfully protected your wallet. Keep your Secret Recovery Phrase safe and secret -- it's your responsibility!

Remember:

- MetaMask can't recover your Secret Recovery Phrase.
- MetaMask will never ask you for your Secret Recovery Phrase.
- **Never share your Secret Recovery Phrase** with anyone or risk your funds being stolen
- [Learn more](#)

[Advanced configuration](#)

Got it!

---

Hands-on Part

# *ERC20/ERC721*のデータ構造について

株式会社Neo Breakthrough

山科 優希

---



- 
- *ERC20*や*ERC721*は*Token*に関する*ERC*
  - *Token*は『インターネット上で価値を持つもの』と捉えるとわかりやすい (個人的に)
  - ***ERC20***は***Fungible Token***と呼ばれ、代替可能性を持つトークン
  - ***Fungibility***とは、どの100円であっても同じ価値で、交換可能であるということ
  - ***ERC721***は***Non-Fungible Token***と呼ばれ、非代替可能性を持つトークン
  - ***Non-Fungibility***とは、一点ものの絵画が他の絵画と交換不可能であるということ
  - *ERC20*では各アドレスの保有量が記録され、*ERC721*では各トークン (*ID*) について保有者アドレスを記録する

\* *ERC20*と*ERC721*のコードレベルでの比較は*Hands-on*にて行う。

---

---

# *ERC20*のデータ構造について

---

## GLDToken.sol

```
// contracts/GLDToken.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract GLDToken is ERC20 {
    constructor(uint256 initialSupply) ERC20("Gold", "GLD") {
        _mint(msg.sender, initialSupply);
    }
}
```

## IERC20Metadata



```
import "@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol";
```

Interface for the optional metadata functions from the ERC20 standard.

*Available since v4.1.*

### FUNCTIONS

name()

symbol()

decimals()

---

totalSupply()

IERC20

balanceOf(account)

transfer(to, amount)

allowance(owner, spender)

approve(spender, amount)

transferFrom(from, to, amount)

# decimals (小数点)

## A Note on decimals

Often, you'll want to be able to divide your tokens into arbitrary amounts: say, if you own 5 GLD, you may want to send 1.5 GLD to a friend, and keep 3.5 GLD to yourself. Unfortunately, Solidity and the EVM do not support this behavior: only integer (whole) numbers can be used, which poses an issue. You may send 1 or 2 tokens, but not 1.5.

To work around this, [ERC20](#) provides a `decimals` field, which is used to specify how many decimal places a token has. To be able to transfer 1.5 GLD, `decimals` must be at least 1, since that number has a single decimal place.

How can this be achieved? It's actually very simple: a token contract can use larger integer values, so that a balance of 50 will represent 5 GLD, a transfer of 15 will correspond to 1.5 GLD being sent, and so on.

It is important to understand that `decimals` is *only used for display purposes*. All arithmetic inside the contract is still performed on integers, and it is the different user interfaces (wallets, exchanges, etc.) that must adjust the displayed values according to `decimals`. The total token supply and balance of each account are not specified in GLD: you need to divide by  $10^{**decimals}$  to get the actual GLD amount.

You'll probably want to use a `decimals` value of 18, just like Ether and most ERC20 token contracts in use, unless you have a very special reason not to. When minting tokens or transferring them around, you will be actually sending the number  $num\ GLD * (10^{**decimals})$ .

### NOTE

By default, [ERC20](#) uses a value of 18 for `decimals`. To use a different value, you will need to override the `decimals()` function in your contract.

”ERC20ではトークンの小数点以下の桁数を表すための `decimals` フィールドを提供している”

”デフォルトでは、ERC20は小数の値として18を使用します”

ERC20がFungible Tokenである所以はこの点でもありと  
考えられる。

後の説明にもあるがERC721の方には`decimals`がなく、  
代替不可能であると分かる。

---

# *ERC721*のデータ構造について

---

---

## GameItem.sol

```
// contracts/GameItem.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/utils/Counters.sol";

contract GameItem is ERC721URIStorage {
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;

    constructor() ERC721("GameItem", "ITM") {}

    function awardItem(address player, string memory tokenURI)
        public
        returns (uint256)
    {
        uint256 newItemId = _tokenIds.current();
        _mint(player, newItemId);
        _setTokenURI(newItemId, tokenURI);

        _tokenIds.increment();
        return newItemId;
    }
}
```

---

## GLDToken.sol

```
// contracts/GLDToken.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract GLDToken is ERC20 {
    constructor(uint256 initialSupply) ERC20("Gold", "GLD") {
        _mint(msg.sender, initialSupply);
    }
}
```

## GameItem.sol

```
// contracts/GameItem.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/utils/Counters.sol";

contract GameItem is ERC721URIStorage {
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;

    constructor() ERC721("GameItem", "ITM") {}

    function awardItem(address player, string memory tokenURI)
        public
        returns (uint256)
    {
        uint256 newItemId = _tokenIds.current();
        _mint(player, newItemId);
        _setTokenURI(newItemId, tokenURI);

        _tokenIds.increment();
        return newItemId;
    }
}
```



## IERC721Metadata



```
import "@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol";
```

See <https://eips.ethereum.org/EIPS/eip-721>

### FUNCTIONS

name()

symbol()

tokenURI(tokenId)

---

balanceOf(owner)

IERC721

ownerOf(tokenId)

safeTransferFrom(from, to, tokenId, data)

safeTransferFrom(from, to, tokenId)

transferFrom(from, to, tokenId)

approve(to, tokenId)

setApprovalForAll(operator, \_approved)

getApproved(tokenId)

isApprovedForAll(owner, operator)

---

supportsInterface(interfaceId)

IERC165

# ERC20 Metadata

# ERC721 Metadata

## IERC20Metadata #

```
import "@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol";
```

Interface for the optional metadata functions from the ERC20 standard.

*Available since v4.1.*

**FUNCTIONS**

- name()
- symbol()
- decimals()

---

- totalSupply() IERC20
- balanceOf(account)
- transfer(to, amount)
- allowance(owner, spender)
- approve(spender, amount)
- transferFrom(from, to, amount)

## IERC721Metadata #

```
import "@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol";
```

See <https://eips.ethereum.org/EIPS/eip-721>

**FUNCTIONS**

- name()
- symbol()
- tokenURI(tokenId)

---

- balanceOf(owner) IERC721
- ownerOf(tokenId)
- safeTransferFrom(from, to, tokenId, data)
- safeTransferFrom(from, to, tokenId)
- transferFrom(from, to, tokenId)
- approve(to, tokenId)
- setApprovalForAll(operator, \_approved)
- getApproved(tokenId)
- isApprovedForAll(owner, operator)

---

- supportsInterface(interfaceId) IERC165

*token*

*URI*

New items can be created:

```
> gameItem.awardItem(playerAddress, "https://game.example/item-id-8u5h2m.json")  
7
```

And the owner and metadata of each item queried:

```
> gameItem.ownerOf(7)  
playerAddress  
> gameItem.tokenURI(7)  
"https://game.example/item-id-8u5h2m.json"
```

This `tokenURI` should resolve to a JSON document that might look something like:

```
{  
  "name": "Thor's hammer",  
  "description": "Mjölfnir, the legendary hammer of the Norse god of thunder.",  
  "image": "https://game.example/item-id-8u5h2m.png",  
  "strength": 20  
}
```

新しいアイテムの作成

所有者とアイテムのメタデータのクエリ

*tokenURI*のJSONファイル例

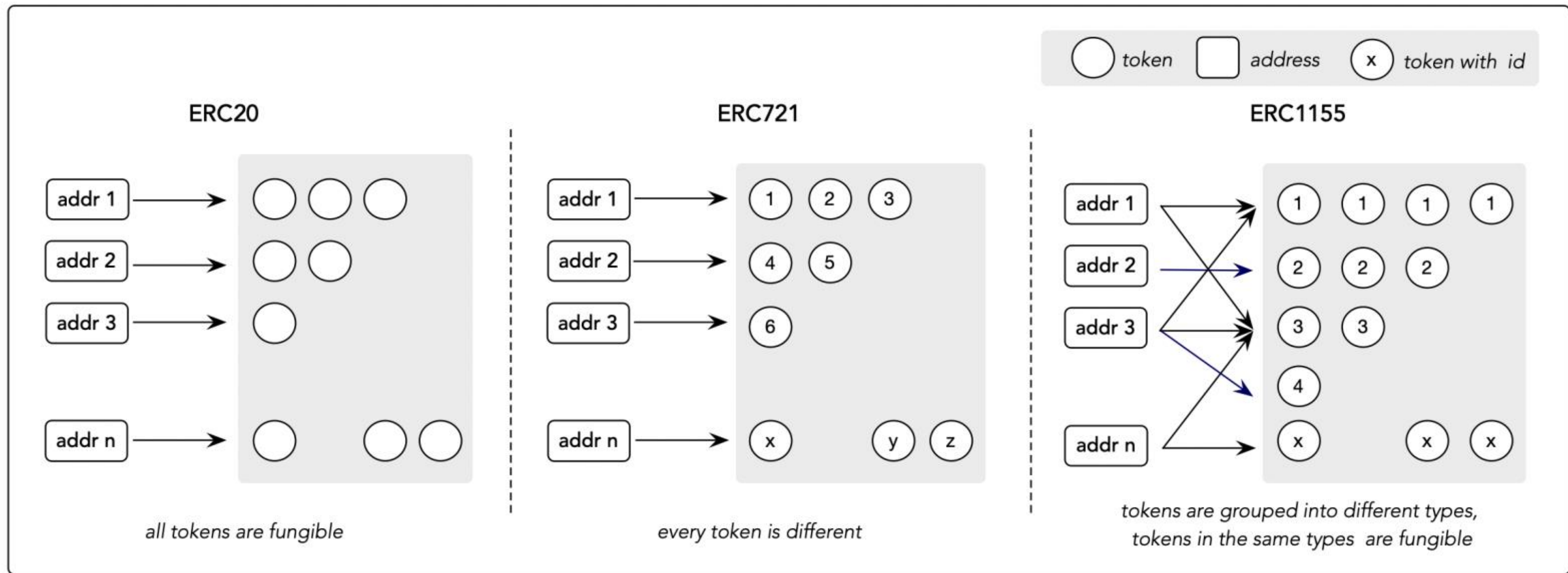


Fig. 2: NFT-related Token Standards

---

*ERC721*の*token URI*を見てもみよう

---

# メタデータ(メタデータ自身もホスト先がHTTPかIPFS等かに分かれる)

```
{  
  "title": "Asset Metadata",  
  "type": "object",  
  "properties": {  
    "name": {  
      "type": "string",  
      "description": "Identifies the asset to which this NFT represents"  
    },  
    "description": {  
      "type": "string",  
      "description": "Describes the asset to which this NFT represents"  
    },  
    "image": {  
      "type": "string",  
      "description": "A URI pointing to a resource with mime type image/* representing the asset to which this NFT  
    }  
  }  
}
```

画像データ参照先

https://~

ipfs://~

TokenID\*を指定して  
メタデータを参照

ブロックチェーン



\*TokenID: それぞれのトークンを指定する数字。

---

# HTTP

ロケーション指向型

『<https://aaa.com/b1>』

この表記の場合、[aaa.com](https://aaa.com)というサーバーの  
**b1**という場所にアクセスしている。

画像データがここで管理されており、  
当該サーバーが運用停止した場合

NFTの参照先画像が紛失されるため永続性がない。

また、**b1**にある画像を差し替えることで

NFTで表現している画像も（当然）差し代わる。

---

---

# IPFS

コンテンツ指向型

『ipfs://aQ1bsi...』

この表記の場合、『aQ1bsi...』はハッシュ値であり、当該ハッシュ値は画像データをハッシュ関数に投じて得られたもの。

動画や音楽もハッシュ化することでIPFSによるホストが可能。

但しIPFSも永続性が”完璧”とはいえないため、その点に留意する必要がある。

---



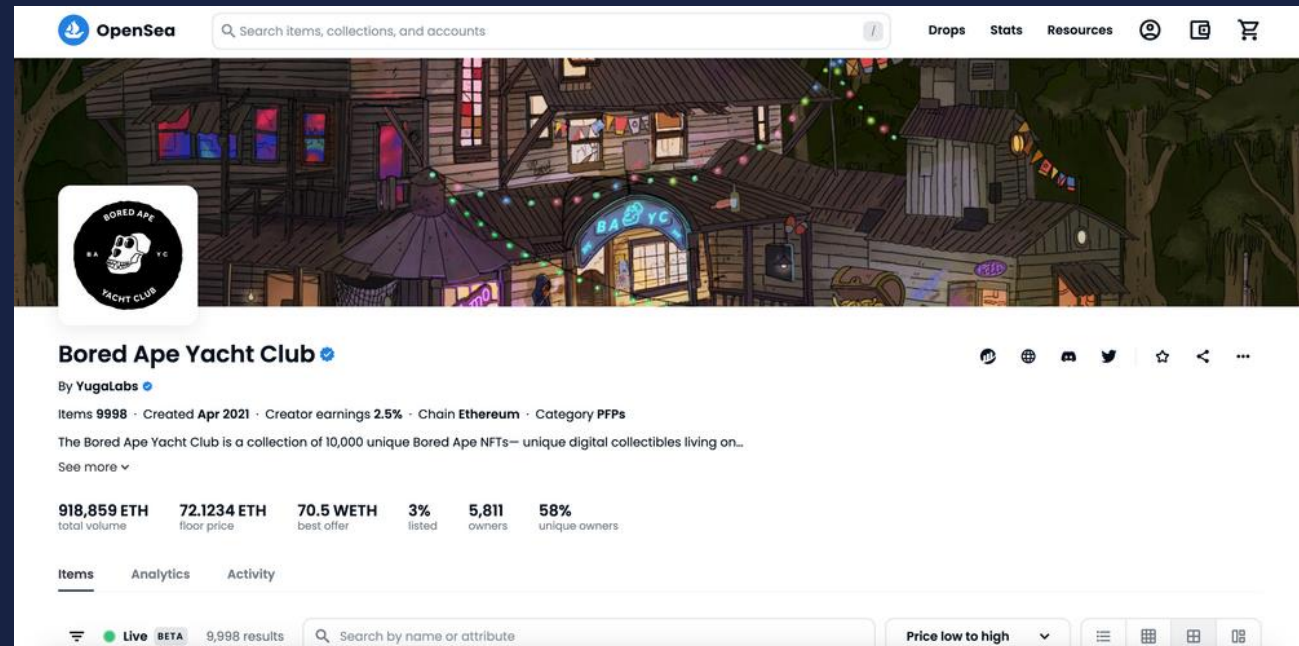
---

手を動かして確認してみよう  
(IPFS/中央サーバー)

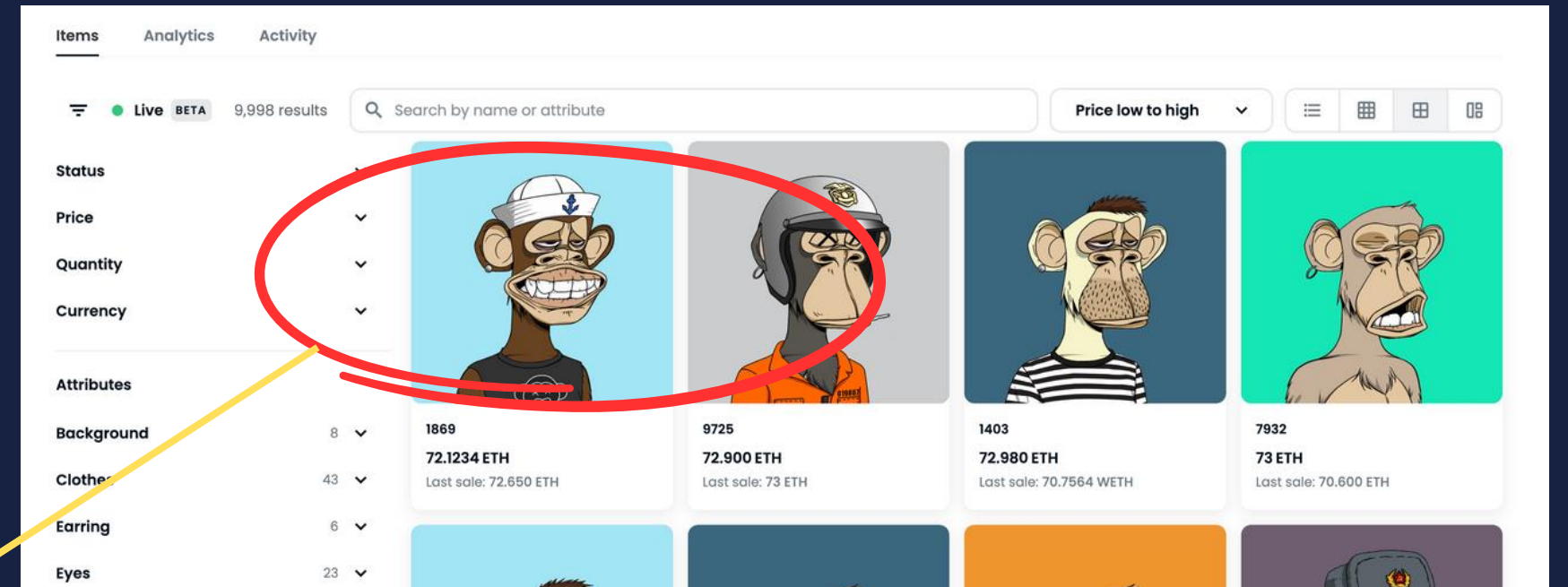
---

# IPFS型NFTの場合

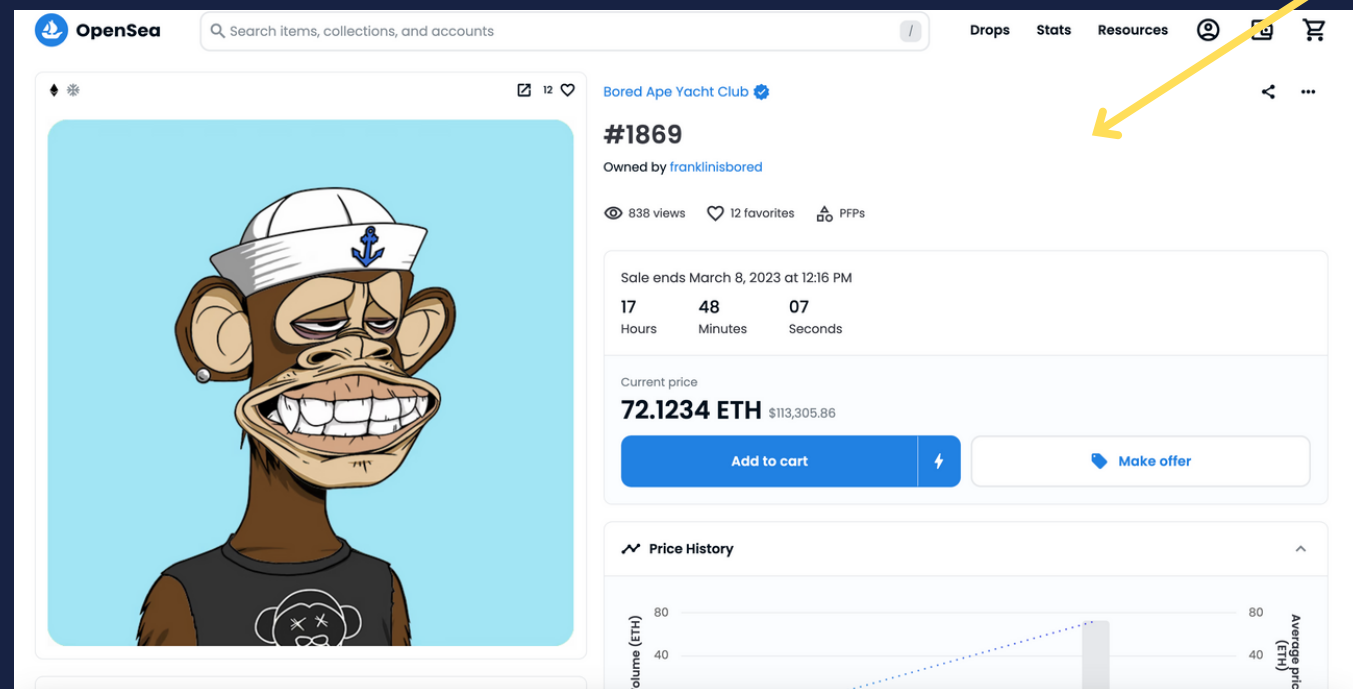
## 1: OpenSea\*



## 2: 一つのNFTを選択



## 3: コンテンツページ→スクロール



\*OpenSea  
NFTを販売しているECサイト(的な)  
今回はBAYCのページを利用。  
→<https://opensea.io/collection/boredapeyachtclub>

#### 4: Detailsを参照→0xbc4c...f13dをクリック(Token IDもメモ)

The screenshot shows the OpenSea interface for an item from the 'About Bored Ape Yacht Club' collection. The 'Details' section is expanded, showing the following information:

- Contract Address: [0xbc4c...f13d](#)
- Token ID: 1869
- Token Standard: ERC-721
- Chain: Ethereum
- Metadata: Frozen
- Creator Earnings: 2.5%

At the top right of the item card, the following details are visible:

- Price: 68.5 WETH
- USD Price: \$107,613.50
- Discount: 5% below
- Availability: in 4 days
- Item ID: FOF454

The 'Item Activity' section is also visible, showing a table of recent transactions:

Event	Price	From	To	Date
Transfer		<a href="#">chamathp</a>	<a href="#">franklinisbored</a>	20h ago
Sale	72.650 ETH	<a href="#">chamathp</a>	<a href="#">franklinisbored</a>	20h ago

## 5: Etherscan\*に遷移

ETH Price: \$1,567.07 (+0.09%) Gas: 22 Gwei

Search by Address / Txn Hash / Block / Token / Domain Name

Etherscan

Home Blockchain Tokens NFTs Resources Developers

Contract 0xBC4CA0EdA7647A8aB7C2061c2E118A18a936f13D Buy Exchange

Bored Ape Yacht Club: BAYC Token Source Code # Token Contract

Overview

ETH BALANCE  
0.010201 ETH

ETH VALUE  
\$15.99 (@ \$1,567.07/ETH)

TOKEN HOLDINGS  
\$407.65 (32 Tokens)

More Info

PRIVATE NAME TAGS  
+ Add

CONTRACT CREATOR  
Bored Ape Yacht Club: ... at txn 0x22199329b0aa1aa6...

TOKEN TRACKER  
BoredApeYachtClub (BAYC)

Multi Chain

MULTICHAIN ADDRESSES  
8 addresses found via Blockscan

Transactions Internal Transactions Token Transfers (ERC-20) NFT Transfers **Contract** Events Analytics Comments

Latest 25 from a total of 70,951 transactions (+3 Pending)

Transaction Hash	Method	Block	Age	From	To
0x2c89d45fdeda72532...	Set Approval ...	(pending)	21 hrs 16 mins ago	0x759a40...9506d657	Bored Ape Yacht Club:...
0x5a57c9b1fdacf3b					
0x2e8258318c305b17...	Set Approval ...	(pending)	12 days 4 hrs ago	0x759a40...9506d657	Bored Ape Yacht Club:...

This website uses cookies to improve your experience. By continuing to use this website, you agree to its Terms and Privacy Policy. Got it!

## 6: Contractページ

Transactions Internal Transactions Token Transfers (ERC-20) NFT Transfers **Contract** Events Analytics Comments

Code Read Contract Write Contract

Contract Source Code Verified (Exact Match)

Contract Name: BoredApeYachtClub Optimization Enabled:

Compiler Version v0.7.0+commit.9e61f92b Other Settings:

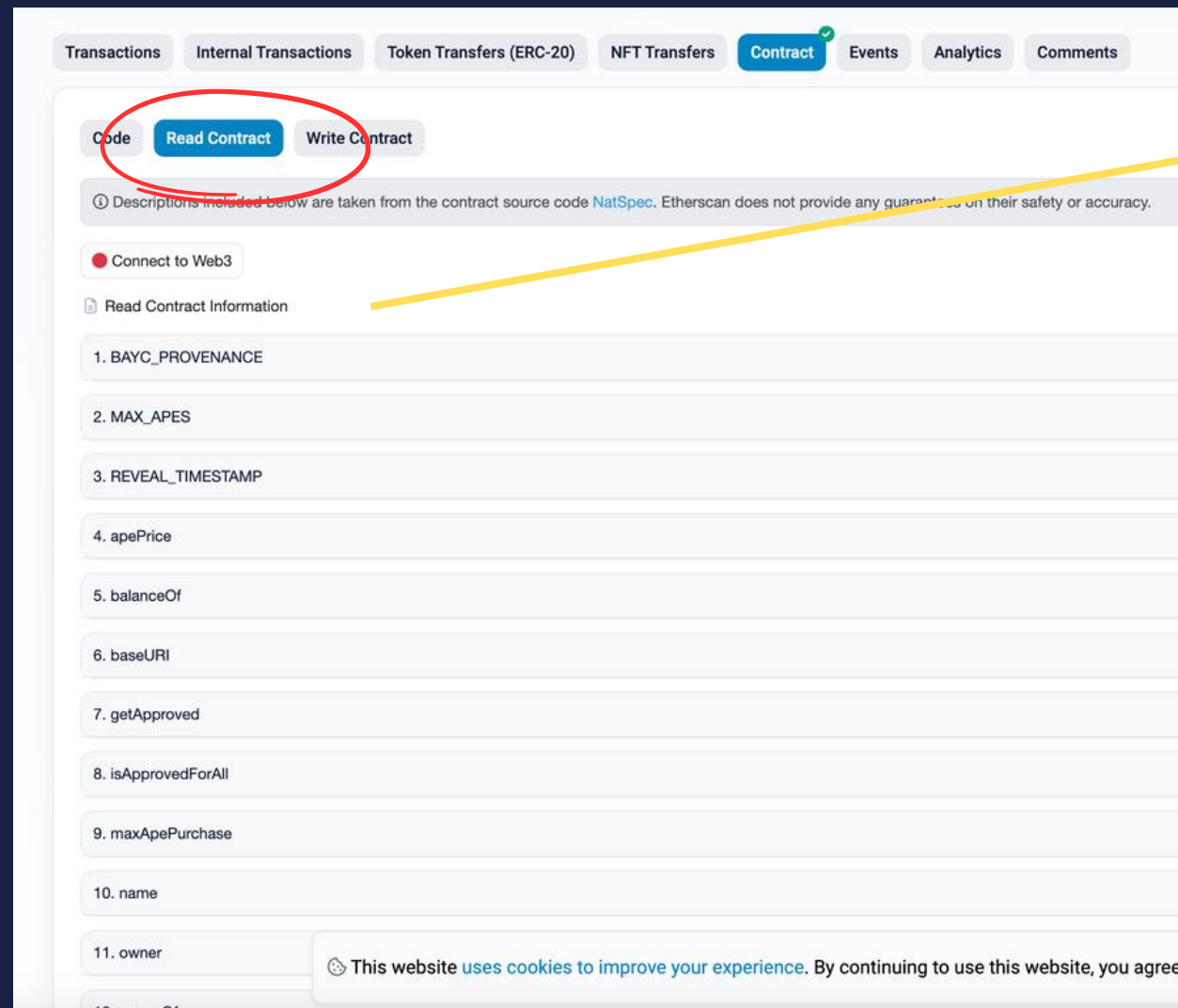
Contract Source Code (Solidity)

```
1 /**
2  *Submitted for verification at Etherscan.io on 2021-04-22
3  */
4
5  // File: @openzeppelin/contracts/utils/Context.sol
6
7  // SPDX-License-Identifier: MIT
8
9  pragma solidity >=0.6.0 <0.8.0;
10
11 /**
12  * @dev Provides information about the current execution context, including the
13  * sender of the transaction and its data. While these are generally available
14  * via msg.sender and msg.data, they should not be accessed in such a direct
15  * manner, since when dealing with GSN meta-transactions the account sending and
16  * paying for execution may not be the actual sender (as far as an application
17  * is concerned).
18  *
19  * This contract is only required for intermediate, library-like contracts.
20  */
21 abstract contract Context {
22     function _msgSender() internal view virtual returns (address payable) {
23         return msg.sender;
24     }
25 }
```

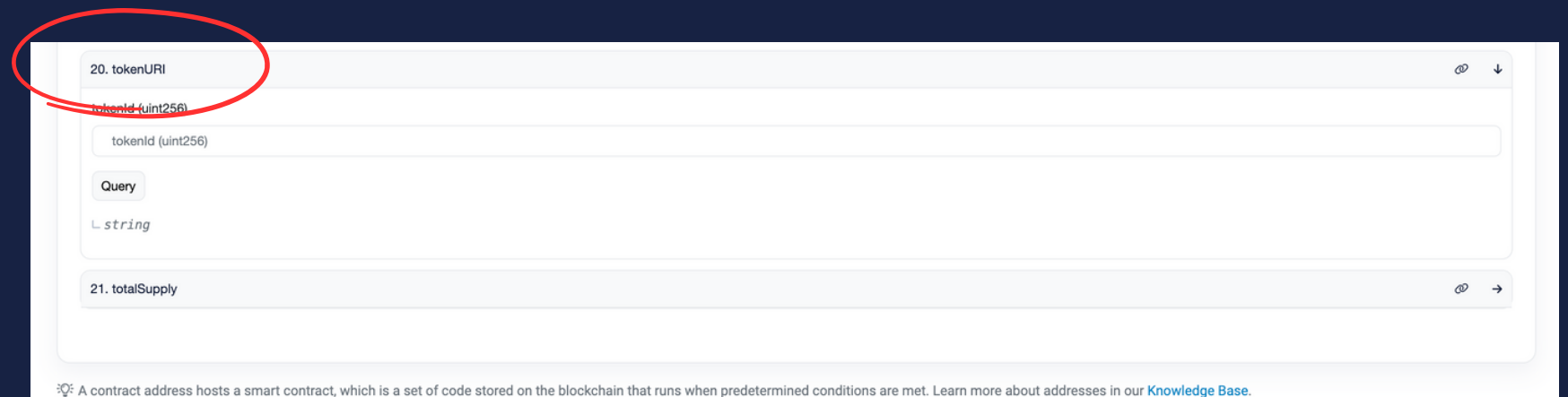
This website uses cookies to improve your experience. By continuing to use this website, you agree to its Terms and Privacy Policy. Got it!

\*Etherscan: Ethereum上の取引履歴を見ることのできるツール

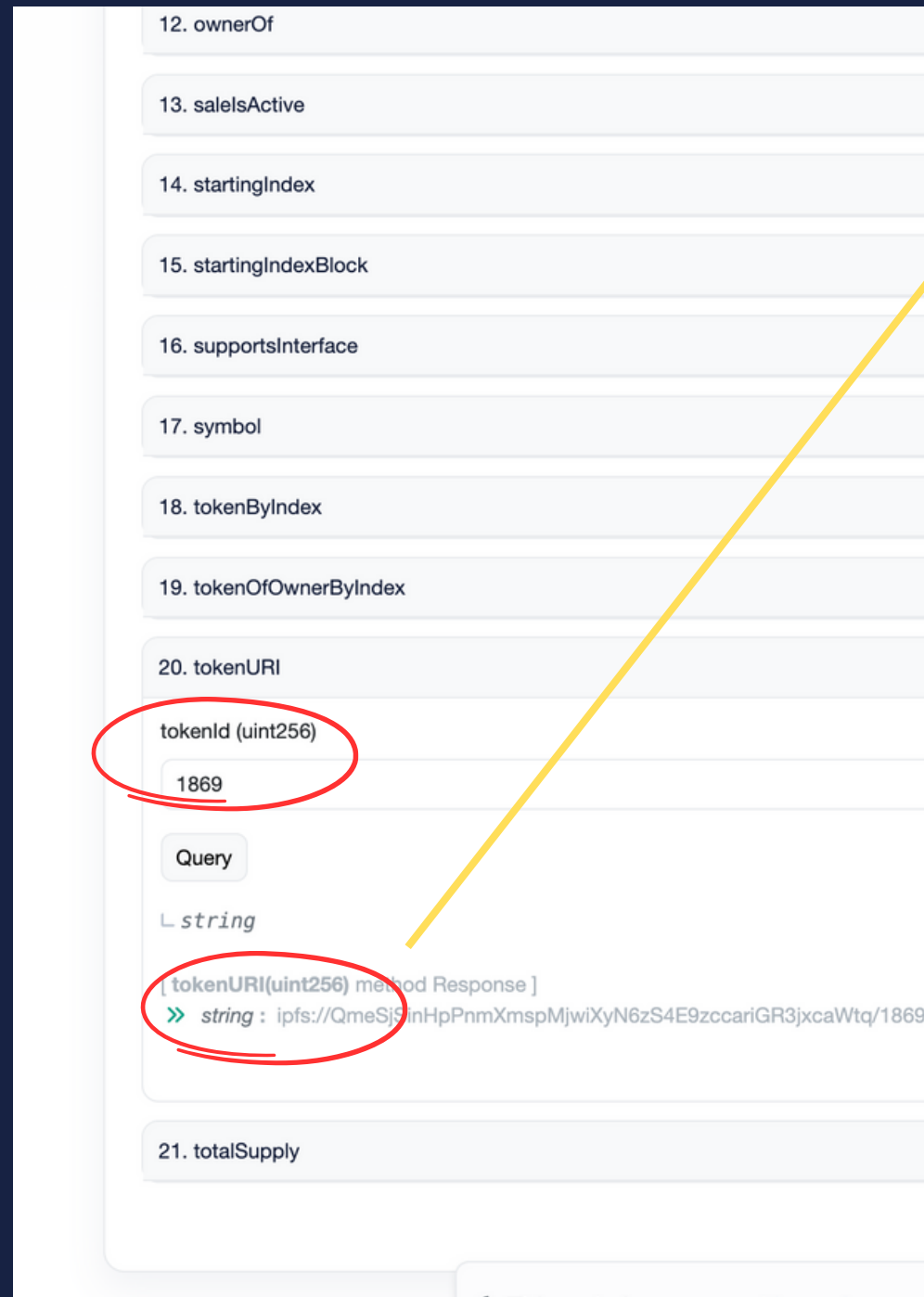
## 7: Read Contractをクリック



## 8: 20のtokenURIをクリック



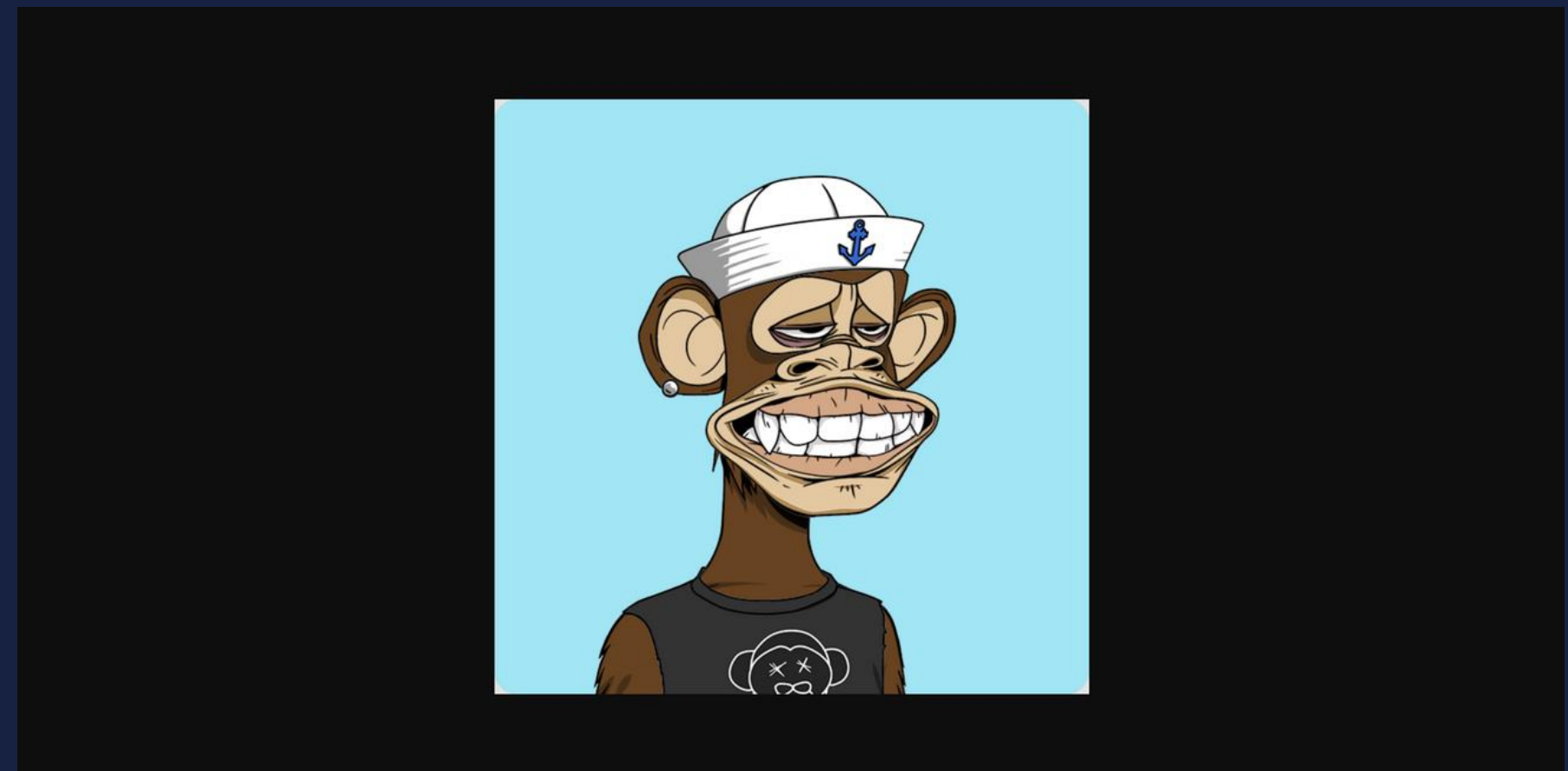
9:4でメモしたToken IDを入力



10: stringに記載してあるipfs://で検索

```
{  
  "image": "ipfs://QmeFA7hXyGav7DXrnEpU4GyUCTXAsfsiydyvfkPL5JB5W5",  
  "attributes": [  
    {"trait_type": "Earring", "value": "Silver Stud"},  
    {"trait_type": "Clothes", "value": "Sleeveless Logo T"},  
    {"trait_type": "Hat", "value": "Seaman's Hat"},  
    {"trait_type": "Background", "value": "Blue"},  
    {"trait_type": "Mouth", "value": "Grin"},  
    {"trait_type": "Fur", "value": "Dark Brown"},  
    {"trait_type": "Eyes", "value": "Sleepy"}  
  ]  
}
```

11: 10のJSONデータ内"image"に続くIPFSリンクを検索→画像表示



# 中央サーバー型NFTの場合

The screenshot shows the OpenSea interface for an NFT listing. On the left is a large image of a pink and white cartoon character wearing a black headband and holding a green leaf and a brush. The right side contains the listing details:

- Collection: CryptoNinja Partners
- Item Name: Leelee-Pink(dango) #16101
- Owned by: Dr\_KC
- Stats: 100 views, 6 favorites, PFPs
- Sale ends: September 12, 2023 at 3:53 PM
- Current price: 1.5979 ETH (\$2,661.38)
- Buttons: Add to cart, Make offer
- Price History: A bar chart showing a single transaction on May 15, 2022, with a volume of approximately 0.06 ETH and an average price of about 3.5 ETH.

<https://opensea.io/assets/ethereum/0x845a007d9f283614f403a24e3eb3455f720559ca/16101>

ETH Price: \$1,661.97 (+0.11%) Gas: 28 Gwei Search by Address / Txn

- 11. mintCost
- 12. name
- 13. owner
- 14. ownerOf
- 15. paused
- 16. supportsInterface
- 17. symbol
- 18. tokenURI

tokenId (uint256)  
16101

Query  
string

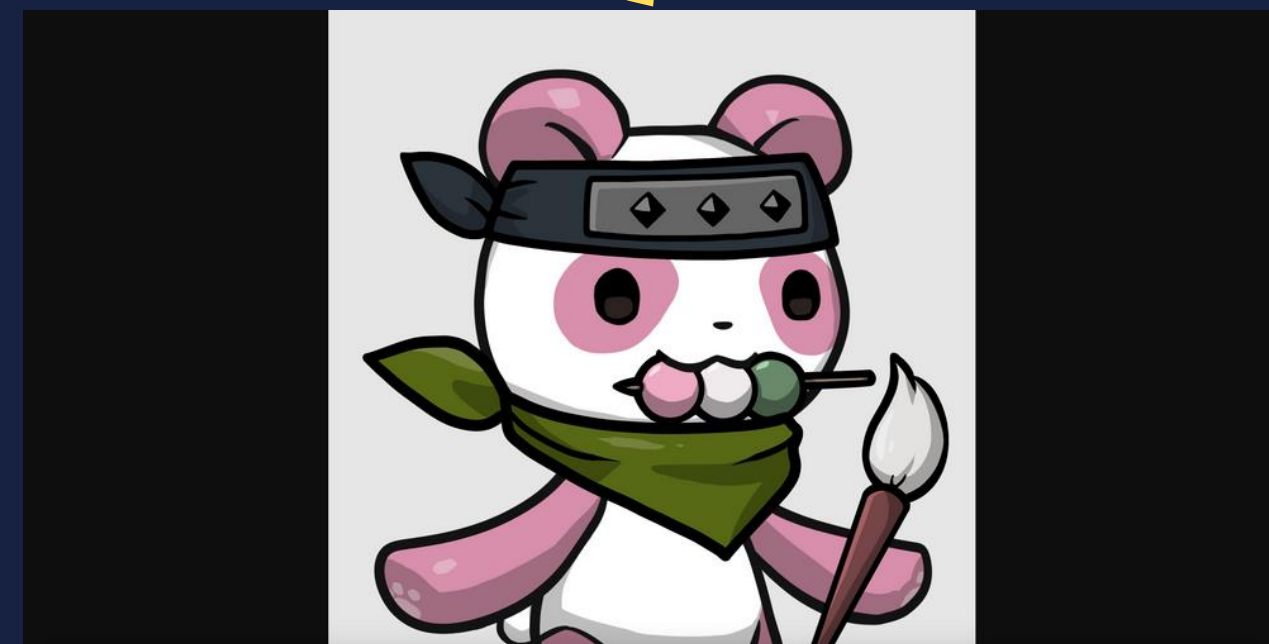
[ tokenURI(uint256) method Response ]  
>> string : https://data.cryptoninjapartners.com/vjson/16101.json

19. totalSupply  
Burned tokens are calculated here, use `_totalMinted()` if you want to count just minted tokens.  
22222 uint256

20. withdrawAddress

This website uses cookies to improve your experience. By continuing to use this website, you agree to

```
{  
  "name": "Leelee-Pink(dango) #16101",  
  "description": "The panda \"Leelee\" is Xiaolan's partner in CryptoNinja.They seem to  
  一。 \n任務を遂行するため、さまざまな姿に変身しているようです...!",  
  "image": "https://data.cryptoninjapartners.com/images/16101.png",  
  "edition": 16101,  
  "attributes": [  
    {  
      "trait_type": "NINJUTSU",  
      "value": "None"  
    },  
    {  
      "trait_type": "WEAPON(BACK)",  
      "value": "None"  
    },  
    {  
      "trait_type": "CHARACTER",  
      "value": "Leelee"  
    },  
    {  
      "trait_type": "CLAN",  
      "value": "Koka"  
    },  
    {  
      "trait_type": "COSPLAY",  
      "value": "None"  
    }  
  ]  
}
```





---

## IPFSについてもう少し詳しく

- *IPFS*はコンテンツ指向型のハイパーメディア分散プロトコル
  - *IPFS*は*HTTP*をベースとした既存のネットワークとは独立したネットワーク  
→既存のブラウザからアクセスするには*IPFS*に橋渡ししてくれる存在が必要  
→*IPFS Gateway*
  - *IPFS Gateway*の中でも任意ユーザーに常時公開されている***IPFS Public Gateway*** (<https://ipfs.io/ipfs>など) を利用する必要がある
  - <https://ipfs.io/ipfs/~~~~>の『~~~~』部にコンテンツ (*Content*) の*ID* (画像等のコンテンツのハッシュ値) を入れる必要がある (*CID*)  
→逆に言えば、上記のコンテンツ*ID*を知っているものであれば閲覧可能
  - *P2P*ファイル共有システムによって耐障害性、耐検閲性、耐改ざん性がある
  - 詳細な仕組みについては添付リンク参照
-