نویسنده

حمیدرضا عسگری

asgari@coiniran.com

عضو تیم کوین ایران از سال ۱۳۹۶

مسئولیت ها در تیم کوین ایران از ابتدا تاکنون:

- نویسنده مقالات خبری
- تولید محتواهای آموزشی
- سرپرست تحریریه
- مدیر وب سایت
- CTO

منتشر شده توسط

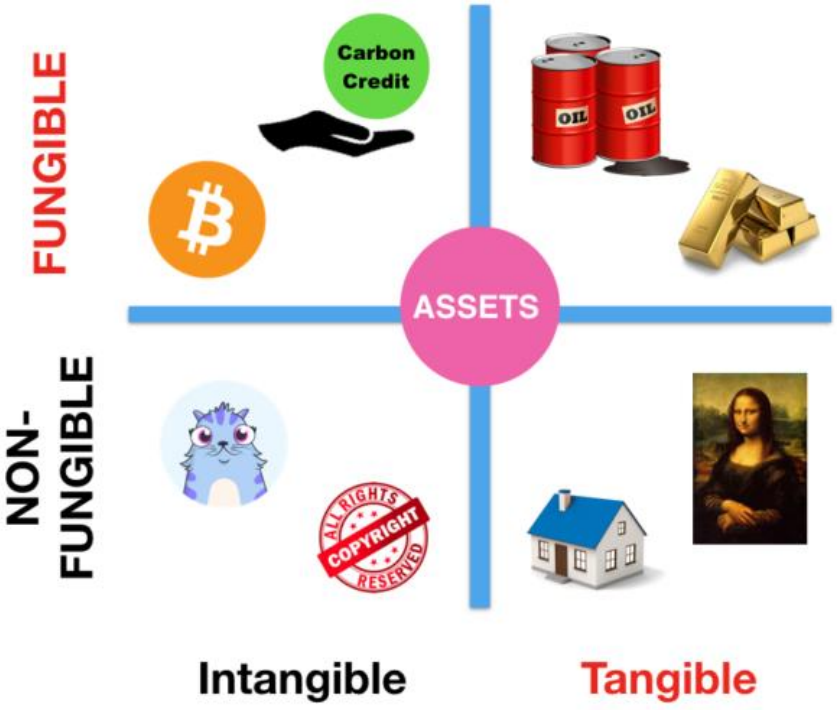
کوین ایران

کوین ایران، اولین پایگاه خبری فارسی زبان در حوزه فناوری بلاکچین، ارزهای رمزنگاری شده و پلتفرم های مرتبط با بلاکچین است. این وب سایت در سال ۱۳۹۲ توسط بابک جلیلوند و آرش محبوب، با هدف اطلاع رسانی، آموزش و مشاوره به جامعه فارسی زبان علاقه مند به رمز ارزها راه اندازی شد و اولین مقاله آن در ۱۴ دی ماه همان سال، با عنوان «بیت کوین چیست؟» منتشر گردید . هدف کوین ایران از معرفی این فناوری ایجاد محیطی پژوهشی و آموزشی در راستای استفاده صحیح از این فناوری جهت ارائه تسهیلات و رفاه به جوامع فارسی زبان است.

www.coiniran.com

مقدمه

توکن های ERC721 که عموما از آنها به عنوان توکن های غیر قابل تبدیل یا غیر قابل تعویض یاد می‌شود (Non-Fungible tokens) از زمانی که اولین بار در سپتامبر ۲۰۱۷ به عنوان یک پروتکل توسعه ای (EIP) روی اتریوم مطرح شد، توجه بسیاری از توسعه دهندگان را به خود معطوف داشته است. اصولا کالا ها یا دارایی های Non-Fungible آن دسته از دارایی یا کالا محسوب می‌شوند که قابل تهاتر با کالا یا دارایی دیگر و یا حتی دارایی مشابهی از همان نوع نیستند و این دارایی ها در واقع منحصر به فرد هستند.



این توکن ها آزادی عمل بسیار زیادی برای توسعه دهندگان فراهم می‌کند تا بتوانند مالکیت دنباله هایی از اطلاعات را روی بلاکچین اتریوم فراهم آورند. مشخصه بارز توکن های غیر قابل تبدیل این است که هر توکن به یک شناسه متفاوت مرتبط شده که آن را برای مالک توکن منحصر به فرد می‌نماید. این ویژگی یک تفاوت بنیادی با توکن های ERC-20 است که در آنجا هر توکن قابل تعویض است. در استاندارد ERC-20، توسعه دهندگان می‌توانند هر تعدادی از توکن ها را در یک اسمارت کانترکت ایجاد کنند اما در ERC721 هر توکن در قرار داد در بر دارنده مقدار متفاوتی است.

استاندارد ERC721

مانند استانداردهای قبلی، این استاندارد نیز شامل مجموعه ای از قوانین است که توکن های روی شبکه اتریوم با آنها سر و کار دارند. این استاندارد ها مشخصا ویژگی های زیر را برای یک توکن تصریح می‌کنند:

- **تصمیم گیری در خصوص مالکیت توکن**
- **نحوه ایجاد توکن**
- **نحوه ارسال توکن**
- **نحوه سوزاندن توکن**

ERC721 به خاطر گستره استفاده از آن در زیرساخت ها و اکوسیستم ها، اهمیت روز افزونی پیدا کرده است. یک نمونه عبارت است امکانی که این استاندارد به والت ها و اکسچنج ها می‌دهد که بتوانند اینترفیس هایی برای ایجاد این نوع توکن که ارزش بالایی هم پیدا خواهد کرد داشته باشند. در نهایت نیز منجر به گستردگی و سیالیت بیشتر اکوسیستم شده و ارزش دارایی ها در این اکوسیستم افزایش خواهد یافت چرا که هرکس می‌تواند دارایی غیر قابل تعویض خود را داشته باشد.

کاربرد ERC721 در بازی ها

توکن های غیر قابل تعویض امروزه کاربرد رو به گسترشی به عنوان دارایی دیجیتال در بازی های مبتنی بر بلاکچین یافته است. یکی از همین بازی ها که سرو صدای زیادی نیز به پا کرده بازی CryptoKitties است. در این بازی هرکس می‌تواند گربه دیجیتال خود را طوری طراحی کند که منحصر به فرد باشد. در اینجا این گربه دیجیتالی در واقع یک دارایی غیر قابل تعویض (Non-Fungible) است.

بازی کریپتو کیتیز منجر به نیروی محرکه ای در استاندارد ERC721 روی شبکه اتریوم گردید. دارندگان درایی دیجیتال در این بازی می‌توانند اطمینان داشته باشند که دارایی شان (گربه دیجیتال) قابل کپی شدن یا تصرف نیست.

این بازی باعث شد که بازی های دیگری با این مفهوم و در زمینه های متفاوت روی استاندارد ERC721 شکل بگیرد.

توابع ERC721

استاندارد ERC721 توابع و رخداد های زیر را تعریف می‌کند:

name, symbol, totalSupply, balanceOf, ownerOf, approve, takeOwnership, transfer, tokenOfOwnerByIndex, tokenMetadata

رخداد ها:

Transfer, Approval

پیش از پرداختن به توابع، لازم است دو مفهوم مالکیت توکن (Token Ownership) و ایجاد توکن (Token Creation) توضیح داده شود.

مالکیت توکن

موقعی که شما یک توکن ERC-20 را خریداری می‌کنید، مجوز مالکیتی شما در همان اسمارت کانترکت نوشته می‌شود. در این اسمارت کانترکت همچنین اطلاعات تعداد و نوع توکن های هر آدرس پس از هر توافقی ثبت می‌شود و تمام. نکته حائز اهمیت در اینجا این است که در این اسمارت کانترکت ها نگرانی در خصوص توکن های خاص و مشخص وجود ندارد زیرا همگی مانند هم و قابل تعویض (fungible) هستند.

اما در ERC721 اینطور نیست و ارزش هر توکن مانند دیگری نیست چرا که قابل تعویض نیست. بنابراین در اسمارت کانترکت مربوطه تنها اضافه کردن بالانس و آدرس کافی نیست و باید برای هر توکن، جزئیات مالکیتی نیز ثبت گردد.

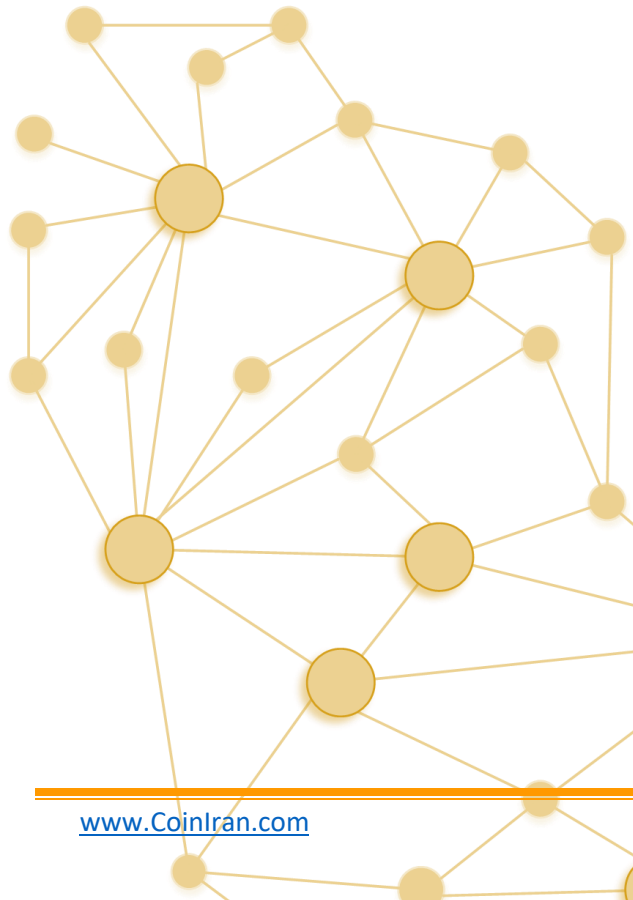
ایجاد توکن

ایجاد توکن ERC-20 درواقع ایجاد بالانس توکن ها است. تمام کاری که باید انجام دهید این است که یک حد بالا را ست کنید طوری که کسی نتواند بیشتر از آن، (در آن اسمارت کانترکت) توکن ایجاد کند. اما ایجاد توکن ERC721 پیچیدگی نسبتا زیادی دارد. استاندارد ERC721 آرایه ای از توکن ها را نگهداری می‌کند و هریک از این توکن ها باید به طور جداگانه به این آرایه اضافه شود.

کد زیر یک کد نمونه برای ایجاد قرارداد ECR721 است:

```
contract ERC721 {

// ERC20 compatible functions

function name() constant returns (string name);

function symbol() constant returns (string symbol);

function totalSupply() constant returns (uint256 totalSupply);

function balanceOf(address _owner) constant returns (uint balance);

// Functions that define ownership

function ownerOf(uint256 _tokenId) constant returns (address owner);

function approve(address _to, uint256 _tokenId);
```

```
function takeOwnership(uint256 _tokenId);

function transfer(address _to, uint256 _tokenId);

function tokenOfOwnerByIndex(address _owner, uint256 _index) constant returns (uint t
okenId);

// Token metadata

function tokenMetadata(uint256 _tokenId) constant returns (string infoUrl);

// Events

event Transfer(address indexed _from, address indexed _to, uint256 _tokenId);

event Approval(address indexed _owner, address indexed _approved, uint256 _tokenId);

}
```



حال نگاهی می اندازیم به کد توابع سازگار با ERC20 که در کد اسمارت کانترکت فوق آمده است:

```
name()
This function is used to define the name of the token to outside contracts and
applications. Let’s see how this works.
contract Blockgeeks {
 function name() constant returns (string name){
   return "Read Blockgeeks";
 }
}
symbol()
The symbol() function helps with token identification, by creating its shorthand
and symbol. The function also provides compatibility with the ERC20 token standard.
contract Blockgeeks {
 function symbol() constant returns (string symbol) {
   return "BG";
 }
}
totalSupply()
The totalSupply() function defines the total number of the tokens available in the
contract and it also returns the total number of coins available on the blockchain.
The supply does not have to be constant.
contract Blockgeeks {
 // This can be an arbitrary number
 uint256 private totalSupply = 1000000000;
 function totalSupply() constant returns (uint256 supply) {
   return totalSupply;
 }
}
balanceOf()
This function is used to find the number of tokens that a given address owns.
contract Blockgeeks {
 mapping(address => uint) private balances;
 function balanceOf(address _owner) constant returns (uint balance)
{
   return balances[_owner];
 }
{
```

توابع مالکیتی (Ownership)

دو حوزه ای که توکن های ERC721 کیفیت منحصر بفرد خود را نشان می‌دهند حوزه های نحوه مالیکت و ایجاد آنها هستند. این توابع عبارنتد از:

ownerOf()

این تابع آدرس مالک توکن رو بر می‌گرداند. از آنجایی که توکن های ERC721 غیر قابل تعویض و منحصر بفرد هستند، روی بلاکچین بصورت یک ID یونیک و منحصر بفرد نشانه گذاری می‌شود. هرکسی می‌تواند از طریق همین ID به آسانی مالک توکن را تشخیص دهد.

```
contract Blockgeeks {

 mapping(uint256 => address) private tokenOwners;

 mapping(uint256 => bool) private tokenExists;

 function ownerOf(uint256 _tokenId)

 constant returns (address owner) {

   require(tokenExists[_tokenId]);

   return tokenOwners[_tokenId];

 }

}
```

approve()

این تابع اجازه ارسال توکن از سوی مالک آن را توسط شخص دیگری تایید می‌کند. مثلا آلیس مالک یک توکن BG است، او با فراخوانی تابع فوق (درصورت موفقیت آمیز بودن) به باب اجازه کنترل کامل مالکیتی روی این توکن مشخص به جای آلیس را در زمان دیگری خواهد داد.

```
contract Blockgeeks {


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

 function approve(address _to, uint256 _tokenId){
   require(msg.sender == ownerOf(_tokenId));
   require(msg.sender != _to);
```

```
    allowed[msg.sender][_to] = _tokenId;
    Approval(msg.sender, _to, _tokenId);
  }
}
```

takeOwnership()

این تابع کاری شبیه برداشت (withdraw) انجام می‌دهد. یک شخص بیرونی با فراخوانی این تابع می‌تواند توکن ها را از اکانت یک کاربر خارج نماید. بنابراین اگر آلیس به باب اجازه دهد که مالکیت مقدار مشخصی از توکن ها را داشته باشد و بخواهد باب توکن ها را از موجودی اکانت یک کاربر برداشت نماید، این تابع را فراخوانی می‌کند.

```
contract Blockgeeks {

 function takeOwnership(uint256 _tokenId){
   require(tokenExists[_tokenId]);
   address oldOwner = ownerOf(_tokenId);
   address newOwner = msg.sender;
   require(newOwner != oldOwner);
   require(allowed[oldOwner][newOwner] == _tokenId);
   balances[oldOwner] -= 1;
   tokenOwners[_tokenId] = newOwner;
   balances[newOwner] += 1;
   Transfer(oldOwner, newOwner, _tokenId);
 }
}
```

transfer()

تابع فوق نیز روش دیگری برای انتقال توکن ها است. به مالک توکن اجازه می‌دهد که توکن را به کاربر دیگر مانند یک رمزارز ارسال نماید. نکته مهم این است که این تابع تنها موقعی می‌تواند استفاده شود که مالکیت اکانت گیرنده قبلا توسط اکانت فرستنده تایید شده باشد.

```
contract Blockgeeks {

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

 function removeFromTokenList(address owner, uint256 _tokenId) private {

   for(uint256 i = 0;ownerTokens[owner][i] != _tokenId;i++){

     ownerTokens[owner][i] = 0;

   }

 }

 function transfer(address _to, uint256 _tokenId){

   address currentOwner = msg.sender;

   address newOwner = _to;

   require(tokenExists[_tokenId]);

   require(currentOwner == ownerOf(_tokenId));

   require(currentOwner != newOwner);

   require(newOwner != address(0));

   removeFromTokenList(_tokenId);

   balances[oldOwner] -= 1;

   tokenOwners[_tokenId] = newOwner;
   balances[newOwner] += 1;

   Transfer(oldOwner, newOwner, _tokenId);

 }
}
```

tokenOfOwnerByIndex()

مالک هر توکن غیر قابل تعویض می‌تواند در هر زمان مالکیت بیش از یک توکن را داشته باشد. اما از آنجایی که گفته شد هر توکن دارای ID یونیکی است لذا برای مالک توکن واقعا مشکل خواهد بود که همه اینها را بتواند دنبال کند و وضعیت آنها را داشته باشد. برای ساده سازی این فرایند، اسمارت کانترکت مربوطه یک رکورد از ID ها را ثبت می‌کند که در آن رکورد اطلاعات توکن های یک کاربر نگهداری می‌شود. این عمل توسط بازیابی ایندکس یک لیست یا آرایه از توکن هایی که کاربر مالک آن است انجام می‌شود. با این تابع بازیابی یک توکن از آرایه مذکور امکان پذیر خواهد شد.

```
contract Blockgeeks {

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

 function tokenOfOwnerByIndex(address _owner, uint256 _index) constant returns (uint
tokenId){

   return ownerTokens[_owner][_index];

 }

}
```

متا دیتا

متا دیتا ها مجموعه ای از اطلاعات است که مشخصات و ویژگی های بیشتری را برای دیتای اصلی فراهم می‌کند. متا دیتا ها اطلاعات خیلی مهمی را ارائه می‌دهند مانند توضیحات و شرح دیتا، انتقال، مشاهده و مرور و ... که به این ترتیب نقش بسیار مهمی در مدیریت منابع دیجیتال ایفا می‌کنند. در استاندارد ERC721 نیز تابع tokenMetadata() کمک می‌کند تا بتوانیم متادیتای یک توکن را تعریف کنیم.

```
contract Blockgeeks {

 mapping(uint256 => string) tokenLinks;
 function tokenMetadata(uint256 _tokenId) constant returns (string infoUrl) {

   return tokenLinks[_tokenId];
 }
}
```

وقایع (Events)

وقتی یک اسمارت کانترکت این وقایع را فراخوانی می‌کند آنها فعال (روشن) می‌شوند و برای برنامه هایی (خارج از اسمارت کانترکت) که به این وقایع نیاز دارند منتشر می‌شوند. برنامه های مذکور به محض روشن شدن و دریافت این وقایع کدهایی را اجرا می‌کنند. در استاندارد ERC721 دو نوع از این وقایع تعریف شده است: Transfer() و Approval().

Transfer

به محض انتقال مالکیت یک توکن از یک اکانت به اکانت دیگر این ایونت فعال می‌شود. اطلاعاتی که این ایونت با خود می‌آورد شامل اکانت ارسال کننده، اکانت دریافت کننده، ID توکن مربوطه.

```
<div style="background: #ffffff; overflow:auto;width:auto;border:solid gray;border-wi
dth:.1em .1em .1em .8em;padding:.2em .6em;"><pre style="margin: 0; line-height: 125%"
>contract Blockgeeks {

 event Approval(address indexed _owner, address indexed _approved, uint256 _tokenId);

}
</pre></div>
```

Approval

این ایونت وقتی فعال می‌شود که یک کاربر مالکیت یک توکن مشخص را به کاربر دیگری واگذار نماید. این ایونت علاوه بر چک کردن ID توکن، حاوی اطلاعات اکانت دارنده و دریافت کننده مالکیت آن توکن است.

```
contract Blockgeeks {

 event Approval(address indexed _owner, address indexed _approved, uint256 _tokenId);

}
```

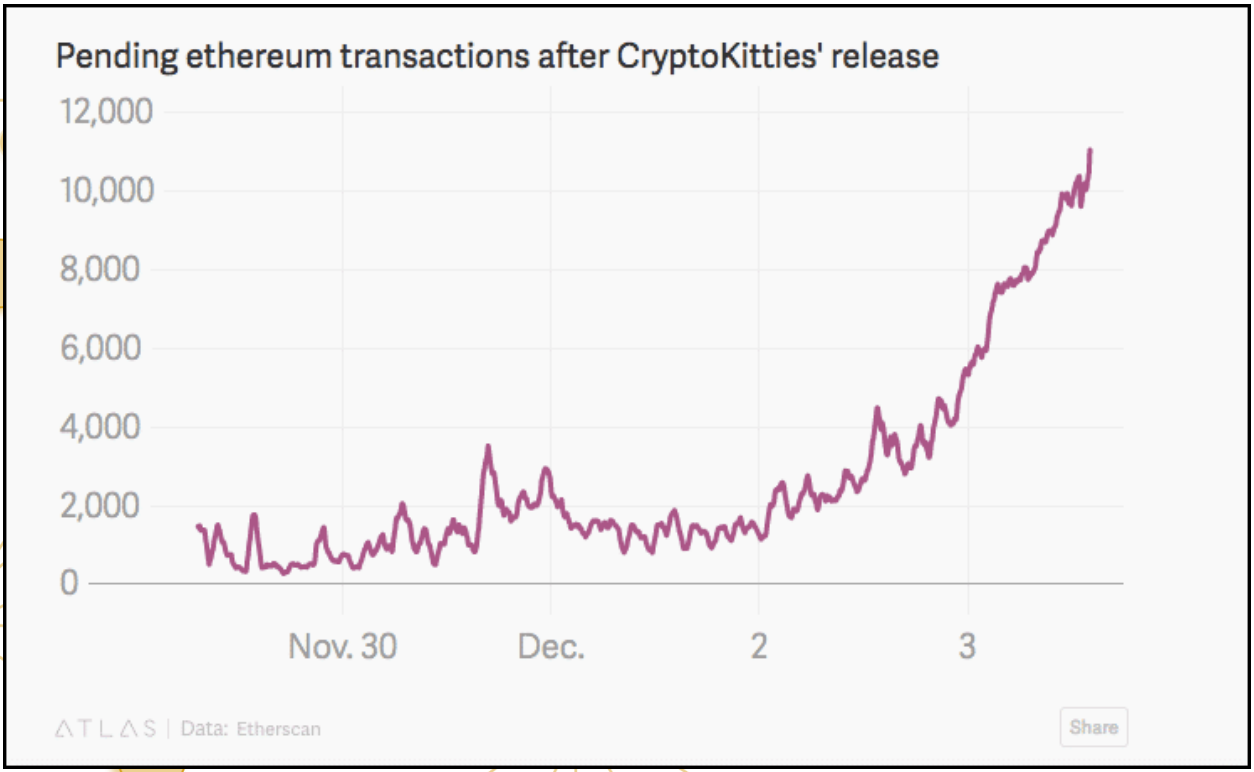
ERC721 و آینده

بازی کریپتوکیتی توانست به سرعت بین کاربران پخش شده و در سطح وسیعی منتشر شود (شبیه انتشار ویروس!) و به سومین اسمارت کانترکت روی شبکه اتریوم از نظر میزان مصرف gas تبدیل شد.

Top 10 ETH Contracts By Transaction Count Over Last 1,500 Blocks

Address	ID	Pct Total Tx Count
0x8d12a197cb00d4747a1fe03395095ce2a5cc6819	Etherdelta	4.65
0x3f5ce5fbfe3e9af3971dd833d26ba9b5c936f0be		3.37
0x06012c8cf97bead5deae237070f9587f8e7a266d	Cryptokitties	2.20
0x93e682107d1e9defb0b5ee701c71707a4b2e46bc		1.68
0x5baeac0a0417a05733884852aa068b706967e790		1.10
0xc8ffd394421e09cb48b620dda56168171ca35ab7		1.06
0x86fa049857e0209aa7d9e616f7eb3b3b78ecfdb0		1.02
0xd26114cd6ee289accf82350c8d8487fedb8a0c07		0.71
0x70faa28a6b8d6829a4b1e649d26ec9a2a39ba413	Shapeshift	0.69
0x876eabf441b2ee5b5b0554fd502a8e0600950cfa		0.61

اما نقطه ضعف این شهرت سریع و وسیع، خفه کردن شبکه اتریوم بود به طوری که سایر تراکنش ها و کارهای روی این شبکه با کندی بسیار زیاد مواجه شد.



سازندگان بازی برای کنترل این وضعیت و جلوگیری از انتشار بی حد و مرض این بازی ناچار شدند هزینه gas را بالا ببرند که برخلاف میل باطنی شان بود اما برای پایداری شبکه اتریوم این اقدام لازم بود.

از این اتفاق دو درس آموخته شد:

یکی این که شبکه اتریوم در زمان ارائه این بازی آمادگی اپلیکیشن های غیر متمرکز (Dapps) در مقیاس وسیع را نداشت و دوم این که تقاضای زیادی برای کاربردها و برنامه های کلکسیونی مبتنی بر رمزارز وجود دارد.

از این رو این استاندارد پتاسنیل بسیار زیادی برای پذیرش رمزارز ها در جامعه ایجاد می‌کند که کسب و کار ها می‌توانند به آن بپردازند. از جمله این کاربردها می‌توان موارد زیر را ذکر کرد:

- **املاک**
- **هنر**
- **وام و موارد مرتبط به امور مالی**
- **کشاورزی (پروژه Fieldcoin)**

منابع:

https://education.district0x.io/general-topics/understanding-ethereum/erc-721-tokens/

https://blockgeeks.com/guides/erc-721-cryptokitty-token/#Intro_to_ERC-721_The_CryptoKitty_Token

Images: 1, 2, 3, 4