title: Smart contracts description: A non-technical introduction to smart contracts lang: en

Introduction to smart contracts {#introduction-to-smart-contracts}

Smart contracts are the fundamental building blocks of Ethereum's application layer. They are computer programs stored on the blockchain that follow "if this then that" logic, and are guaranteed to execute according to the rules defined by its code, which cannot be changed once created.

Nick Szabo coined the term "smart contract". In 1994, he wrote an introduction to the concept, and in 1996 he wrote an exploration of what smart contracts could do.

Szabo envisioned a digital marketplace where automatic, cryptographically-secure processes enable transactions and business functions to happen without trusted intermediaries. Smart contracts on Ethereum put this vision into practice.

Trust in conventional contracts {#trust-and-contracts}

One of the biggest problems with a traditional contract is the need for trusted individuals to follow through with the contract's outcomes.

Here is an example:

Alice and Bob are having a bicycle race. Let's say Alice bets Bob $10 that she will win the race. Bob is confident he'll be the winner and agrees to the bet. In the end, Alice finishes the race well ahead of Bob and is the clear winner. But Bob refuses to pay out on the bet, claiming Alice must have cheated.

This silly example illustrates the problem with any non-smart agreement. Even if the conditions of the agreement get met (i.e. you are the winner of the race), you must still trust another person to fulfill the agreement (i.e. payout on the bet).

A digital vending machine {#vending-machine}

A simple metaphor for a smart contract is a vending machine, which works somewhat similarly to a smart contract - specific inputs guarantee predetermined outputs.

- You select a product
- The vending machine displays the price
- You pay the price
- The vending machine verifies that you paid the right amount
- The vending machine gives you your item

The vending machine will only dispense your desired product after all requirements are met. If you don't select a product or insert enough money, the vending machine won't give out your product.

Automatic execution {#automation}

The main benefit of a smart contract is that it deterministically executes unambiguous code when certain conditions are met. There is no need to wait for a human to interpret or negotiate the result. This removes the need for trusted intermediaries.

For example, you could write a smart contract that holds funds in escrow for a child, allowing them to withdraw funds after a specific date. If they try to withdraw before that date, the smart contract won't execute. Or you could write a contract that automatically gives you a digital version of a car's title when you pay the dealer.

Predictable outcomes {#predictability}

Traditional contracts are ambiguous because they rely on humans to interpret and implement them. For example, two judges

might interpret a contract differently, which could lead to inconsistent decisions and unequal outcomes. Smart contracts remove this possibility. Instead, smart contracts execute precisely based on the conditions written within the contract's code. This precision means that given the same circumstances, the smart contract will produce the same result.

Public record {#public-record}

Smart contracts are useful for audits and tracking. Since Ethereum smart contracts are on a public blockchain, anyone can instantly track asset transfers and other related information. For example, you can check to see that someone sent money to your address.

Privacy protection {#privacy-protection}

Smart contracts also protect your privacy. Since Ethereum is a pseudonymous network (your transactions are tied publicly to a unique cryptographic address, not your identity), you can protect your privacy from observers.

Visible terms {#visible-terms}

Finally, like traditional contracts, you can check what's in a smart contract before you sign it (or otherwise interact with it). A smart contract's transparency guarantees that anyone can scrutinize it.

Smart contract use cases {#use-cases}

Smart contracts can do essentially anything that computer programs can do.

They can perform computations, create currency, store data, mint NFTs, send communications and even generate graphics. Here are some popular, real-world examples:

- Stablecoins
- Creating and distributing unique digital assets
- An automatic, open currency exchange
- Decentralized gaming
- An insurance policy that pays out automatically
- A standard that lets people create customized, interoperable currencies

More of a visual learner? {#visual-learner}

Watch Finematics explain smart contracts:

Further reading {#further-reading}

- How Smart Contracts Will Change the World
- Smart Contracts: The Blockchain Technology That Will Replace Lawyers
- Smart contracts for developers
- Learn to write smart-contracts
- Mastering Ethereum - What is a Smart Contract?