Notes on Blockchain & Web3

# About Me at Consensys

* While creating technical content at Consensys, I learned about the self-custodial crypto wallet, **MetaMask**. (It’s self-custodial because you’re the only one with access to your private keys.)
* I also became familiar with the **Hyperledger Besu Ethereum execution client**.
* And finally, I spent most of my time documenting **Infura**, a node provider service for DApps to interact with the Ethereum L1 blockchain and L2 (Polygon) blockchains, for both Mainnet (Ethereum L1 mainnet) and Testnets (Goerli).

# Blockchain

**Blockchain** is a distributed, immutable ledger (**database**) where transactions made using decentralized applications (DApps) are recorded in blocks (**ordered records**).

**Blockchain** is a **distributed database** that maintains a continuously growing list of **ordered records**, called **blocks**. These blocks are linked using cryptography (**Transaction ID Hash**). Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data.

The underlying technology, *blockchain*, is what’s called a *distributed ledger* — a database hosted by a network of computers (**nodes**) instead of a single server — that offers users an immutable and transparent way to store information.

## Web3

Web1 was read-only, Web2 is read-write, Web3 is read-write+own. Users do not just contribute data; they own their data. Web3 aims to give users more control and ownership over their digital interactions (smart contracts, etc.) and assets (tokens, cryptocurrencies, data)

Web3 uses a stack of technologies based on decentralized blockchains that enable new business and social models. Users own their data, identity, content, and algorithms and participate as “shareholders” by owning the protocol’s tokens or cryptocurrencies.

Web3 is a new iteration of the World Wide Web which incorporates concepts such as decentralization, blockchain technologies, and token-based economics.

The Web3 stack comprises a variety of technologies that work together to create a decentralized and user-centric web. Let’s explore the key components:

1. **Blockchain Platforms:**
   * **Ethereum:** Ethereum is the most prominent blockchain platform for Web3. It enables smart contracts, decentralized applications (DApps), and the creation of new tokens (ERC-20, ERC-721, etc.). Ethereum’s native cryptocurrency is Ether (ETH).
   * **Other Blockchains:** Besides Ethereum, there are other blockchain platforms like Binance Smart Chain (BSC), Solana, Polkadot, and Avalanche, each with its unique features.
2. **Smart Contracts:**
   * **Solidity:** Solidity is the primary programming language for writing smart contracts on Ethereum. It allows developers to define rules and logic for DApps.
   * **WebAssembly (Wasm):** Some blockchains use Wasm-based smart contracts for improved efficiency and security.
3. **Decentralized File Storage:**
   * **IPFS (InterPlanetary File System):** IPFS is a peer-to-peer (P2P) protocol for storing and sharing files in a decentralized manner. It replaces traditional centralized servers with a distributed network of nodes.
   * **Filecoin:** Filecoin leverages IPFS and incentivizes users to share their storage space by rewarding them with FIL tokens.
4. **Identity and Authentication:**
   * **DID (Decentralized Identifiers):** DID standards enable self-sovereign identity. Users control their identity without relying on centralized authorities.
   * **Verifiable Credentials:** These allow users to prove specific attributes (e.g., age, education) without revealing unnecessary personal information.
5. **Cryptocurrencies and Tokens:**
   * **ERC-20 Tokens:** These fungible tokens follow the Ethereum standard and are widely used for ICOs and DeFi projects.
   * **ERC-721 Tokens (NFTs):** Non-fungible tokens represent unique assets (e.g., digital art, collectibles) and are indivisible.
   * **Stablecoins:** Stablecoins are pegged to real-world assets (e.g., USD, gold) provide stability for transactions.
6. **Web3 Libraries and Frameworks:**
   * **Web3.js:** A JavaScript library for interacting with Ethereum and other Web3-compatible blockchains.
   * **ethers.js:** Another JavaScript library for Ethereum development.
   * **Truffle:** A development framework for Ethereum DApps.
   * **Hardhat:** A popular Ethereum development environment.
7. **Consensus Mechanisms:**
   * **Proof of Work (PoW):** Used by Ethereum and Bitcoin, PoW requires miners to solve complex mathematical puzzles (nonce) to validate transactions.
   * **Proof of Stake (PoS):** Ethereum is transitioning to PoS with Ethereum 2.0. Validators are chosen based on the amount of cryptocurrency they hold.
   * **Proof of Authority (PoA)** — A reputation-based consensus algorithm that provides a practicable and efficient solution for blockchain networks, particularly **private networks**.

The PoA consensus algorithm leverages the value of identities, so block validators do not stake currencies but rather their own reputation.

Proof of Authority is a **highly scalable** system because it relies on a small number of block validators. The system is moderated by pre-approved participants who **verify blocks and transactions**.

1. **Infrastructure Providers:**
   * **Infura:** A service that provides **Ethereum nodes** for DApps to interact with a network, such as Arbitrum.
   * **Alchemy:** Similar to Infura, Alchemy offers Ethereum API services.
2. **User Experience (UX) Layers:**
   * **Wallets:** MetaMask, Trust Wallet, and other wallets allow users to manage their crypto assets and interact with DApps.

MetaMask is self-custodial because you’re the only one with access to your private keys.

* + **Browser Extensions:** Extensions like MetaMask integrate with browsers to facilitate seamless DApp interactions.
  + **Mobile Apps:** Mobile wallets and DApps enhance the user experience.

1. **Layer 2 (L2) Solutions:**

A Layer-2 blockchain is designed to improve a Layer-1 blockchain's scalability by taking some of the heavy lifting from the main (Layer-1) chain to increase throughput and lower [transaction fees](https://www.moonpay.com/learn/defi/what-are-ethereum-gas-fees).

This is achieved by allowing transactions to take place off-chain and only recording the final result on-chain. This way, the load on the main chain is significantly reduced, resulting in less throughput and improved performance.

**The goal is to make the main chain be used for fewer transactions with lower fees, which increases transaction speed.**

* + **Rollups:** Optimistic rollups and zk-rollups improve scalability by processing transactions off-chain and submitting cryptographic proofs to the main chain.

Optimistic Rollups: Enhance blockchain scalability by batch processing transactions off-chain and only reverting to the main chain when disputes arise, which helps keep costs down while maintaining a high level of security.

Optimistic Rollups: The sidechain assumes that transactions are valid by default. It processes them off-chain and only submits a summary (or “rollup”) to the main chain. If any invalid transactions are detected, they can be challenged and proven incorrect.

Optimistic Rollups: These are a type of scaling solution that allows Ethereum smart contracts to scale by passing messages between smart contracts on the Ethereum main chain and those on the Arbitrum second layer chain1. Essentially, it bundles multiple transactions into a single transaction, reducing the load on the Ethereum network.

Rollups bundle multiple Ethereum transactions off-chain and generate cryptographic proofs (e.g., SNARKs) before submitting them to the main chain.

* + **Sidechains**: These are separate blockchains that interact with the Ethereum Mainnet. They handle transactions off-chain and periodically settle on the Mainnet. Examples include **Polygon (previously Matic)** and **xDai**.
  + **State Channels:** These allow off-chain interactions between users while minimizing on-chain transactions.

A **Layer 2 State Channel** is a blockchain second layer solution that allows a group of participants to perform an unlimited number of private transactions off-chain. Unlike conventional on-chain transactions, state channel transactions are not made public and are only visible to participants on the channel

Remember that Web3 is an evolving ecosystem, and new technologies and standards continue to emerge. Developers and users alike contribute to shaping the decentralized web! 🌐🚀

## EIP and ERC

An **EIP** (**E**thereum **I**mprovement **P**roposal) is the primary mechanisms for proposing new features for Ethereum, collecting technical input on an issue from the community, and documenting design decisions in Ethereum.

EIPs are sorted into categories:

* Core
* Interface
* Networking
* Meta
* Informational
* **ERC** (**E**thereum **R**equest for **C**omment)

EIPs improve the Ethereum protocol itself, while ERCs enable applications and contracts to interact based on development standards and protocol rules.

**EIP** (**E**thereum **I**mprovement **P**roposal) and **ERC** (**E**thereum **R**equest for **C**omment) are both crucial for the growth of the **Ethereum ecosystem**. EIP is the initial proposal for a technical change in the Ethereum network. The EIP turns into an ERC upon approval through the **on-chain governance** model.

EIP-4337 is the proposal behind **ERC-4337**. EIP-4337 later turned into ERC-4337 when it got approved through Ethereum’s **on-chain governance** model in March 2023.

**EIP-4337** (also known as the **ERC-4337 standard**), is an Ethereum Improvement Proposal that introduced **Account Abstraction** to the Ethereum network ~~in the form of~~ **~~Gas Station~~**. Account Abstraction allows the **Paymaster smart contract** (ERC-4337 compliant) to act as a wallet and execute transactions on behalf of users. Transactions are gas-free. The sponsored gas fees are charged to the credit card on file, eliminating the need for end-users to hold native tokens in their wallets.

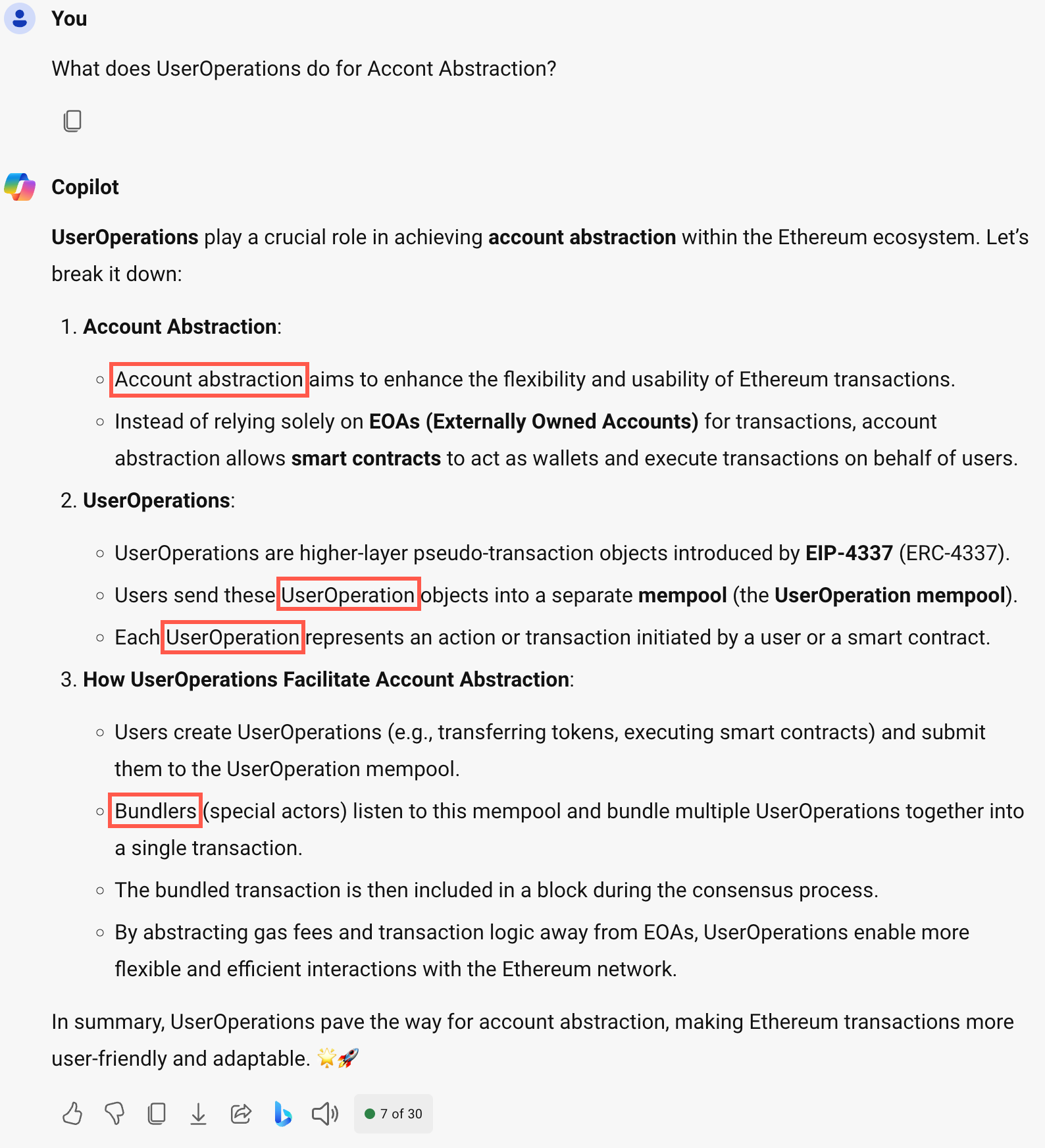
**ERC-4337** provides a solution for **Account Abstraction** by leveraging **UserOperations** and the **Alt Mempool**. It’s a departure from consensus-layer protocol changes, allowing users to wield smart contract wallets with ease.

**Account Abstraction** – Instead of relying solely on **EOAs (Externally Owned Accounts)** for transactions, account abstraction allows **smart contracts** (**Paymaster smart contract** [ERC-4337 compliant]) to act as **wallets** and execute transactions on behalf of users.

**Alt Mempool** – The mempool (short for **alternative memory pool**) is a term used in Ethereum blockchain that refers to a set of **unconfirmed transactions** that are waiting to be included in the next block on the blockchain. Note: I think this is also referred to as the **UserOperation mempool**.

**Bundlers**: Collect and bundle multiple transactions (**UserOperations**) together into a single transaction. The bundled transaction is then included in a block during the consensus process.

**UserOperations** – Introduced by **EIP-4337** (ERC-4337), a UserOperation represents an action or transaction initiated by a user or a smart contract.



**ERC20** (**E**thereum **R**equest for **C**omments 20) is a standard for **fungible tokens** on the Ethereum blockchain. **Fungible tokens** are tokens that are interchangeable with each other, like cryptocurrencies such as Ethereum or Bitcoin.

A screenshot of a screenshot of a document

Description automatically generated

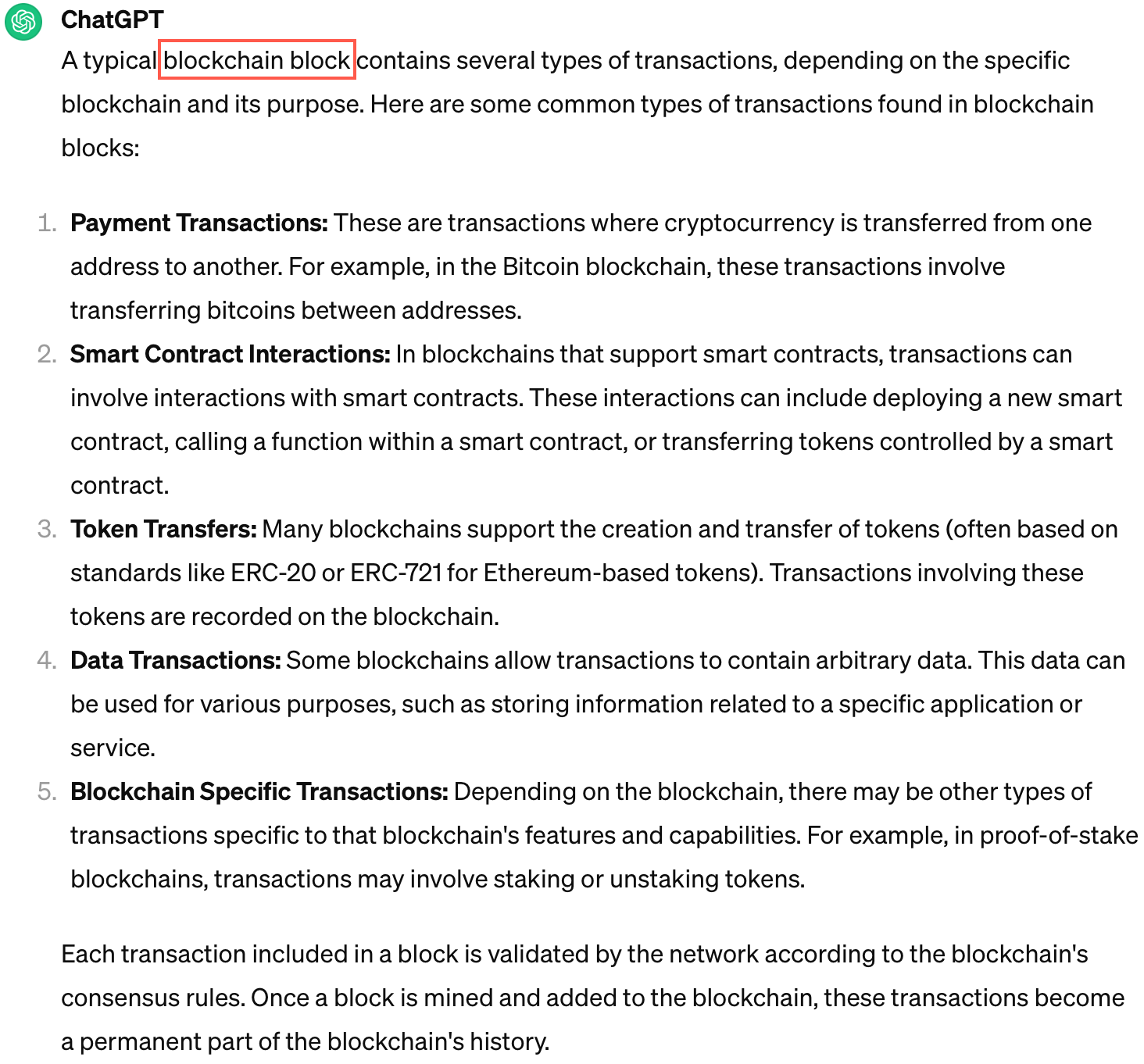
## Block

The transactions made by **dApps** during a given period are recorded into **a file called a block**, which is the basis of the blockchain network. A block stores information.

There are many pieces of information included within a block, but it doesn't occupy a large amount of storage space. A block contains:

* Cryptographic hash
* Cryptographic hash of the previous block
* Timestamp
* Transaction data

While the block itself does not "contain" the dApp in the traditional sense, it does contain the transactions and smart contract interactions that make up the dApp's functionality.



A diagram of a blockchain

Description automatically generated

A diagram of a block chain diagram

Description automatically generated

## Layer 1 (L1) Blockchain

A **baseblockchain** like Bitcoin or Ethereum. It is the first level of the ecosystem and corresponds to the main chain of the network. Layer 2 solutions and **side blockchains** can be built on top of this basis that layer 1 provides.

**Zero-knowledge (ZK) blockchains** — Grant developers a way to leverage the security of an existing **layer-1 blockchain** like **Ethereum** while enabling dApps to scale through higher throughput and faster transactions, protecting users’ personal information by keeping it hidden off-chain, and lowering the cost for end-users by **publishing transactions in batches**. Ultimately these advantages enable projects to build advanced dApps that rival the performance and functionality of Web2 systems while maintaining the benefits of decentralization.

While **zero-knowledge blockchains** do not fit neatly into the Layer 2 categorization, they can be complementary to Layer 1 and Layer 2 solutions by providing enhanced privacy features for transactions and data on the blockchain.

## Layer 2 (L2) Blockchain

Layer 2 solutions are protocols or frameworks that are built on top of the base layer (Layer 1) of a blockchain network, such as Ethereum. They aim to improve scalability, reduce transaction costs, or add additional features without compromising the security and decentralization of the base layer. Examples of Layer 2 solutions include: **state channels**, **sidechains**, and **rollups**.

An **L2 (Layer 2) blockchain** can serve as a **testnet**.

Testnets are primarily for testing, while L2 chains are for scaling and improving transaction efficiency.

Testnets are used during the development phase, whereas L2 chains are used in production to handle real transactions.

### Sidechain

Separate blockchain network that connects to another blockchain – called a parent blockchain or **Mainnet** (**Main Chain**)– via a two-way peg. These secondary blockchains have their own **consensus protocols** allowing a blockchain network to improve its privacy and security, and minimize the additional trust required to maintain a network.

A **side chain** is a type of **Layer 2 (L2) blockchain**. Layer 2 solutions are designed to help scale blockchain networks by handling transactions off the main chain (Layer 1), reducing congestion and costs. Side chains are one approach to achieving this scalability. They are separate blockchains that run in parallel to the main blockchain but can interact with it through various mechanisms.

**Side chains** typically have their own consensus mechanisms and rules but can connect to the main chain to transfer assets or data. This allows for faster transaction speeds and lower fees, as transactions can be processed on the side chain without requiring validation by the main chain. However, security and decentralization considerations must be carefully managed when using side chains.

### Rollup

Layer 2 (L2) blockchain that processes transactions away from the main blockchain to reduce transaction costs and increase throughput on the main chain. Polygon is an Ethereum Layer 2 rollup.

## Hash

In blockchain, a **Hash (Transaction ID Hash)** is a unique identifier for each block (unique digital fingerprint for data), generated by applying a cryptographic hash function to the block's data. (**SHA-256** is the *hashing algorithm* for Bitcoin.) A good hash function should be **deterministic**, meaning that the same input will always produce the same output.

## Consensus mechanisms/algorithms

Proof of Work (PoW) and Proof of Stake (PoS) are two common consensus algorithms used in blockchain networks to **validate transactions** and **create new blocks**.

A screenshot of a computer

Description automatically generated

### Proof of Work (PoW) consensus

This consensus mechanism requires network participants (often referred to as **miners**) to expend significant computational effort to solve an encrypted hexadecimal number (hash).

**Mining** involves repeatedly guessing nonces until a valid hash is found. This process requires substantial computational power.

The first miner to solve the hash gets to add a new block to the blockchain and receives a reward (block reward plus transaction fees).

**Nonce** — Short for “number used once,” is a special number that is **assigned to a block** in a blockchain during the **mining process**. It’s part of the proof-of-work (**PoW**) **consensus mechanism**. The nonce, serving as a **cryptographic puzzle**, is a variable that miners manipulate to produce a hash value that satisfies particular requirements.

### Proof of Stake (PoS) consensus

This consensus mechanism is used to verify new transactions and create new blocks in a blockchain. Unlike the traditional **Proof of Work (PoW)**, where miners compete to solve complex mathematical puzzles, PoS relies on the amount of stake (or value) held by participants in the system to determine consensus.

### Proof of Authority (PoA) consensus

A reputation-based consensus algorithm that provides a practicable and efficient solution for blockchain networks, particularly **private networks**. The PoA consensus algorithm leverages the value of identities, so block validators do not stake currencies but rather their own **reputation**. Proof of Authority is a **highly scalable** system because it relies on a small number of block validators. The system is moderated by pre-approved participants who **verify blocks and transactions**.

A **public blockchain** is an open network that allows **anyone** to participate in the network. It is **permissionless**, meaning anyone can join, read, write, or participate within the blockchain.

A **private blockchain** is managed by a network administrator. Participants need **consent** to join the network. It is **permissioned**, meaning only verified participants can access it.

## Tokens

A screenshot of a document

Description automatically generated

A screenshot of a text

Description automatically generated

## Cryptocurrency

Digital payment system that doesn't rely on banks to verify transactions. It’s a **peer-to-peer (P2P)** system that can enable anyone anywhere to send and receive payments. Instead of being physical money carried around and exchanged in the real world, cryptocurrency payments exist purely as digital entries to an online database describing specific transactions. When you transfer cryptocurrency funds, the transactions are recorded in a **public ledger**. Cryptocurrency is stored in **digital wallets**.

**Cryptocurrencies** run on a *distributed public ledger* called *blockchain*, a record of all transactions updated and held by currency holders.

## Smart contract

Digital agreement signed and stored on a *blockchain network* that executes automatically when the contract’s terms and conditions (T&C) are met. The T&C is written in blockchain-specific programming languages like *Solidity*.

Smart contracts can be viewed as *blockchain applications* that enable all parties to carry out their part of a transaction. Apps powered by smart contracts are frequently referred to as **decentralized applications** or **DApps**.

**Smart contracts** — Let’s say someone comes up with a program (**on Ethereum, programs are called *smart contracts***) that allows you to take out a loan. In *Decentralized Finance (DeFi)*, this would be called a “lending protocol”. These DeFi protocols work predictably, as programmed, no matter who’s accessing them, with total transparency into their usage through the blockchain record.

Smart contracts are a fundamental building block for decentralized applications (dApps). They run on blockchain networks and enable automated, trustless interactions. Here’s a concise overview:

1. **What Are Smart Contracts?**
   * Smart contracts are code written into a blockchain that executes the terms of an agreement or contract from outside the chain.
   * Smart contracts are self-executing agreements with predefined rules.
   * They’re written in languages like **Solidity** (for Ethereum) or **Rust** (for Polkadot).

**Note:** Both Vyper and Serpent are used for writing smart contracts on the Ethereum blockchain, although they are less common than Solidity.

* + Once deployed, they execute automatically when specific conditions are met.
  + Smart contracts run on the Ethereum Virtual Machine (EVM); a distributed computing network made up of all the devices running Ethereum nodes.

1. **How They Work:**
   * Users interact with smart contracts via transactions (e.g., sending tokens).
   * The contract’s code defines the logic (e.g., token transfers, voting, or NFT ownership).
   * Miners/validators validate and execute the contract on the blockchain.
2. **Use Cases:**
   * **DeFi (Decentralized Finance):** Lending, yield farming, decentralized exchanges (DEX), etc.
   * **NFTs (Non-Fungible Tokens):** Representing ownership of unique digital assets (one of a kind).
   * **Supply Chain:** Tracking goods, verifying authenticity, and ensuring transparency.
   * **Gaming:** In-game assets, rewards, and provably fair gameplay.
3. **Examples:**

* Tracking property details in real estate
* Protecting sensitive medical data in healthcare
* Improving the convenience and integrity of elections
* Reducing claims fraud in the insurance industry

Remember, each blockchain (Ethereum, Binance Smart Chain, etc.) has its own smart contract ecosystem.

Both **Vyper** and **Serpent** are used for writing smart contracts on the Ethereum blockchain, although they are less common than Solidity.

* **Vyper**: This is a newer language designed to be more secure and readable than Solidity. It has a simpler syntax and aims to minimize the risk of vulnerabilities in smart contracts.
* **Serpent**: This was one of the earlier languages for Ethereum smart contracts but is now largely deprecated. It has been replaced by more secure and efficient languages like Solidity and Vyper.

## dApps

A collection of interconnected **smart contracts**—which are automatically executable bits of code on a blockchain network. Behind the scenes, each *smart contract* performs a specific function within the application. Think of smart contracts as programmable Lego blocks: By stacking and compiling smart contracts, developers can create entire decentralized apps.

Instead of storing an app on centralized servers, DApps are stored across the many distributed computers—or “nodes”—that make up a blockchain network.

**Decentralized applications (dApps)** are often composed of **smart contracts**, among other components. **Smart contracts** are self-executing contracts with the terms of the agreement between buyer and seller being directly written into lines of code. They run on the blockchain, ensuring that the contract's execution is transparent, tamper-proof, and automated. In the context of DApps, smart contracts often govern the rules and logic of the application, such as handling transactions, managing tokens, and enforcing various business rules.

Unlike traditional applications that are hosted on centralized servers, dApps operate on a **peer-to-peer (P2P)** network of computers (**nodes**) that collectively maintain the application's data and code. Apps powered by smart contracts are frequently referred to as **decentralized applications or dApps**.

### Crypto wallet

* Your gateway to Web3
* Cryptocurrency is stored in digital wallets
* A digital identity management system
* Contains a **private key** that identifies and assesses the assets that are yours
* Once you’ve created your wallet, your **public wallet address** (often referred to as your **public key**), identifies all the transactions associated with you.
* A *user interface* that stores and manages your digital keys
* A Web3 *permissions manager* that allows you to grant access and approvals to **decentralized applications** (**dApps**) that you interact with
* Enables you to store the information that identifies you in one place
* When you create your wallet, you are given 12 words known as your **Secret Recovery Phrase** (**SRP**). The SRP is typically 12 random words.
* Crypto wallets like MetaMask, a crypto wallet is called **self-custodial** because you are the custodian and the only one who can access your private keys. MetaMask is a *self-custodial wallet*.

**Public wallet address/public key** — Once you’ve created your wallet, your public address (often referred to as your public key), identifies all the transactions associated with you. A public wallet address is a string of letters and numbers that represents your account, similar to a bank account number. For example, 0xd8da6bf26964af9d7eed9e03

### DAO

A **DAO (decentralized autonomous organization)** is a community-led entity that uses Ethereum *smart contracts* to establish the foundational rules and execute the agreed upon decisions. DAOs use smart contracts to manage many of the same processes and responsibilities found in corporations or nonprofits. Members of DAOs use tokens to vote on the rules governing the protocols and systems for which their DAO is responsible.

These *smart contracts* lay out the foundational framework by which the DAO is to operate. They are highly visible, verifiable, and publicly auditable so any potential member can fully understand how the protocol is to function at every step.

But an undeniable draw of Web3 is the way in which leaderless online groups of like-minded people can quickly gather, collectively pool capital, and make decisions.

# API

**API** (Application Programming Interface) — Set of definitions and protocols for building and integrating application software. It’s sometimes referred to as **a contract between** an information provider and an information user—establishing the content required from the consumer (**the call**) and the content required by the producer (**the response**). For example, the API design for a weather service could specify that the user supply a zip code and that the producer reply with a 2-part answer, the first being the high temperature, and the second being the low.

**API / Method** — In the context of APIs (Application Programming Interfaces), a **method** refers to a specific function or operation that can be called or executed *as part of an API*. An API is a broader concept that encompasses a collection of (1) **methods**, (2) **data structures**, and (3) **protocols** that allow different software applications to communicate with each other.

In summary, a **method** is not an API, but rather a part of an API that defines a specific action or function that can be performed.

**API endpoint** — Where an API receives requests about a specific resource on its server. For most services, these endpoints are URLs (Uniform Resource Locator), just like the ones you use to navigate to a website.

**REST API** (also called a “RESTful” API) — Representational State Transfer (REST) APIs use HTTP requests to perform **CRUD** (**C**reate, **R**ead, **U**pdate, **D**elete) operations on resources. They are widely used for **building web services**. REST is an architectural style for designing networked applications. RESTful APIs use standard **HTTP methods** (GET, POST, PUT, DELETE) to perform actions.

*Note: For the Ethereum Mainnet node, the consensus client uses a* ***REST API*** *and the execution client uses a JSON-RPC API.*

# JSON-RPC

JSON-RPC (**J**ava**S**cript **O**bject **N**otation - **R**emote **P**rocedure Call) – A protocol used for communication between a client and a server. In the context of blockchain, JSON-RPC is often used to interact with **blockchain nodes** or **clients**.

JSON-RPC (JavaScript Object Notation - Remote Procedure Call) – A lightweight, stateless protocol used for remote procedure calls (RPC). It uses JSON to encode the data, making it easy to read and write for both humans and machines.

A stateless protocol is a type of communication protocol in which each request from a client to a server is treated as an independent transaction. This means the server does not retain any session information or status about the client between requests.

# Abstracting gas costs

Gas abstraction in blockchain refers to a mechanism or technique that allows users to interact with a blockchain without needing to directly pay gas fees in the native cryptocurrency of that blockchain (such as Ether on Ethereum). Instead, the gas fees are abstracted away or paid in alternative ways, making it easier and more flexible for users to interact with decentralized applications (dApps) and smart contracts.

Abstracting gas costs in blockchain refers to the concept of hiding or simplifying the complexity of gas fees for the end user. Gas fees are the costs paid by users to execute transactions or run smart contracts on a blockchain, like Ethereum. These fees are typically paid in the blockchain's native cryptocurrency (e.g., Ether on Ethereum), and the amount can fluctuate based on network demand.

When gas costs are abstracted, the user doesn't need to worry about the details of gas prices, calculating gas limits, or even sometimes paying in the native cryptocurrency directly. This can be achieved in several ways:

* Relayers and Meta-Transactions: In some cases, a third party, known as a relayer, can pay the gas fees on behalf of the user and later get reimbursed, possibly through other means like tokens or off-chain payments. This allows the user to perform transactions without needing to hold the native cryptocurrency.
* Gasless Transactions: Some platforms or applications might allow "gasless" transactions, where the platform itself covers the gas costs, or the costs are included as part of the transaction fee in another form, such as a service fee.
* Simplified User Interfaces: Abstraction can also be done at the user interface level, where the complexity of gas fees is hidden, and users are provided with simple choices, such as "fast," "normal," or "slow" transaction speeds, without needing to understand the underlying gas mechanics.
* **Layer 2 Solutions**: Layer 2 scaling solutions often abstract gas fees by allowing transactions to be bundled or processed off-chain, reducing the direct interaction with gas fees on the main chain.

The goal of gas cost abstraction is to enhance user experience by reducing the friction and complexity associated with interacting with blockchain networks.

# Consensys

The following are the Consensys products that I worked on.

### MetaMask

* A crypto, *self-custodial wallet* where you’re in full control and grant permissions to **decentralized applications** (**dApps**) every time you connect.
* MetaMask is the gateway to a whole world of dApp interactions.
* Enables you to control how your assets and identities interact with an application, all in one place.
* **Secret Recovery Phrase (SRP)**:
  + A *MetaMask* term for **seed phrase**.
  + Your Secret Recovery Phrase (SRP) will always generate the same set of accounts derived from it, making the crypto wallet associated with SRP (seed phrase) **deterministic**.
  + The SRP generates your *public account number* that is similar to a bank account number, it's how you’re associated publicly on the blockchain.

### Infura

Interacting with a blockchain like Ethereum requires you to have direct access to an **Ethereum node**. Running your own node is a great way to contribute to the decentralization of a network. But for developers who want to simply focus on their creativity and deploy cool stuff to a blockchain, the maintenance it requires to run an **Ethereum node** could be a roadblock for some. This is where Infura could help speed things up.

* **Infura is a node provider.** They offer direct access for developers to **L1 blockchains like Ethereum**, and **side chains** including *Polygon*.
* Infura supplies you with an **RPC (remote procedure call) endpoint** that you can store inside your *smart contract* framework. These endpoints are what enable you to interact with a blockchain.
* Infura offers a quick and easy way to deploy your *smart contracts* on the blockchain of your choosing without having to run your own node.

### Ethereum Virtual Machine (EVM)

The computation engine for Ethereum that manages the state of the blockchain and enables *smart contract* functionality. The EVM is contained within the client software (e.g., Geth, Nethermind, and more) that you need in order to run a node on Ethereum.

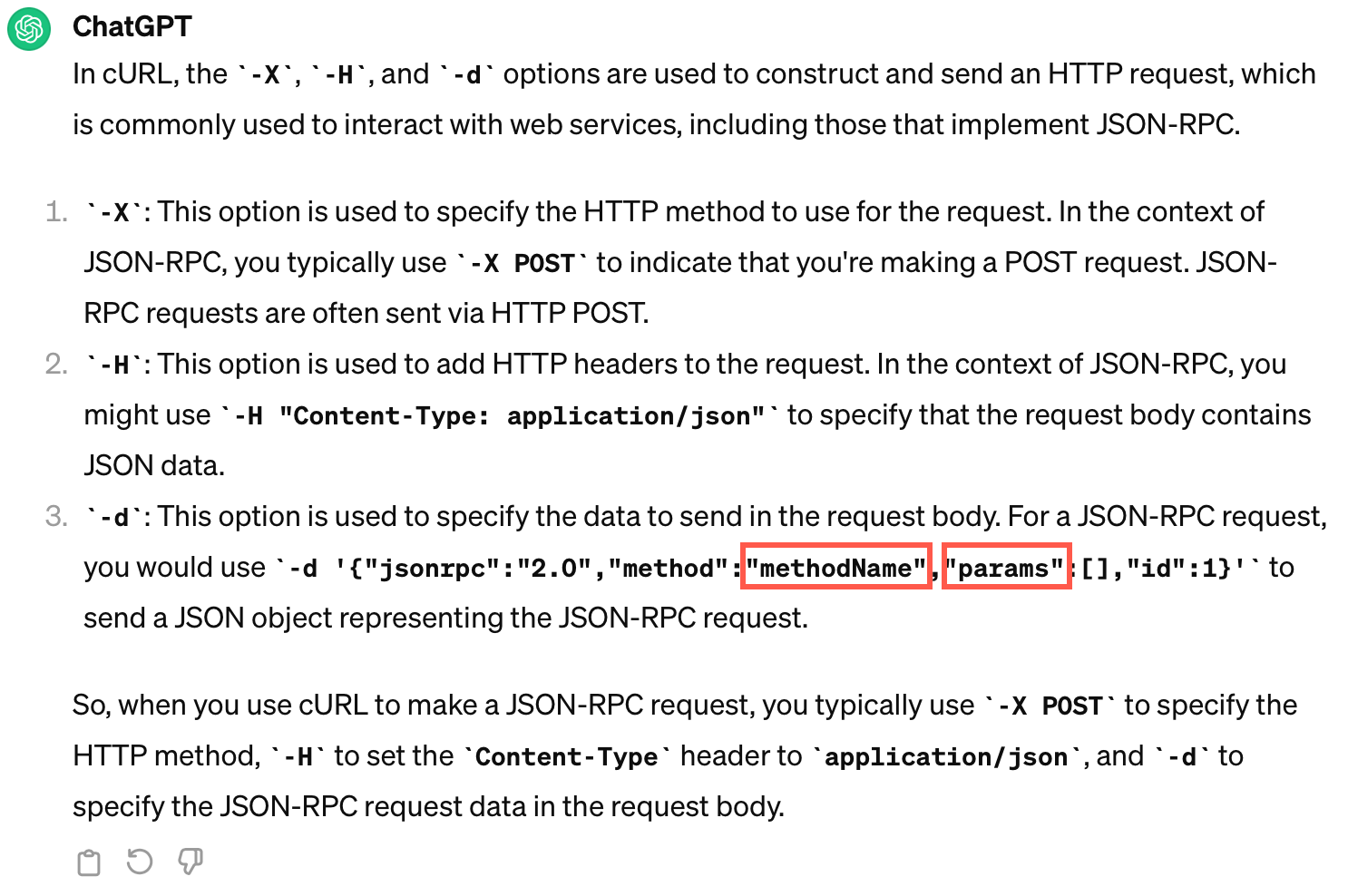
**ETH** — the native cryptocurrency in Ethereum

**Ether** — the native cryptocurrency of Ethereum.

### cURL (client URL)

* Versatile command-line tool to **transfer data to and from servers**.
* cURL enables you to communicate with a server by specifying a **URL** (location) and the data you want to send or retrieve.
* cURL runs on almost every platform, making it ideal for testing communication from local servers to edge devices.
* The most basic cURL command is: curl http://example.com

This retrieves the HTML source from example.com.0



# Tell me what you know about Blockchain

**u Blockchain** is a distributed, immutable ledger (**database**) where transactions made using DApps are recorded in blocks (**ordered records**).

* These blocks are linked together by [**1**] **transaction ID hashes** (a hexadecimal value generated by a cryptographic hash function that identifies the block) to form a blockchain.
* The blockchain is distributed across numerous computers called [**2**] **nodes** (**P2P network**). A node has an **execution client** + **consensus client** (Beacon Node). These two clients work together to sync the Ethereum state.
* The transaction data is kept [**3**] **safe** by the nature of the blockchain architecture. The more nodes having a copy of the distributed ledger, the greater the security. Plus, there’s **transaction ID hashing**.
* The [**4**] **validator node** in the blockchain network determines the **validity of transactions** and the **state of the blockchain** (newly added blocks) and then the other nodes in the blockchain can agree/disagree with findings of the validator node [**5**] **by consensus**. This consensus can be reach by three popular consensus mechanisms: Proof of Work (**PoW**), Proof of Stake (**PoS**), and Proof of Authority (**PoA**).
* [**6**] **dApps** (**decentralized applications**) run on a blockchain or **peer-to-peer** (**P2P**) network of computers (nodes) instead of on a single computer. They are powered by [**7**] **smart contracts**, which are digital agreements coded in the blockchain that auto-execute when specified terms & conditions (T&C) are met.
* [**8**] **Gas** is used on a blockchain to define the **cost of transactions** and the **cost of smart contract execution**. Gas helps to ensure the security, efficiency, and stability of blockchain networks by providing a mechanism for pricing and prioritizing transactions and computations.
* A[**9**] **crypto wallet**—a digital identity management system,enables users to interface with a blockchain. It allows users to store, send, and receive **cryptocurrencies** like *Bitcoin*, *Ethereum*, or any other supported digital asset.
* And finally, many blockchains use [**10**] **tokens** or **cryptocurrencies** to represent the value of a digital assets involved in a transaction within the network. **Cryptocurrencies** can be used to purchase **Non-Fungible Tokens (NFTs)**.
* [**11**] **The blockchain runs on Web3**, which is a **decentralized web** where users have more control over their data and digital assets. It [**12**] **replaces centralized internet services** with decentralized alternatives.
* Blockchain technology has applications across [**13**] **various industries**, including finance (**DeFi**), supply-chain management, healthcare, and others.

Consensys

* While creating technical content at Consensys, I learned about the [**14**] **self-custodial** crypto wallet, **MetaMask**. With this wallet, you’re the only one with access to your private keys.
* I also became familiar with the [**15**] **Hyperledger Besu Ethereum execution client**.
* And finally, I spent most of my time documenting [**16**] **Infura**, a node provider for the Ethereum **L1** blockchain and **L2** (Polygon) blockchains, for both **Mainnet** (Ethereum L1 mainnet) and **Testnets** (Goerli).