Ethereum: The Five Things

A simple 30-minute introduction (BSc, no math)

Prof. Dr. Jörg Osterrieder

BSc Blockchain, Crypto Economy & NFTs

Spring 2026

Today's question. Ethereum sounds complicated. What is it, in five things you can hold in your head? Each concept gets two analogies, one short example, and zero formulas.

The five things, in 30 minutes

The five concepts

1. **Chain.** What you write on.
2. **Account.** Who you are.
3. **Transaction.** What you do.
4. **Smart contract.** Code that lives there.
5. **Gas.** What it costs.

How we will teach

- **Two analogies per concept.** If the first one does not click, the second one will.
- **One concrete example** per concept. No formulas.
- **One pointer** on each Apply slide to the full deck for the next layer of detail.

Companion Colab notebook. Five small cells let you compute hash, address, signature, and gas yourself.

1. The chain (a shared notebook)

Analogy 1: a shared Google Doc. The whole world is looking at the same document. Anyone can read it. New lines can only be appended at the bottom; nobody can edit a past line.

Analogy 2: a village ledger book. Long ago, every village had a leather-bound ledger where the clerk recorded each transaction in chronological order. Everyone in town knew where the book lived. Once a line was written, the clerk drew a line under it.

In one sentence. Ethereum is one giant ledger book that thousands of computers around the world keep an identical copy of, and they only ever add to the bottom.

The chain stores everything: transfers, contracts, contract data. There is one chain. Different blockchains (Bitcoin, Solana) are different ledger books.

Why the chain cannot be quietly edited

Each new page in the ledger ends with a tiny "fingerprint" that summarises every page that came before. Change a single comma on page 5, and the fingerprint at the bottom of page 5 changes; the fingerprint at the bottom of page 6 was computed from the OLD page 5 and no longer matches; page 7 breaks for the same reason; and so on, all the way to today.

Concrete. If you tried to silently move 1 ETH out of someone's account by editing a year-old block, every honest computer in the network would notice within seconds, because the fingerprints from that day forward would all be wrong.

This is what makes the ledger trustworthy without a central authority. The chain is tamper-evident. We do not need a bank to police it.

Technical detail. The fingerprint is a 64-character hex value computed by a one-way function called keccak256. Same input always produces the same fingerprint; finding two different inputs that produce the same fingerprint is computationally infeasible.

See full deck S04 to S05 for the formal definition of hash function and the keccak256 fingerprint.

2. Account (a name plus a key)

Analogy 1: a bank account with a stamp. You have a name (the account number, public to anyone) and a personal stamp that you keep secret. The bank only accepts a withdrawal slip if it carries YOUR stamp; anyone else can read the slip but cannot fake the stamp.

Analogy 2: an email inbox plus a password. The inbox address is on your business card. The password is yours alone. Anyone can mail you; only you can send from your address.

In one sentence. An Ethereum account is an address (your name) plus a secret key (your stamp). The address is public. The key is yours alone, forever.

There are two flavours: personal accounts (a key in your wallet) and contract accounts (code we will meet on slide 9). Today, only the personal flavour matters.

Apply: a real Ethereum address

`0xd8dA6BF26964aF9D7eEd9e03E53415D37aA96045`

What you see. 40 hexadecimal characters after `0x`. That is the public name of one specific account on Ethereum (it happens to belong to vitalik.eth).

What you do not see. The secret key that controls this account. It is a 64-character secret number that the owner stores in a wallet (e.g. MetaMask). Lose the key and the account is locked forever; nobody, not even Ethereum's developers, can reset it.

Practical rule. Treat your private key like the only physical key to a house with no locksmith.

Technical detail. A wallet generates 256 random bits using the operating system's secure randomness source. The public address is the last 20 bytes of keccak256 of the corresponding public key, where the public key itself comes from secp256k1 elliptic curve multiplication of the private key.

See full deck S06 to S07 for how the address is mathematically derived from the key (one-way function).

3. Transaction (a signed cheque)

Analogy 1: a paper cheque. You write the recipient, the amount, the date, and you SIGN it. The bank only accepts the cheque because the signature matches your account.

Analogy 2: a filled-in form at the post office. "From: my address. To: their address. What I am sending: this thing. Sealed with: my stamp."

In one sentence. A transaction is a small message that says "I, the holder of this account, want to send something somewhere", sealed with the secret key so nobody can forge it.

Without a valid signature, the network refuses the transaction. Forgery is mathematically infeasible.

A transaction in three pieces

Three concrete examples.

- **Payment.** From: Alice. To: Bob. What: 1 ETH. Cost: about 1 dollar in fees.
- **Token transfer.** From: Alice. To: a token contract. What: "transfer 100 USDC to Bob".
- **Contract deploy.** From: Alice. To: nobody. What: the source code of a new contract. The chain assigns it a new address.

Each transaction is signed before sending and immutable once accepted into a block.

Technical detail. The signature is ECDSA on the secp256k1 curve, computed over a hash of the (From, To, What, Nonce, Gas) fields. Validators recover the sender's public key from the signature alone, so no separate sender claim is trusted.

See full deck S17 for the formal transaction structure and S22 for the dollar fee math.

4. Smart contract (a vending machine)

Analogy 1: a vending machine. Once installed, the machine accepts coins and dispenses cans. Nobody behind the curtain. The rules are visible from the outside (which buttons cost what); they cannot be changed once the machine is bolted to the wall.

Analogy 2: an automated rule. Like an Outlook auto-reply. When a message arrives, the rule fires. The rule is written once and runs forever; you cannot un-publish it.

In one sentence. A smart contract is a small program that lives on the chain at its own address; anyone can call it; the rules cannot be changed once deployed.

Tokens (USDC, an NFT collection) are smart contracts. "DeFi protocol" is just a fancier smart contract.

Apply: HelloWorld in three lines

```
contract HelloWorld {
    string public greeting = "Hello, Class";
}
```

What this says. Create a contract. It holds a piece of text called `greeting`. The word `public` means anyone can read it from anywhere in the world.

Once you deploy this (a one-time transaction that costs a few dollars), the chain remembers the contract forever. Anyone, including you 50 years from now, can ask the contract "what is your greeting?" and get back "Hello, Class". There is no server to maintain. There is no domain to renew.

This is the smallest possible contract. Real ones add functions, events, and storage; the building block is the same.

Technical detail. Deploying = sending a special transaction whose to-field is empty and whose data-field carries the compiled bytecode. The network assigns the new contract a fresh address computed from (sender, nonce).

See full deck S29 to S32 for ABI, function selector, and how ERC-20 tokens are built from this same shape.

5. Gas (the fee meter)

Analogy 1: a taxi fare meter. The trip costs distance times rate. The city sets the rate; the driver gets a tip on top. You see the meter ticking before you commit.

Analogy 2: postage by weight. A heavy package costs more than a postcard. Same idea: a fancy contract call costs more than a simple ETH transfer.

In one sentence. Every transaction pays a fee proportional to how much computation it asks the network to do. Bigger work, bigger fee.

Why? Computation is not free. Thousands of computers around the world re-run your transaction. The fee compensates them and prevents anyone from spamming the network with junk.

The fee is paid in ETH. Most of it is BURNED (deleted forever); a small tip goes to the validator who included your tx.

Apply: a coffee-priced transaction

Scenario. You send 1 ETH from your wallet to a friend on Ethereum mainnet.

- Computation: a fixed minimum, called 21,000 "gas units".
- Rate: about 21 "gwei" per gas unit (a typical mainnet rate). 1 gwei = 1 billionth of an ETH.
- Fee in ETH: 21,000 $\times$ 21 gwei = 441,000 gwei = 0.000441 ETH.
- Fee in dollars (at \$3,000 per ETH): about \$1.32.

Compare.

- A token transfer (USDC, NFT): about \$2 to \$5.
- Deploying a contract: about \$5 to \$15.
- A complex DeFi swap: about \$5 to \$30 in busy times.

On Layer-2 networks (Arbitrum, Base), all of these are 10x to 100x cheaper.

Technical detail. The protocol adjusts the rate every 12 seconds based on block fullness. Blocks more than half full push the rate up by up to 12.5 percent; less-full blocks push it down. Senders add a small tip on top to incentivise inclusion.

See full deck S22 to S23 for the EIP-1559 fee market mechanics. Cell 5 of the Colab notebook computes any scenario.

Recap, in five lines

1. **Chain.** A shared, append-only ledger book that thousands of computers all keep identical copies of.
2. **Account.** A public address plus a secret key; the key is yours alone forever.
3. **Transaction.** A signed message saying "send something somewhere"; the signature makes forgery infeasible.
4. **Smart contract.** A small program living on the chain at its own address; rules cannot be changed once deployed.
5. **Gas.** The fee for asking the network to do work; bigger work, bigger fee.

If you remember nothing else: the chain is a ledger book everyone keeps a copy of, and the math makes it impossible to silently edit.

Five concepts. Two analogies each. No formulas. Goal achieved.

What we did NOT cover today

We deliberately skipped.

- How the fingerprint (hash) is computed mathematically.
- How thousands of computers actually agree on the next block (consensus, validators, slashing).
- How contract storage really lives in a Merkle Patricia Trie.
- How fees are split between burn and tip (EIP-1559 mechanics).
- How Layer-2 rollups settle to mainnet.
- Account abstraction (ERC-4337), MEV, blob transactions.

Where to go next. The full *Ethereum: Concepts, Theory, Approach* lecture (90 min, 35 main + 5 advanced sidebars) covers every one of those.

Same author, same Colab notebook (extended to 8 cells). Same place: the course site.

Companion notebook and what to read next

Companion Colab notebook (compute every concept yourself).

`https://colab.research.google.com/github/Digital-AI-Finance/Cryptoeconomics-Blockchain/blob/master/lectures/ethereum_simple/notebooks/ethereum_simple_concepts.ipynb`

If you are curious to go deeper.

- **Full teaching deck:** Ethereum: Concepts, Theory, Approach (90 min, 35 + 5 slides).
- **Plain-English intro:** ethereum.org/en/learn/
- **Code-first intro:** solidity-by-example.org
- **Wallet hygiene:** support.metamask.io/managing-my-wallet/securing-your-wallet/