Understanding Cryptocurrency Development

A basic token is just a table and four methods.

Alberto Cuesta Cañada

Follow

HackerNoon.com

--

Listen

Share

“The more that you read, the more things you will know. The more that you learn, the more places you’ll go.” — Dr. Seuss

A fact that most business leaders don’t know is that developing a cryptocurrency in Ethereum is surprisingly simple. Developers discover that as soon as they dive into coding smart contracts. For those without the confidence to code, cryptocurrencies remain arcane and complicated.

Part of my job as Chief Architect for TechHQ is to teach CEOs about blockchain technology in terms that they understand so that we can design the technology under their business processes. I often see them assuming that cryptocurrencies are too complicated for them to understand.

In this article, I’m going to explain what cryptocurrencies and tokens are with a business audience in mind. I’m going to show you the code as well, as most non-developers feel less intimidated once they see that the code for a cryptocurrency fits in a page.

A fact that most business leaders don’t know is that developing a cryptocurrency in Ethereum is surprisingly simple.

Please note that this article refers to cryptocurrencies and tokens in Ethereum and to an extent other distributed computers such as Hyperledger. A cryptocurrency that runs on its own dedicated blockchain, such as Bitcoin, is much more complex to develop.

Are you ready? Hold my hand and come into the rabbit hole…

Cryptocurrencies

Everyone hears first about the ERC20 term when researching cryptocurrencies. ERC is a label in theEthereum GitHub repository that means that someone wants formal feedback on something. In the early days of Ethereum someone wanted feedback for a cryptocurrency specification and got the ERC20 label. That is how history is often made, without much thought.

Please have a look at the ERC20 implementation from OpenZeppelin. That contract is the base for most of the cryptocurrencies out there. Don’t worry if you don’t understand too much of the code. Pay attention to the fact that it only has 228 lines of code, and of those two thirds are comments. It can’t be that complicated.

You could deploy theERC20 contract as-is and you would have a cryptocurrency. Most likely you want to add some functionality such as a token symbol and decimals, or transfer restrictions, or to stop transfers to invalid accounts. All these variations are only specifications to add more methods to an ERC20 contract. They are not that different between them.

You could deploy theERC20 contract as-is and you would have a cryptocurrency.

If you need to add functionality to a cryptocurrency you don’t need to trawl the internet for some ERC token that does what you want. Cryptocurrencies are software applications, and very simple ones at that. If you need to track who are the holders of your token, for example, you can just add a data structure to your cryptocurrency that stores that information and update it when needed. You can make stuff up. Go wild.

To recap, a cryptocurrency in Ethereum is:

- A simple software application
- that might be following some standards
- probably including ERC20
- and probably with some custom code.

In the next section, we are going to go into the code to break it down even further. We are going to get rid of even more complexity to get to the very simple core of what a token is.

The simplest cryptocurrency in town

The ERC20 implementation from OpenZeppelin has three contract variables and 13 methods. Contract variables are the data that is stored in the contract. The methods are what the contract can do with contract variables and data that is passed from outside.

Consider the contract as an application that runs in the cloud, where you can activate its methods but nothing else. The contract sits there waiting for someone to ask for something and then returns a value or updates its contract variables.

To code a very simple cryptocurrency I'm going to take some stuff fromERC20 and I'm going to discard the rest.

I'm going to keep in my contract a ledger for balances. In my cryptocurrency, each address relates to a certain value. In other words, each account holds some tokens.

I am only going to implement four methods to check balances, create tokens, destroy tokens and transfer tokens. With that functionality I have a cryptocurrency that I can use for payments. The whole contract is below:

A cryptocurrency in Ethereum can be as simple as a data structure with the balance of each account and a few functions to manage those balances.

Now that you know what a cryptocurrency is, you can start adding more functionality according to your business models. At least you should implement the ERC20 standard and use SafeMath, the contract above is for educational purposes.

Tokens

I like to use cryptocurrency

and token

as two different terms. I like to say that a token is something that you can have, and a cryptocurrency something that you can have and can use to pay for things.

One example of a token that is not a cryptocurrency is the contracts implemented with theERC721 standard. This is the one that you hear about when people talk about tokenization. The purpose of the ERC721 standard is for each one of its tokens to represent a physical asset in the blockchain.

In technical terms, it replaces the ledger of balances from ERC20 with a table that records who is the holder of each individual token:

With this token, instead of saying that Bob has 100 tokens, I would say that Bob has tokens #100, #42 and #1337, for example. The four methods described for ERC20 change a bit, but in essence that is the difference between an ERC20 cryptocurrency contract and an ERC721 token contract.

By now you have seen that cryptocurrencies and tokens are a table with a few methods to manipulate the data. The description for a token is very flexible and there are many things that you can describe as tokens.

When I design a solution I might fall into the habit of creating a lot of different tokens. It is easy to do since they are just tables linking users to something they have. I will use ERC standards when they are a good fit, but often I will code my token contracts from scratch.

The description for a token is very flexible and there are many things that you can describe as tokens.

A token for payments, another for voting rights, another for assets of some kind and so on. The marketing team always protests — a solution with four tokens is too complex to sell!. They are right. I am making their lives difficult.

The fix is really simple. If a software design has too many tokens, I remove the word token

from the description. I will have voting rights, asset keys, user handles, or whatever name I can come up with that doesn't use the word token and doesn't attract attention. Everyone is happy. A token is just a table and a few methods.

Conclusion

The technology that makes cryptocurrencies possible is complex. The code that creates a cryptocurrency with those tools is not. Business leaders in the blockchain space would do well in trying to understand the basics of cryptocurrencies and tokens. Breaking through the apparent complexity will help them to get their business ideas through to the development team.

In this article I’ve broken down the concept of cryptocurrencies and tokens to their bare minimum, to even less than the well-known ERC20 standard behind most blockchain applications. From that minimal starting point, I’ve shown cryptocurrencies and tokens to be very simple at the core, before business functionality is added.

Do you have an idea for a blockchain solution? Do you know your business but struggle to find a technologist that speaks your language. Please feel free to contact me, I love bringing business and technology together.