Hands-on Part

*Metamask*のインストール

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。

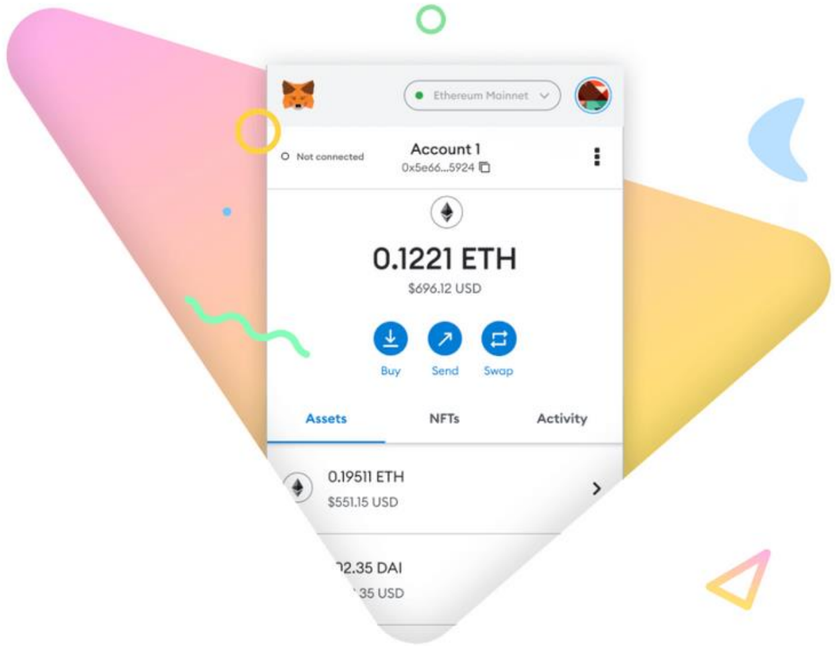
METAMASK

Features Build Support Learn Download

A crypto wallet & gateway to blockchain apps

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

Download for

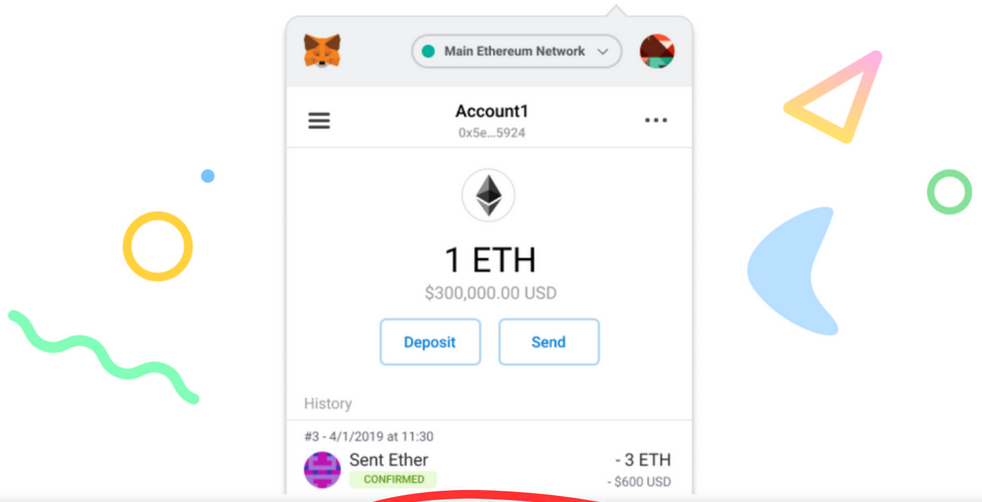


METAMASK

Features Build Support Learn

Chrome iOS Android

Install MetaMask for your browser



Install MetaMask for Chrome

chrome ウェブストア

ホーム > 拡張機能 > MetaMask

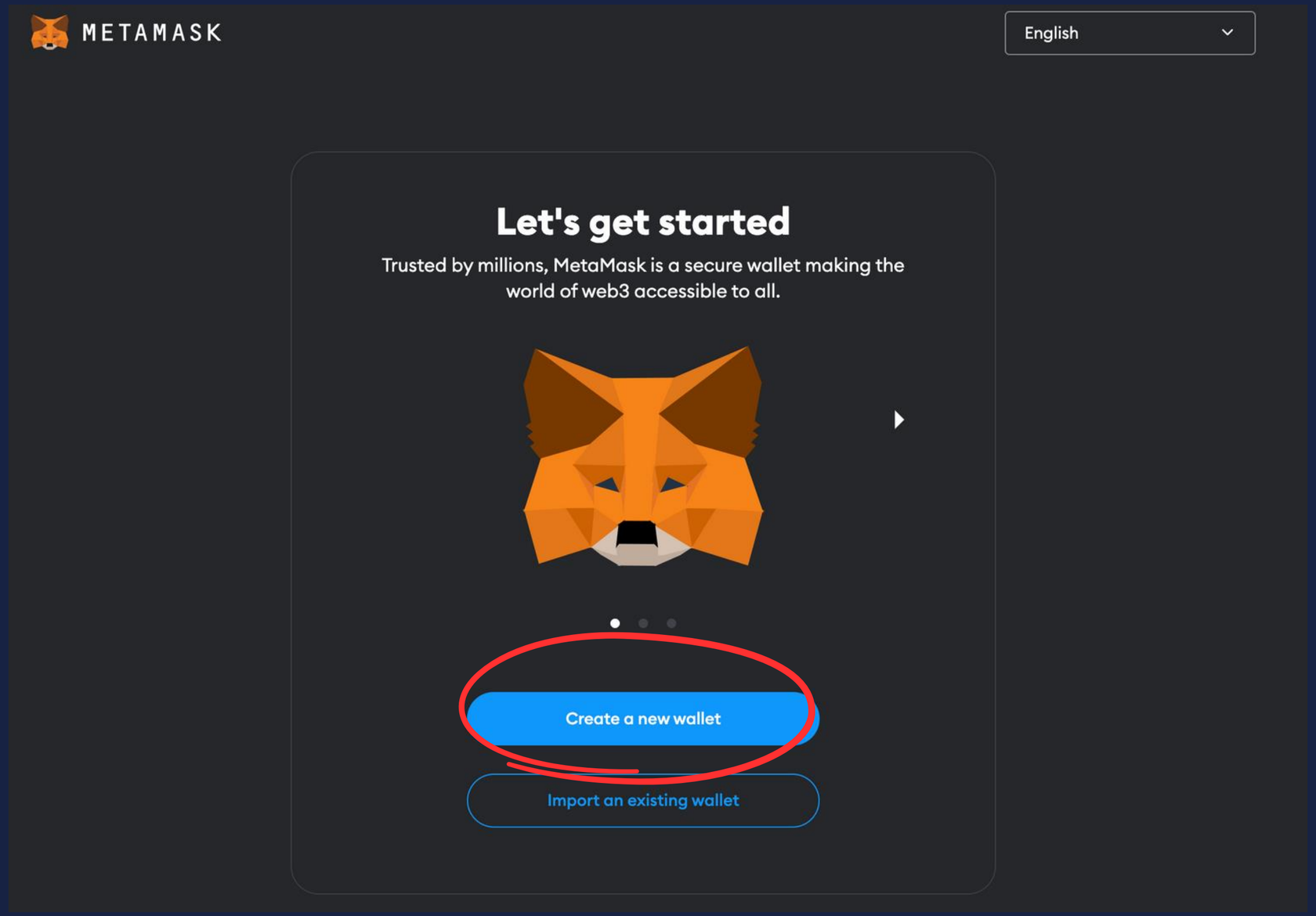
MetaMask

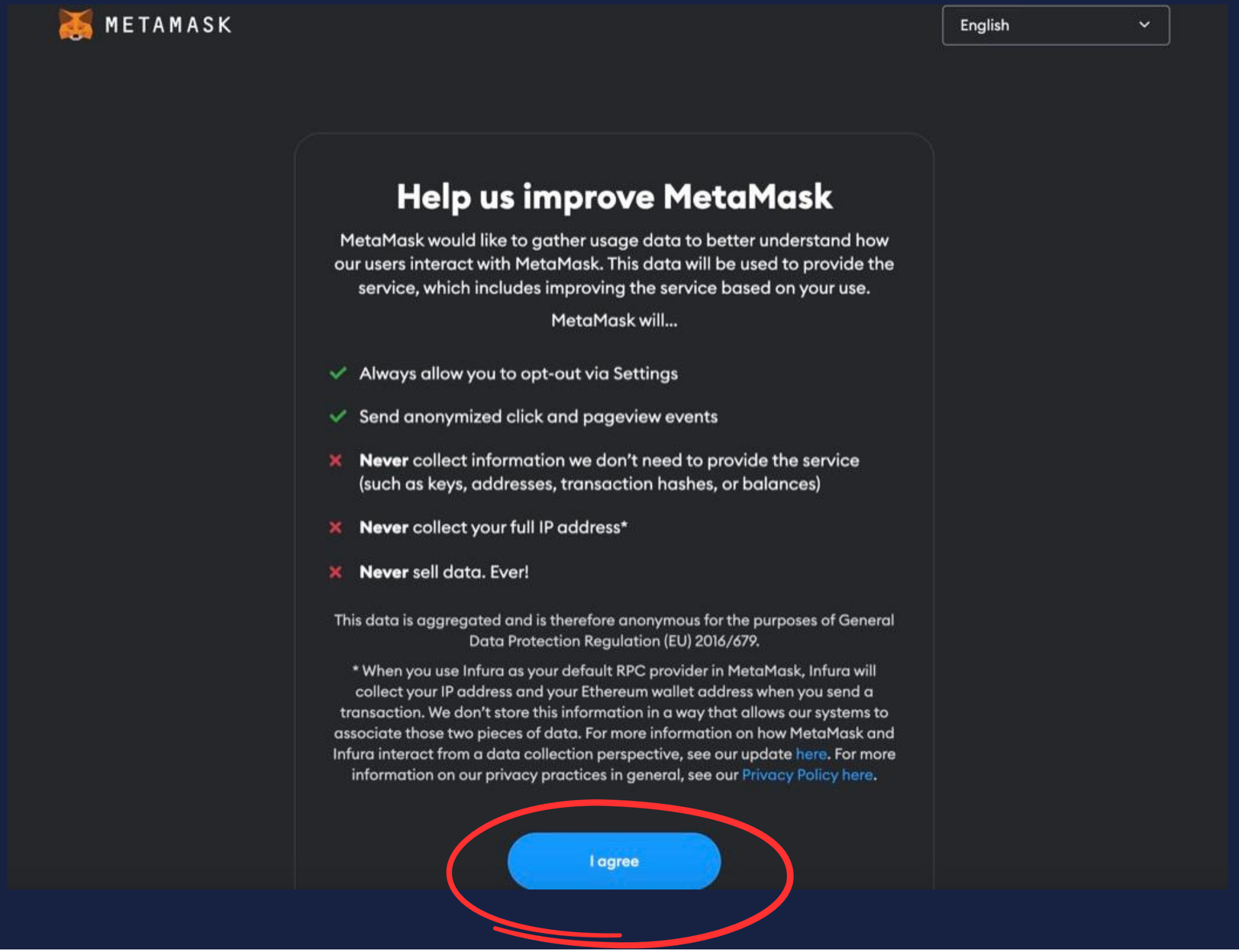
metamask.io

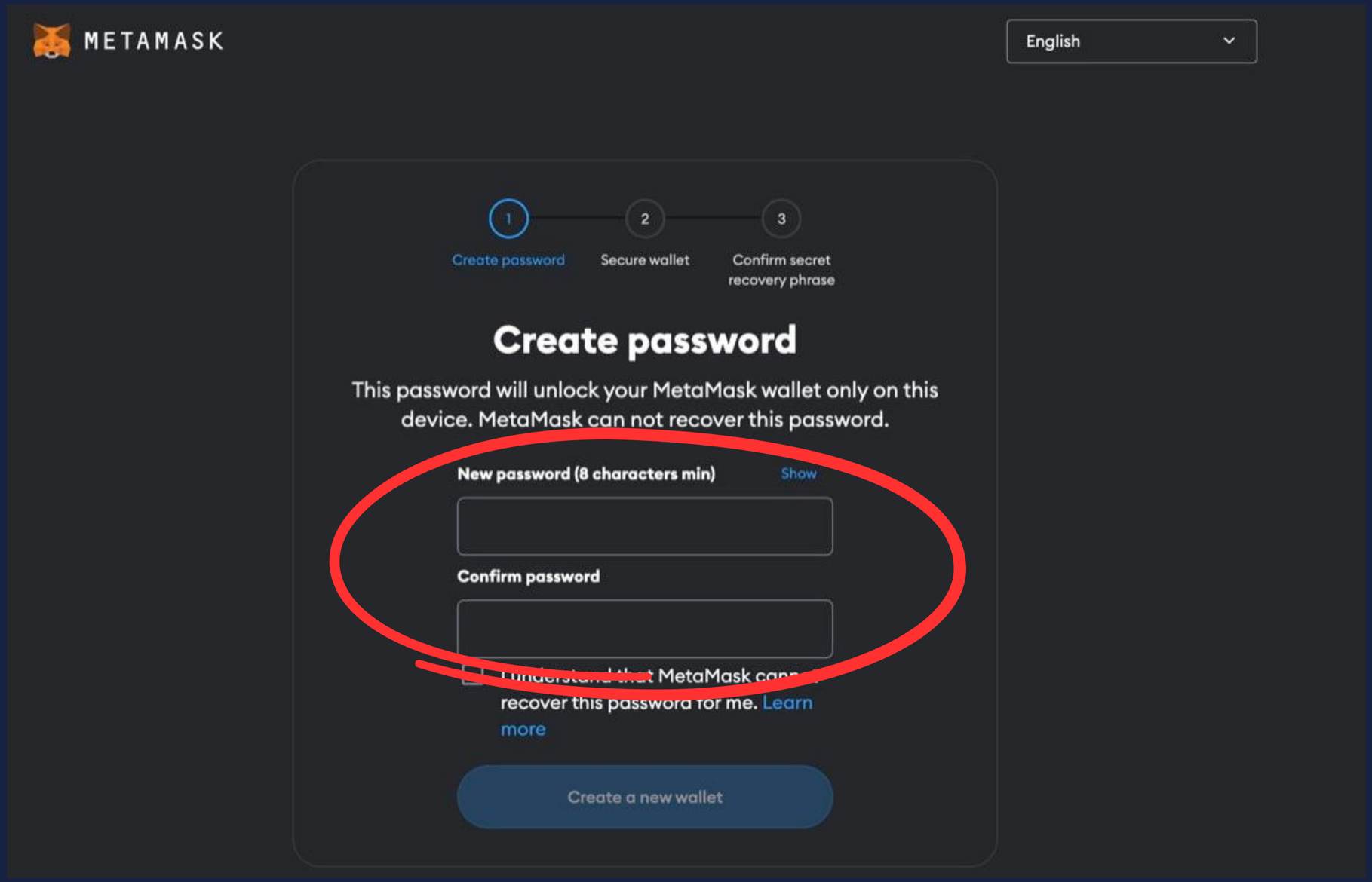
★★★½☆ 2,889 | 仕事効率化 | ユーザー数: 10,000,000+ 人

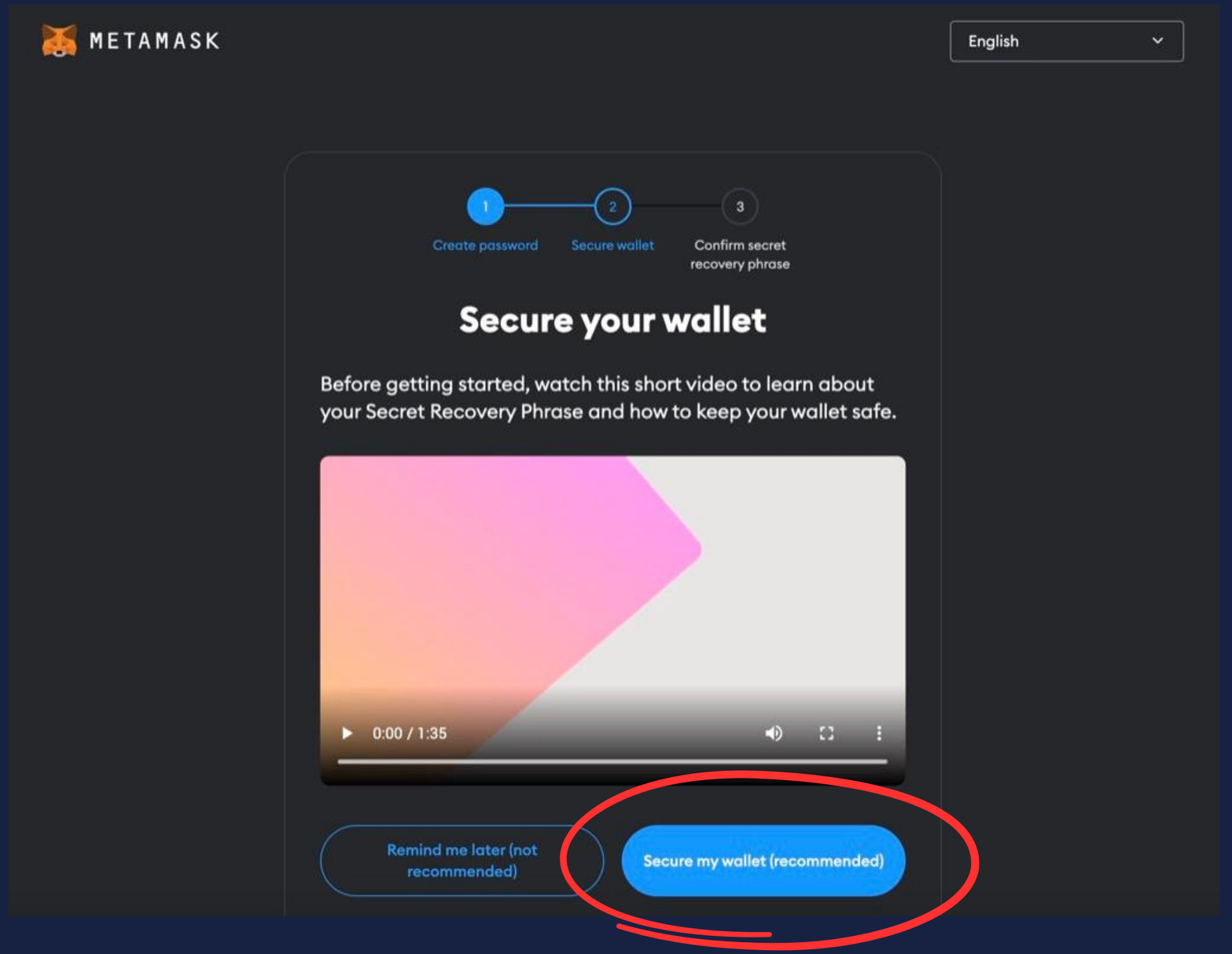
Chrome に追加

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


METAMASK
English
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


METAMASK
English
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


METAMASK
English
1
2
3
Create password
Secure wallet
Confirm secret recovery phrase
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
recover this password for me. Learn more
Create a new wallet


METAMASK
English
1
2
3
Create password
Secure wallet
Confirm secret recovery phrase
Secure your wallet
Before getting started, watch this short video to learn about your Secret Recovery Phrase and how to keep your wallet safe.
0:00 / 1:35
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

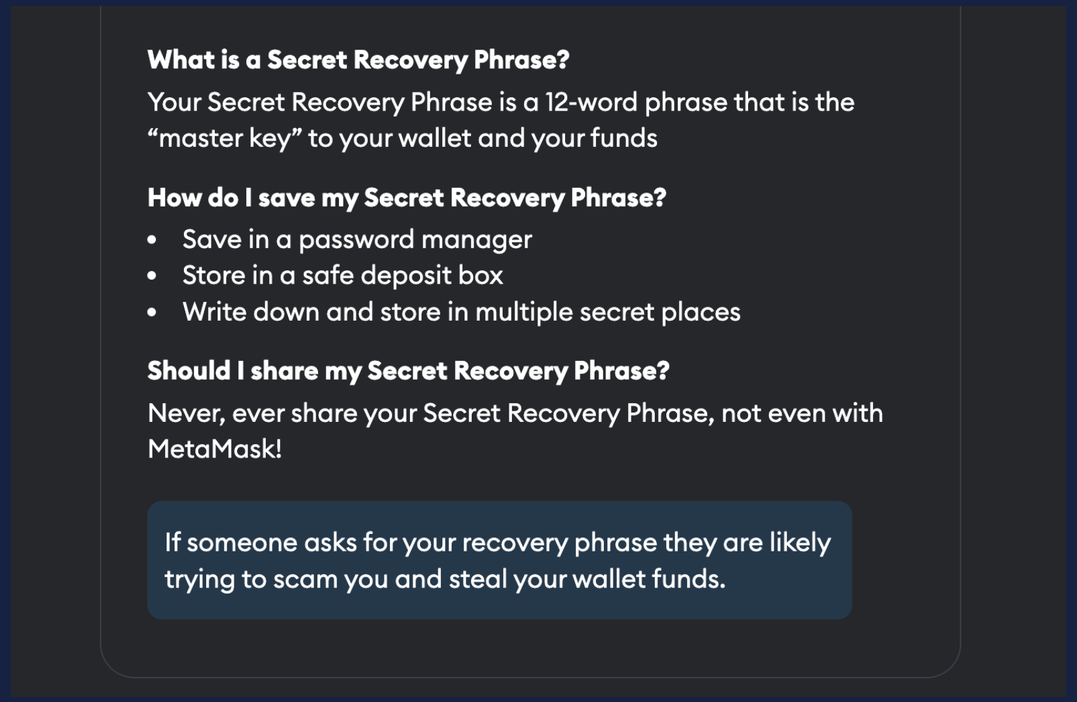
How do I save my Secret Recovery Phrase?

- 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.



Secret Recovery Phraseとは何か？

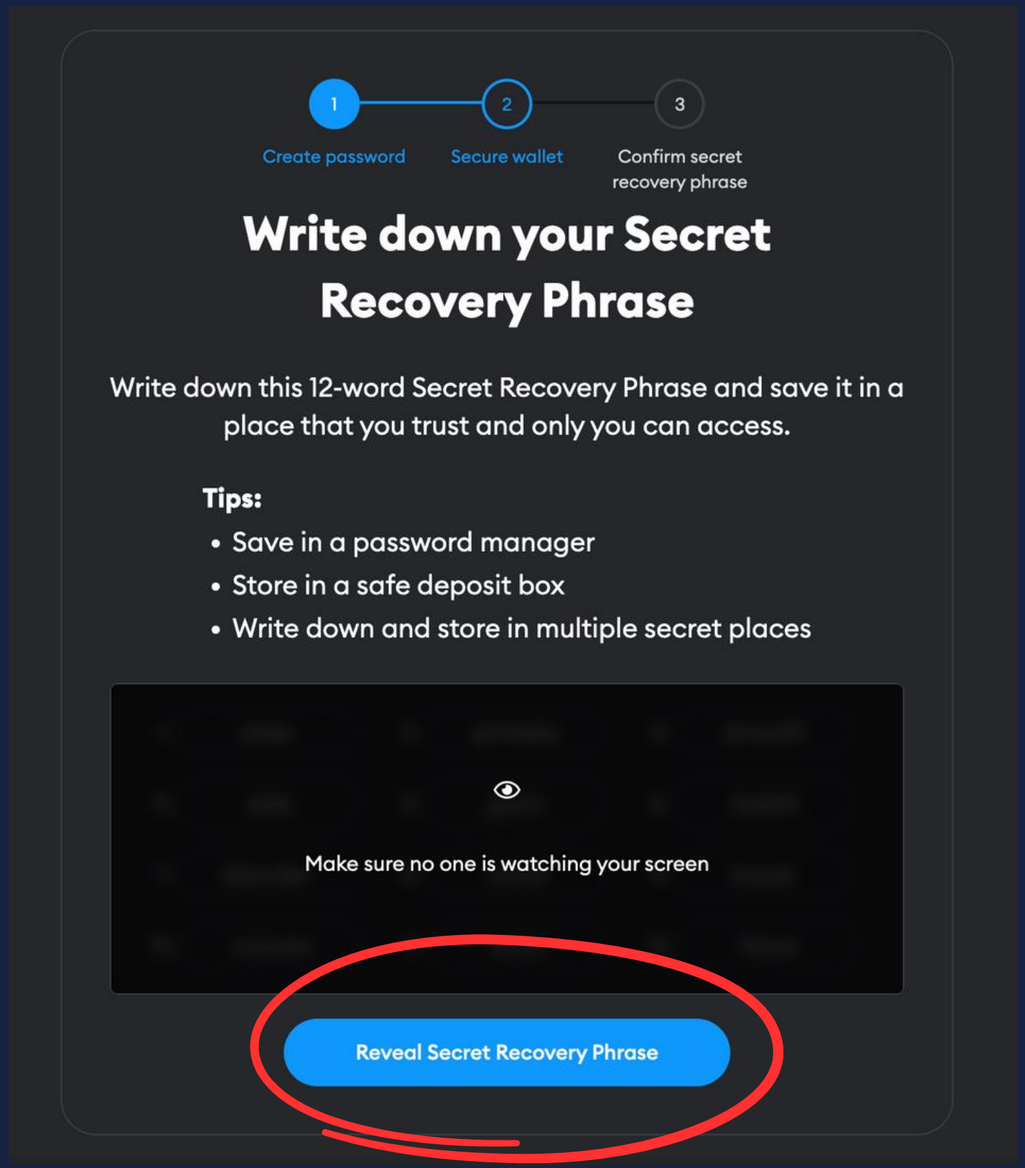
あなたのSRPは12単語で、これはウォレットと資産のMaster Keyです。

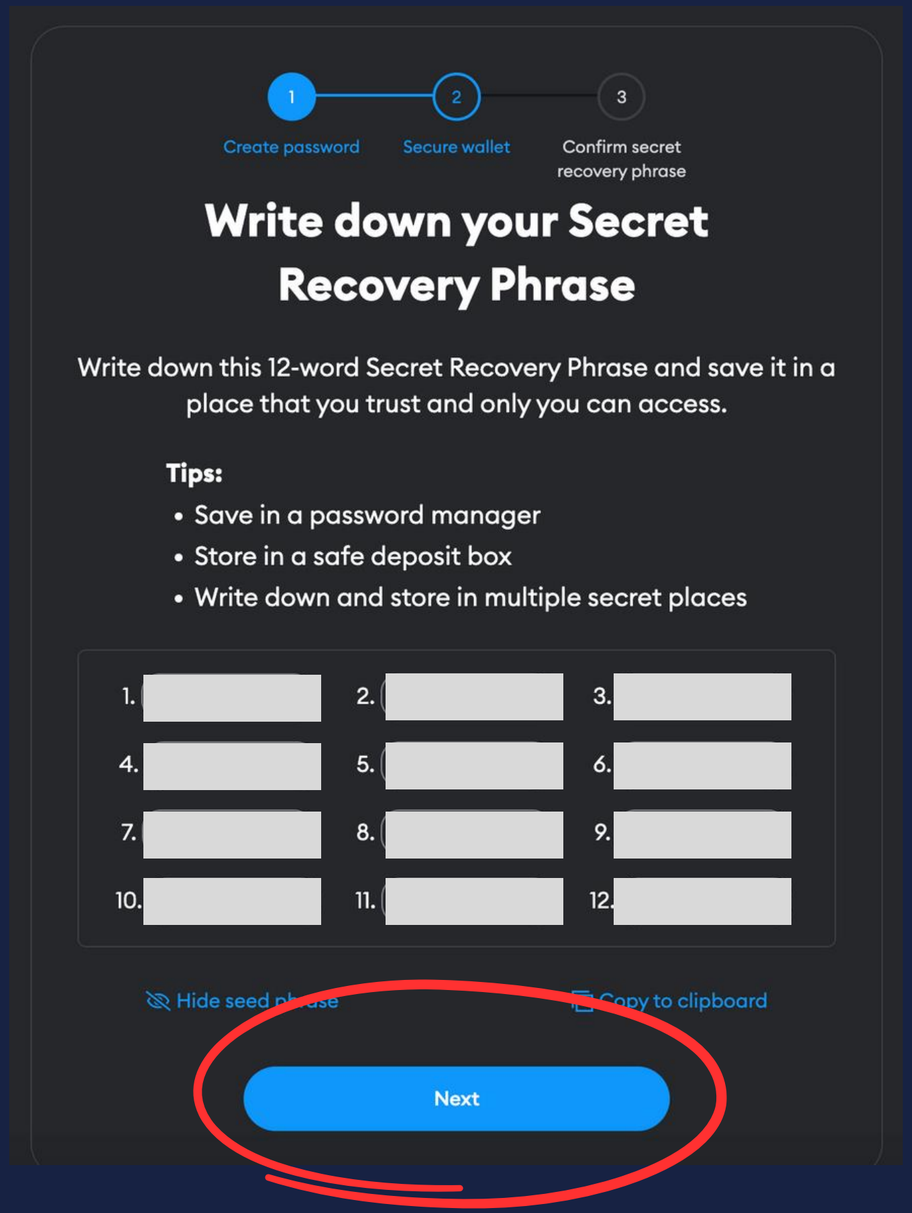
どのようにSRPを保管するか？

・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

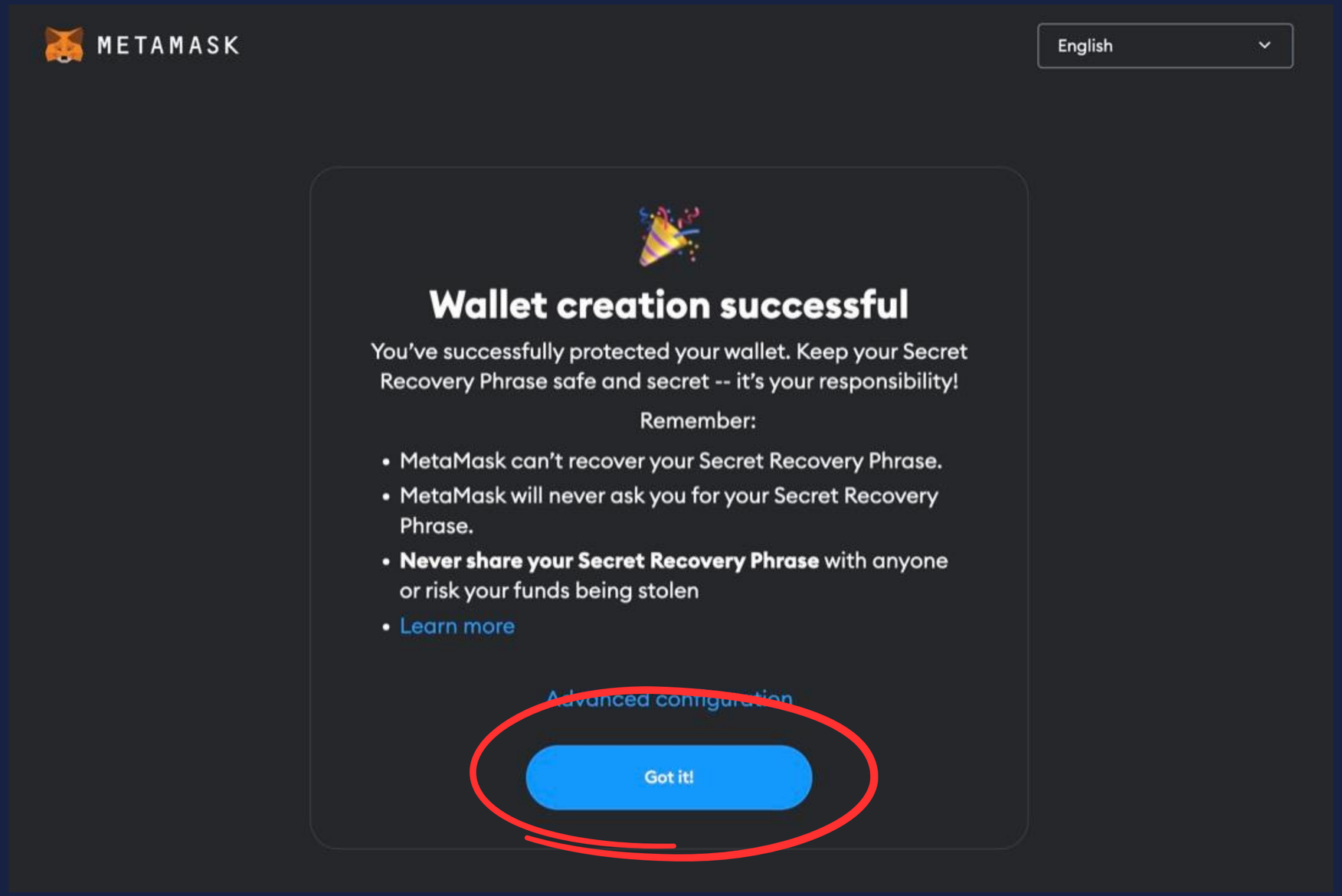
SRPを他の人と共有することはあるか？

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
2
3
Create password
Secure wallet
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


1
2
3
Create password
Secure wallet
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
1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
Hide seed phrase
Copy to clipboard
Next


1
2
3
Create password
Secure wallet
Confirm secret recovery phrase
Confirm Secret Recovery Phrase
Confirm Secret Recovery Phrase
1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
Confirm


METAMASK
English
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
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-Fungiblity*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ると考えられる。
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

```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

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)

ERC721 Metadata

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": "Mjölnir, 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

保有者とアイテムのメタデータのクエリ

*tokuneURI*の*JSON*ファイル例

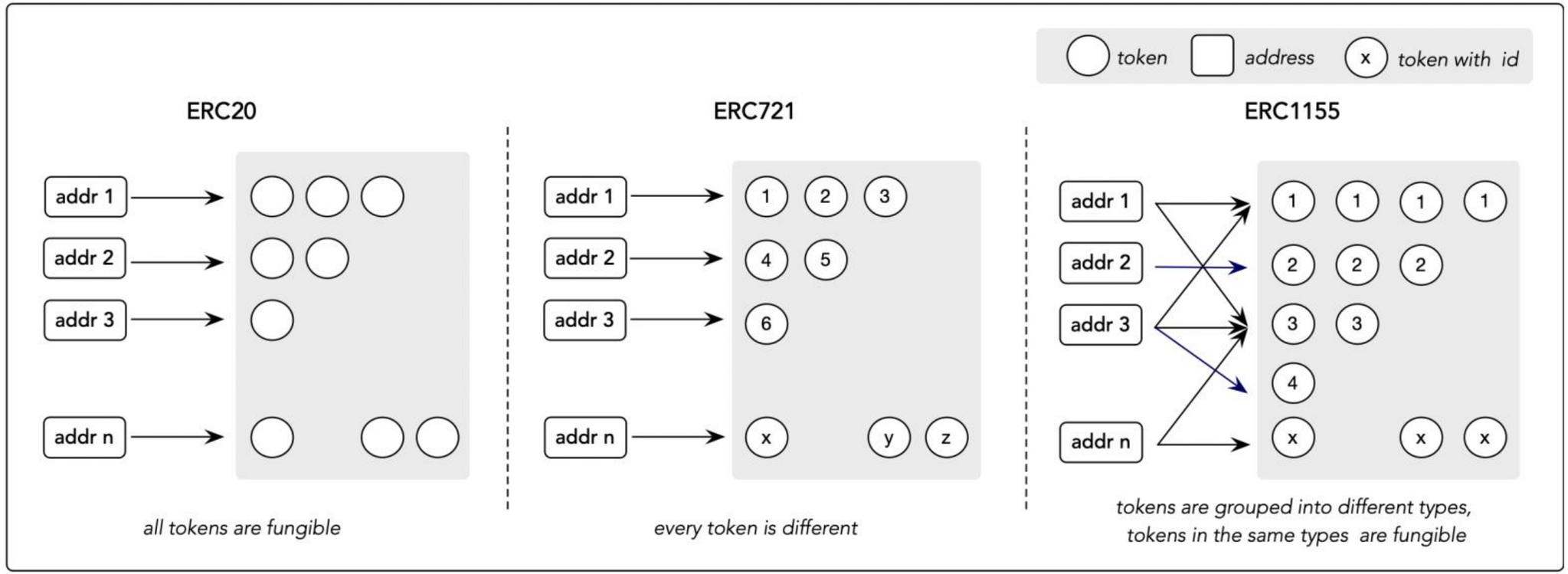


Fig. 2: NFT-related Token Standards

出典：Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges

出典： Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges

ERC721のtoken URIを見てみよう

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

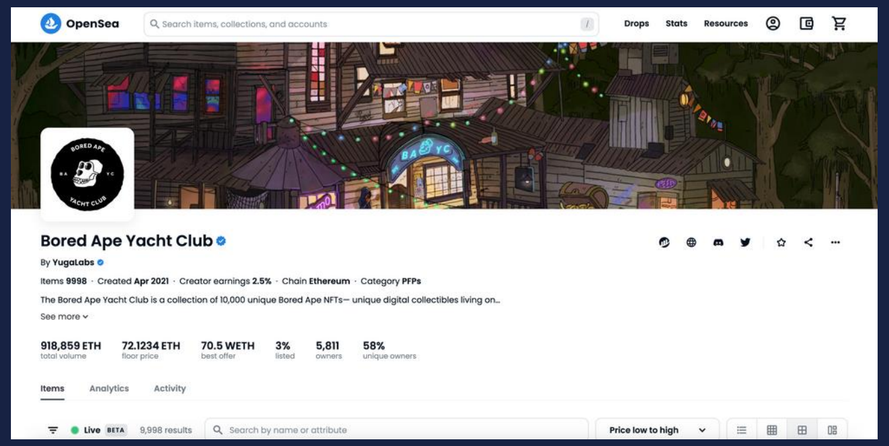
IPFSによるホストが可能。

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。

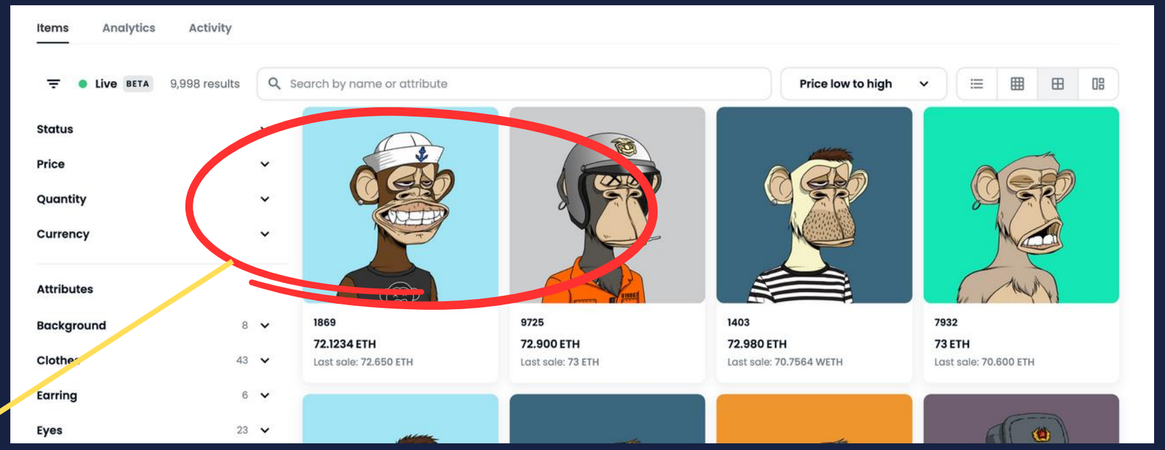
手を動かして確認してみよう
（IPFS/中央サーバー）

IPFS型NFTの場合

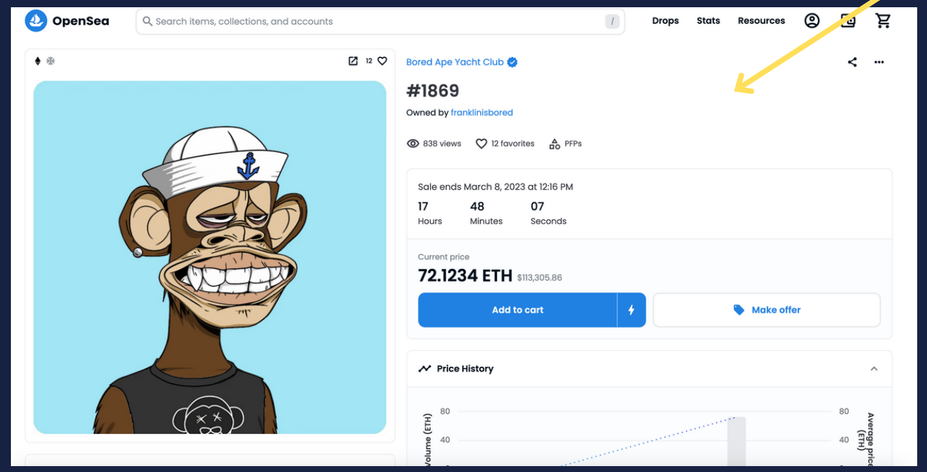
1：OpenSea*



2：一つのNFTを選択

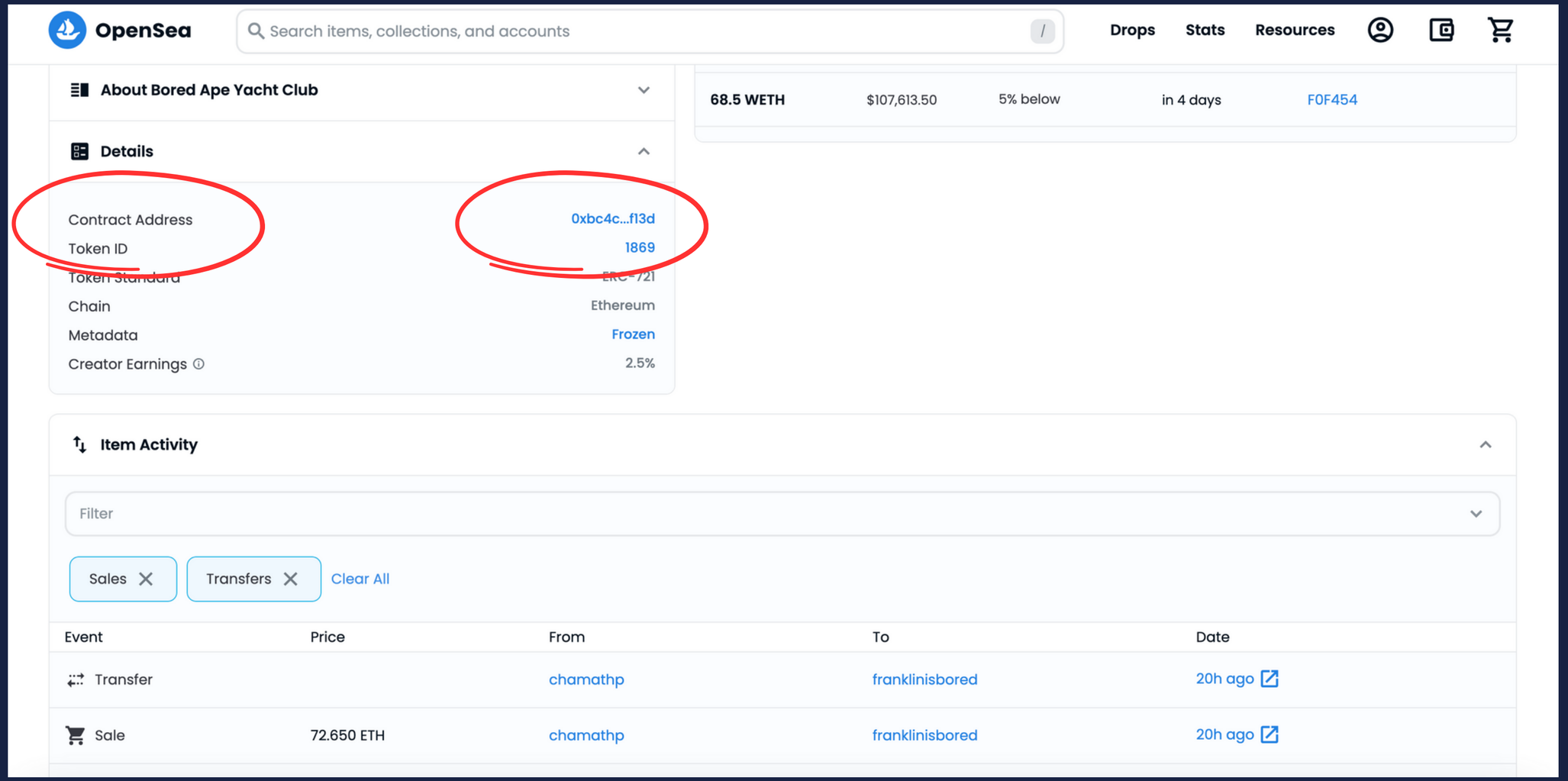


3：コンテンツページ→スクロール

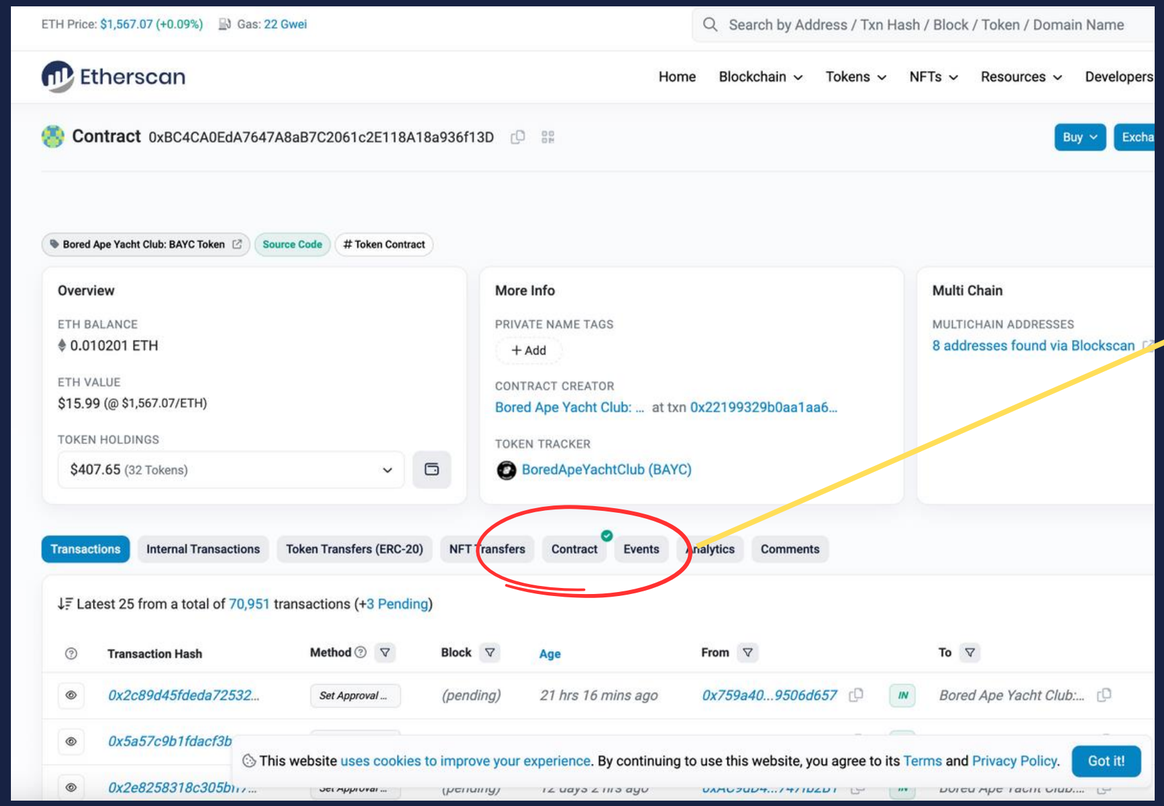


*OpenSea
NFTを販売しているECサイト（的な）
今回はBAYCのページを利用。
→https://opensea.io/collection/boredapeyachtclub

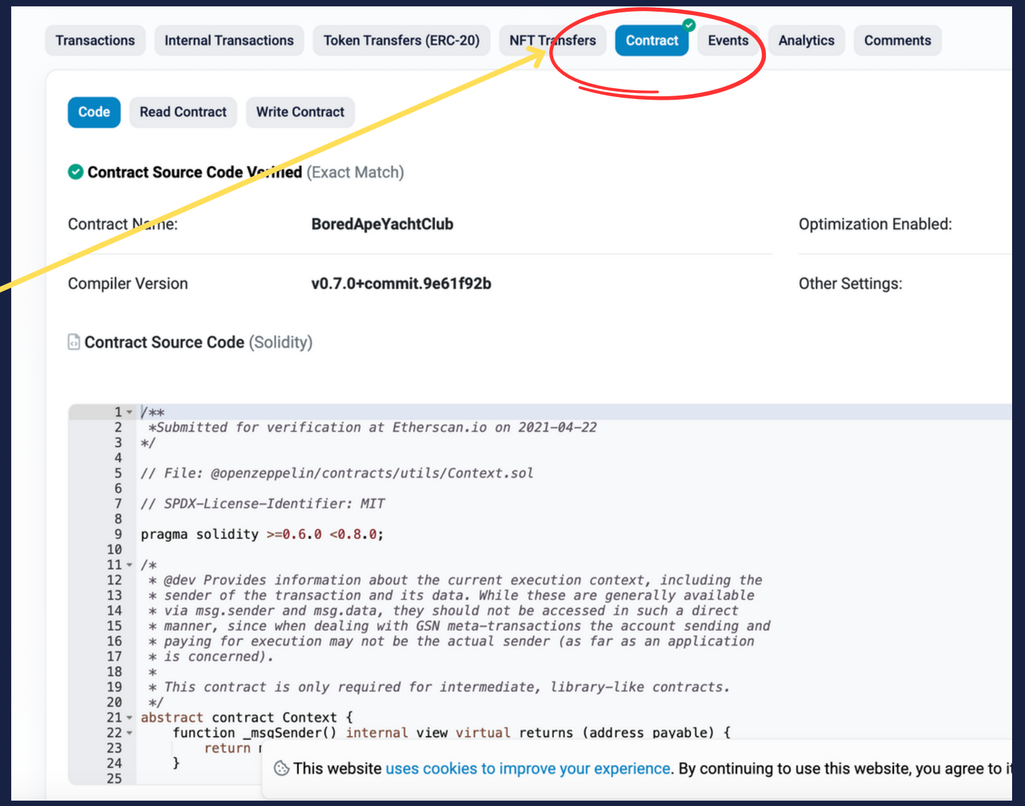
4：Detailsを参照→0xbc4c...f13dをクリック（Token IDもメモ）



5：Etherscan*に遷移

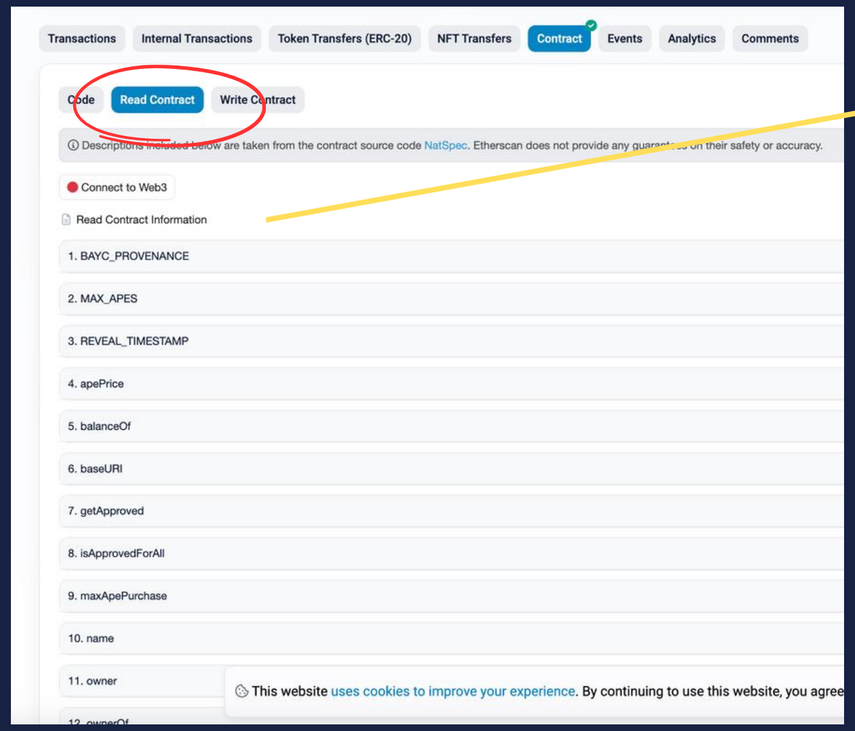


6：Contractページ

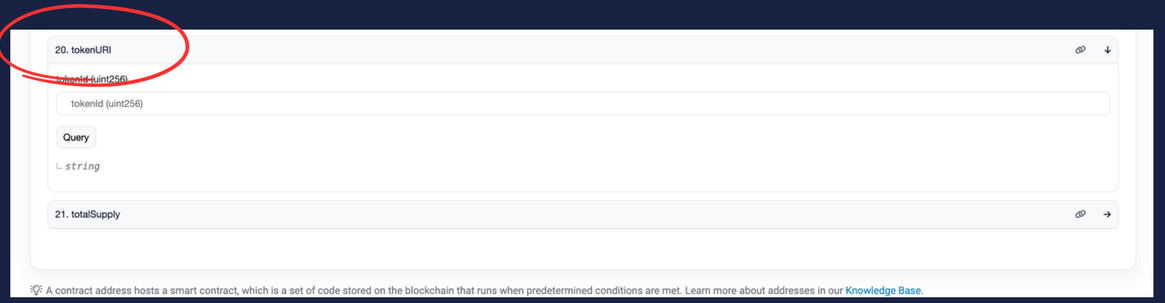


*Etherscan：Ethereum上の取引履歴を見ることのできるツール

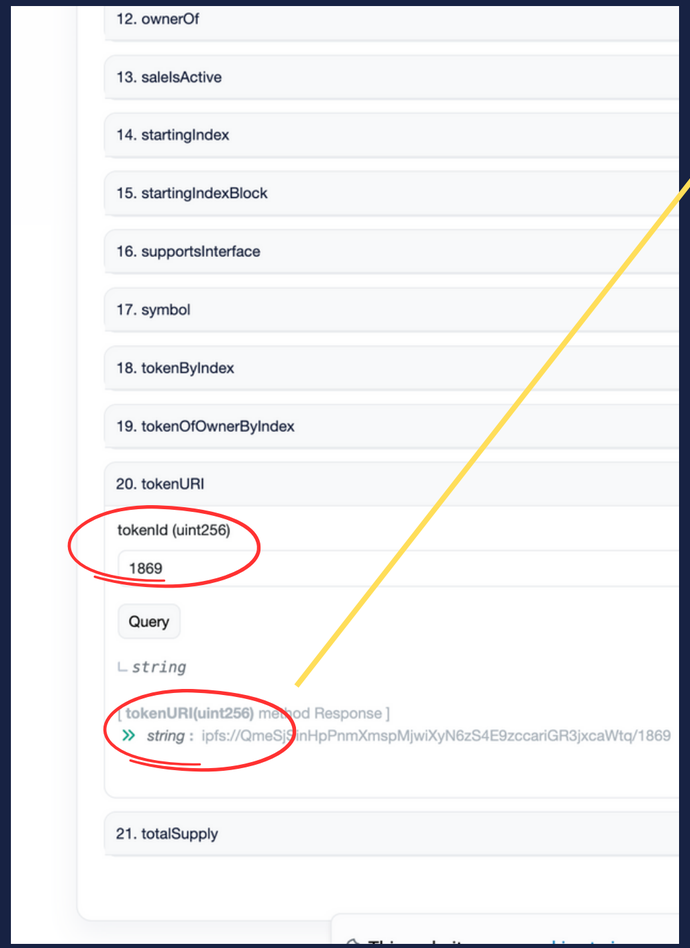
7：Read Contractをクリック



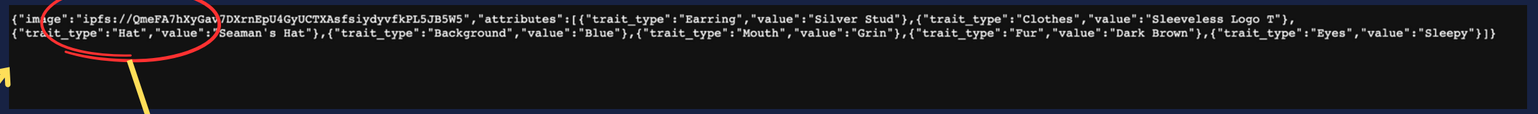
8：20のtokenURIをクリック



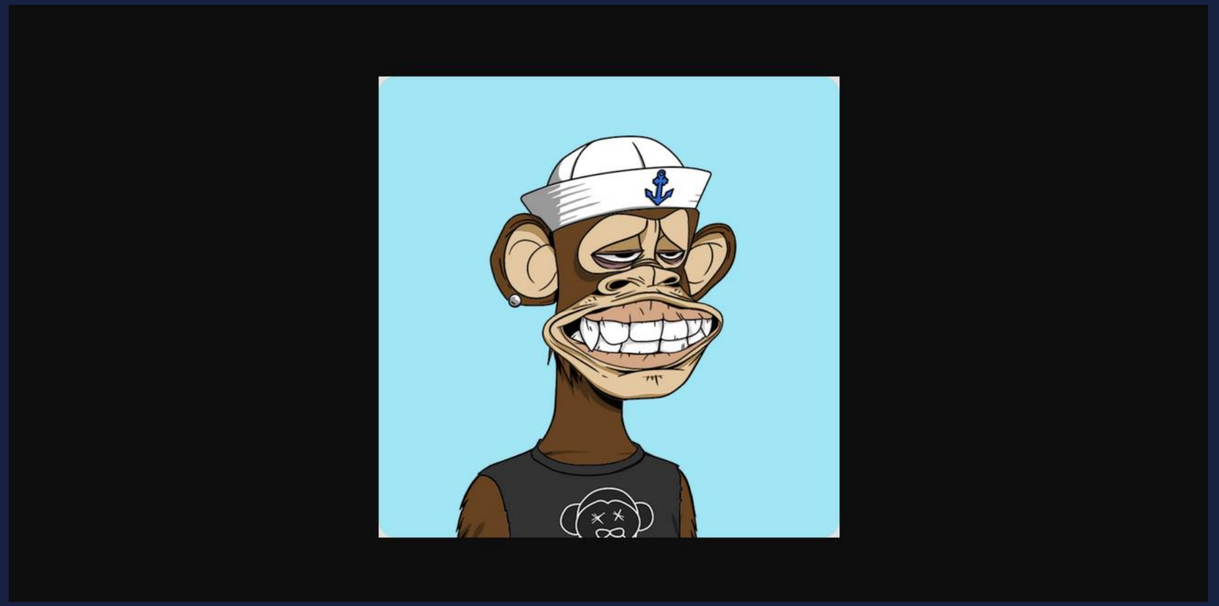
9：4でメモしたToken IDを入力



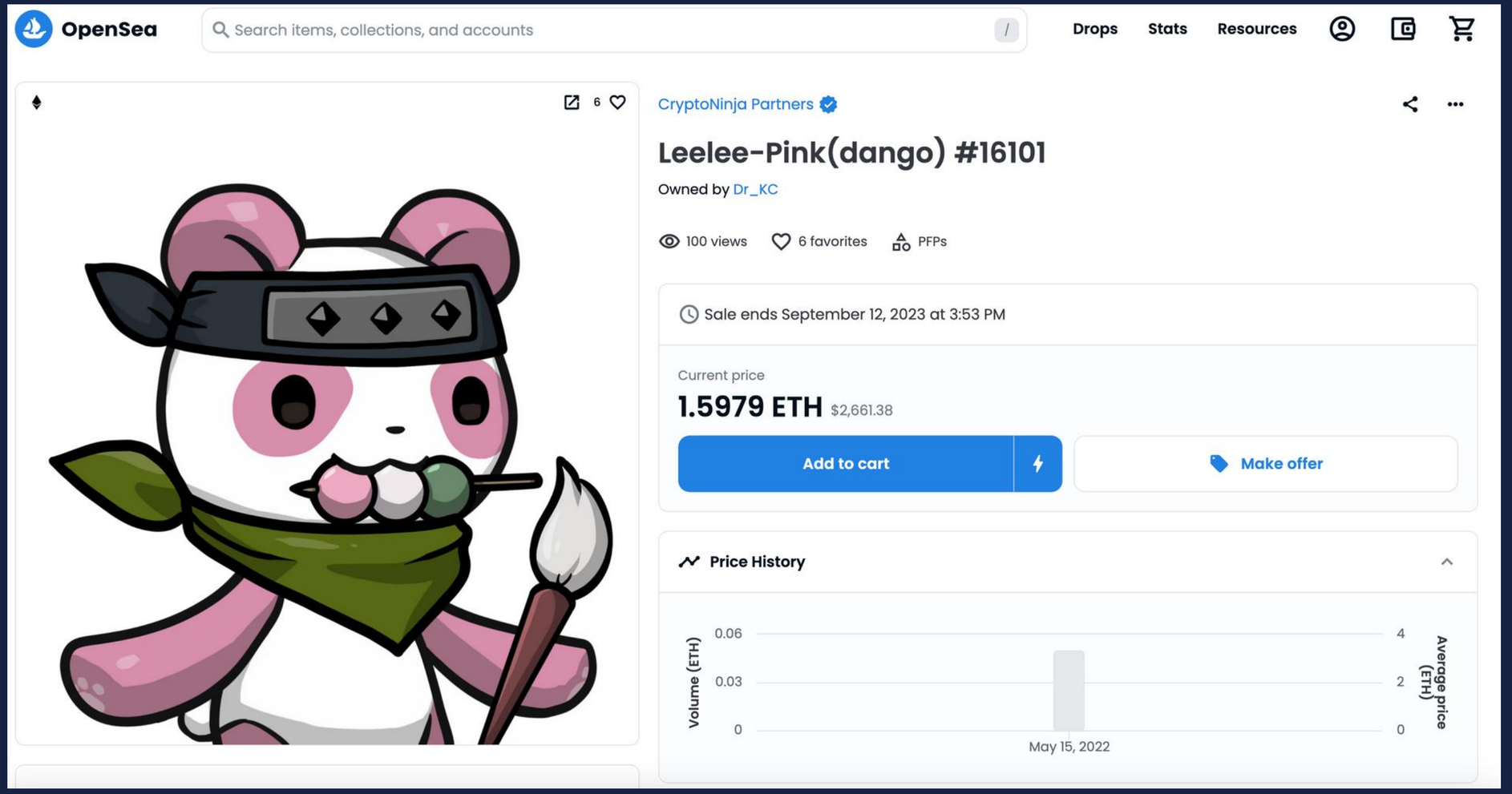
10：stringに記載してあるipfs://~で検索



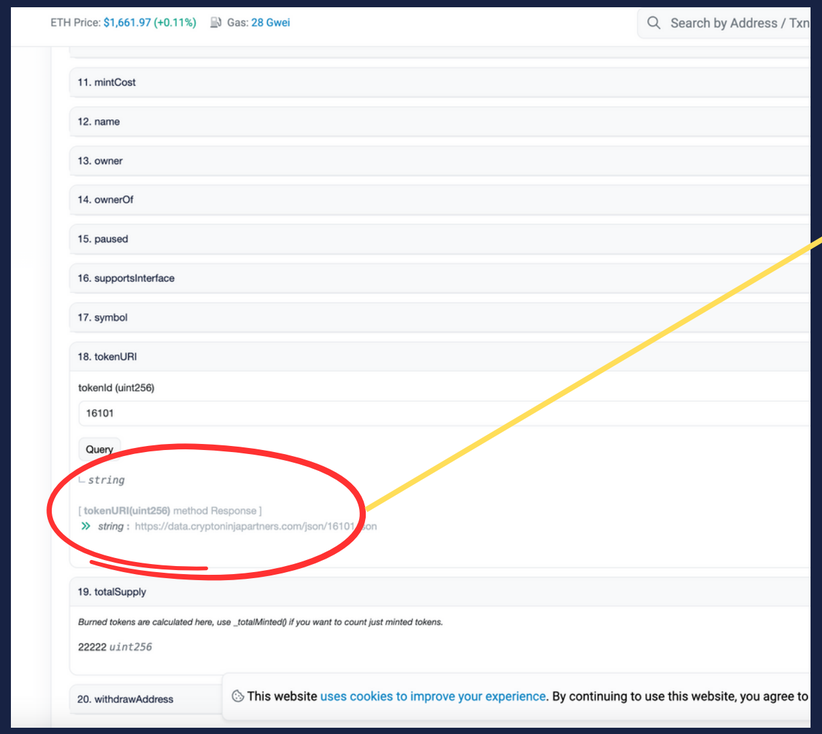
11：10のJSONデータ内"image"に続くIPFSリンクを検索→画像表示

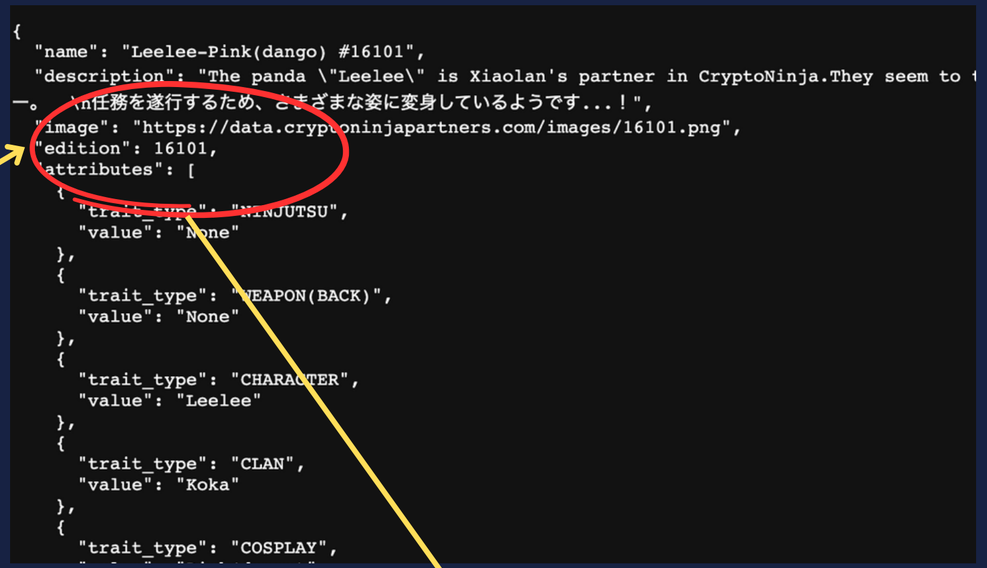


中央サーバー型NFTの場合



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/json/16101
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
ー。 \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",

*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