Ethereum and the EVM

A Standalone Mini-Course

BSc Blockchain Course

What If Your Software Could Make Promises No One Can Break?

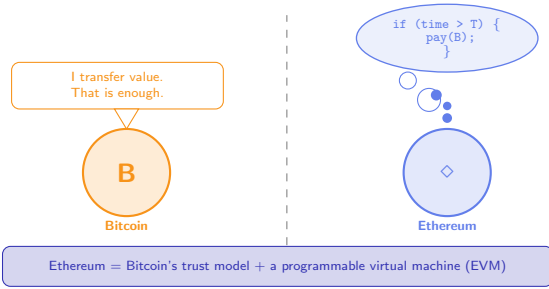
Bitcoin demonstrated that a decentralised network of strangers can agree on ownership of a digital asset without trusting a central authority. But Bitcoin's scripting language is deliberately limited – it was designed for payments, not programs.

In 2013, Vitalik Buterin proposed a radical extension: add a Turing-complete virtual machine to a blockchain so that arbitrary logic could run with the same security guarantees as a money transfer.

Ethereum's core thesis:

- A blockchain can be a **world computer**: a shared, tamper-resistant execution environment.
- Code deployed to it runs identically on every node – *deterministic by consensus*.
- Any two parties can interact via a **smart contract** without trusting each other or a third party.

Bitcoin asks "who owns this coin?" Ethereum asks "what should happen when this condition is met?"



Ethereum's innovation is not a new currency – it is a new *computation substrate*: a global, censorship-resistant, always-on computer where programs execute exactly as written, with no possibility of downtime, fraud, or third-party interference.

Source: Buterin, V. (2013). "Ethereum White Paper." ethereum.org/en/whitepaper

What Would You Build If Bugs Were Permanent and Every Line Cost Money?

In June 2016, an attacker exploited a *reentrancy* vulnerability in The DAO – a decentralised venture fund with $150M in ETH. In a matter of hours, $60M was drained.

The reentrancy pattern in three lines:

1. Attacker calls `withdraw()`.
2. Before balance is updated, the contract calls back to attacker.
3. Attacker calls `withdraw()` again – balance still shows original amount.

The community faced a dilemma:
The Ethereum principle is "code is law" – if the contract says the attacker can withdraw, who is to say that is wrong? But $60M of real value was at stake.

The hard fork decision:
The majority of the community voted to roll back the chain and return funds. A minority refused – they continued the original chain as **Ethereum Classic (ETC)**, preserving the "code is law" principle absolutely.

1. **What is "unstoppable code"?** If a smart contract contains a bug that allows theft, is the thief a criminal? The code permitted the withdrawal – was the attacker just following the rules?
2. **When should a blockchain fork?** The hard fork violated the "immutability" promise. Was this a pragmatic rescue or a dangerous precedent? Who decides when the exception applies?
3. **How do you prevent reentrancy?** The fix is a two-line pattern: update the balance *before* sending funds ("checks-effects-interactions"). Why did the original DAO developers not use it?

Source: Mehar, M. et al. (2019). "Understanding a Revolutionary and Flawed Grand Experiment: The DAO Attack." Journal of Cases on Information Technology, 21(1).

How Do Two Types of Account Run an Entire World Computer?

Ethereum's global state is a key-value store mapping 20-byte addresses to *account objects*. Every entity on the network – whether a person or a program – is one of two account types.

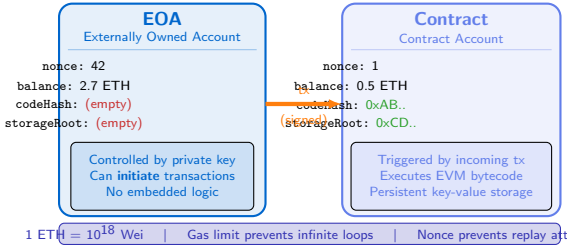
Externally Owned Account (EOA):

- Controlled by a private key (ECDSA secp256k1)
- The only account type that can *initiate* a transaction
- Has no code – `codeHash` = Keccak256 of empty string

Contract Account (CA):

- Created by an EOA sending a "contract creation" transaction
- Address derived from creator address + nonce (deterministic)
- Executes bytecode only when called; cannot act autonomously
- Maintains persistent `storage`: a 256-bit to 256-bit key-value map

Both types share the same four-field structure; they differ only in whether `codeHash` points to actual bytecode.



The four-field account structure is elegant in its uniformity – a human wallet and a complex DeFi protocol look identical at the data level. The difference is whether the code field is empty or contains logic.

Source: Ethereum Yellow Paper (Wood, G., 2014, updated 2024). ethereum.github.io/yellowpaper

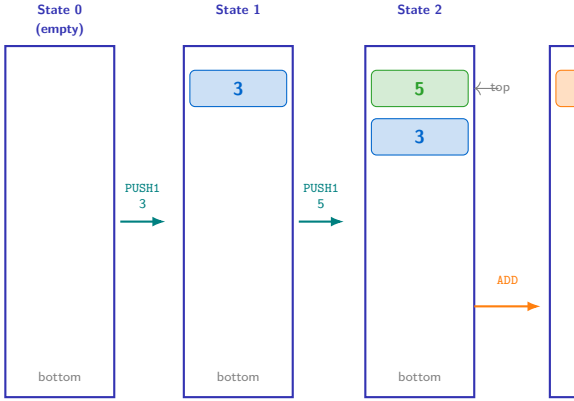
What Happens Inside the EVM When You Add Two Numbers?

The **Ethereum Virtual Machine** is a *stack-based* interpreter: all operands are pushed onto a stack before an opcode consumes them and pushes back results.

Key architectural facts:

- **Word size:** 256 bits (convenient for 32-byte hashes and elliptic curve arithmetic)
- **Stack depth:** maximum 1024 slots – exceeding this raises a stack overflow exception
- **Memory:** byte-addressed, dynamically allocated, cleared after each call
- **Storage:** persistent 256-to-256-bit map (expensive: 20 000 gas to write a new slot)
- **Determinism:** no randomness, no floating point, no system calls – every node gets identical results

Example: computing 3 + 5:
`PUSH1 3` → `PUSH1 5` → `ADD`
Stack transitions are shown on the right.



The stack architecture makes the EVM easy to verify formally: every opcode has a fixed stack effect (how many items it pops and pushes), so a static analyser can prove stack safety before deployment.

Source: Ethereum Yellow Paper, Section 9 (Execution Model). ethereum.github.io/yellowpaper

Why Did Ethereum Reinvent How Users Pay for Transactions?

Gas is Ethereum's anti-spam and resource-pricing mechanism. Every opcode costs a fixed amount of gas; the total cost of a transaction must not exceed the sender's *gas limit*.

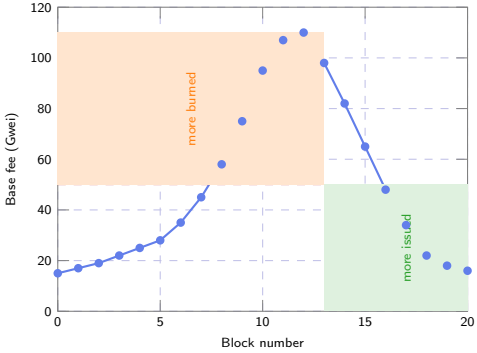
EIP-1559 (August 2021) introduced two-part fees:

- **Base fee:** set by the protocol, burned (removed from supply). Rises when the previous block was $> 50\%$ full; falls when below. Maximum change per block: $\pm 12.5\%$.
- **Priority fee (tip):** paid to the validator as an incentive to include the transaction.

Burning creates deflationary pressure:
When demand spikes (popular NFT mints, DeFi exploits), the base fee rises rapidly and more ETH is burned than issued – net supply falls. During low demand, issuance exceeds burn.

Formula: Total fee $=$ (base fee $+$ priority fee) $\times$ gas used.

Base fee response to block fullness (illustrative):



When base fee $>$ 50% target, each block raises it by up to 12.5%.

EIP-1559 made Ethereum fees more predictable and created a transparent monetary policy: users see the base fee before submitting, and the burn rate ties ETH's supply to network demand.

Source: EIP-1559 (Buterin, Conner, Dudley, Slipper, Norden, Bhargava, 2019). eips.ethereum.org/EIPS/eip-1559

Why Does Storing One Variable Cost More Than a Thousand Additions?

Not all EVM operations cost the same. Arithmetic is cheap; touching persistent state is not. Understanding this hierarchy is essential for writing gas-efficient contracts.

Operation gas cost table (post EIP-2929):

Operation	What it does	Gas
`ADD`	256-bit addition	3
`MUL`	256-bit multiplication	5
`SHA3`	32-byte Keccak256	30
`SLOAD`	Read storage slot (warm)	100
`SLOAD`	Read storage slot (cold)	2,100
`SSTORE`	Write new storage slot	20,000

Why the huge gap?
A storage write persists forever on every full node worldwide. Keeping a 256-bit slot alive costs the network ongoing disk space and I/O. The high gas price compensates all nodes for this perpetual burden.

State bloat: a growing problem

Ethereum's state trie has grown to over 300 GB for archival nodes (2024). Every new storage write makes syncing a fresh node slower.

Proposed mitigations:

- **EIP-4444 (The Purge):** Nodes are no longer required to serve historical data older than one year.
- **State expiry:** Slots not accessed for approximately one year would need a Merkle proof to revive.
- **Verkle trees (The Verge):** Replace MPT with a structure that produces shorter proofs, enabling stateless clients.

Developer rule of thumb:
Pack multiple values into one 32-byte slot (*bit packing*) to reduce SSTORE calls; use `memory` arrays instead of `storage` for temporary data.

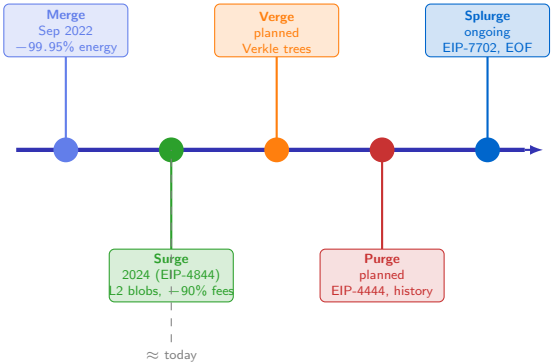
The 6700× difference between ADD and SSTORE is not arbitrary – it reflects the true economic cost of burdening every node on the network forever. Gas pricing is Ethereum's distributed resource allocation mechanism.

Source: EIP-2929 (Buterin & Swende, 2020). eips.ethereum.org/EIPS/eip-2929; EIP-4444 (Stouffer et al., 2021).

How Did Ethereum Change Its Consensus Without a Single Block Missed?

Ethereum's development roadmap is structured around five named phases, each targeting a specific scalability or sustainability bottleneck.

- **The Merge (Sep 2022):** Proof-of-Work retired; Beacon Chain validators took over. Energy consumption fell by approximately 99.95%. Issuance dropped from approximately 13 000 ETH/day to approximately 1 700 ETH/day.
- **The Surge (2024):** EIP-4844 introduced *blob transactions* – cheap, temporary data chunks for L2 rollups. Rollup fees dropped 90%.
- **The Verge (planned):** Verkle trees replace Merkle Patricia Tries. Stateless clients become feasible – a node needs zero historical state to validate blocks.
- **The Purge (planned):** EIP-4444 removes history burden from nodes; protocol complexity pruned (e.g., EVM legacy opcodes).
- **The Splurge (ongoing):** EIP-7702 (account abstraction), EVM Object Format (EOF), single-slot finality.



The roadmap is not a sequential list – phases overlap in research and implementation. The Merge was the hardest because it changed the consensus mechanism of a $200B live network without downtime.

Source: ethereum.org/en/roadmap (2024); Ethereum Foundation Blog: "The Merge" (Sep 2022).

What Do Half a Million Validators and Billions in TVL Actually Mean?

Numbers ground the abstract. Ethereum is not a research prototype – it is production infrastructure with hundreds of billions of dollars at stake.

Why each metric matters:

- **TVL** measures how much value is secured by smart contracts. A high TVL means Ethereum's EVM is trusted by large capital holders.
- **Daily transactions** show actual economic activity – not speculative prices.
- **Active validators** determine the cost of a 51% attack (attacker must control more than 50% of staked ETH, worth tens of billions of dollars).
- **Unique addresses** indicate user adoption, though one user can hold many addresses and one address can serve many users.

These numbers are approximate (2024 snapshots); always verify at etherscan.io or ultrasound.money.

DeFi TVL (Ethereum)

$55B

value locked in smart contracts (Apr 2024)

up from $1B in 2019

Daily Transactions

1.2M

on Ethereum L1 (excl. L2 rollups)

+ 10M+ on L2s

Active Validators

1M+

staking 32 ETH each securing the chain

32 ETH min. stake

Unique Addresses

280M+

ever used Ethereum mainnet

approx. 500K daily active

Ethereum is the only smart-contract platform to simultaneously clear $1B+ daily settlement volume, host a $50B+ DeFi ecosystem, and maintain a decentralised validator set numbering over one million.

Source: DefiLlama (defillama.com); Etherscan (etherscan.io); beaconcha.in – approximate 2024 figures.

Six Takeaways and a Vocabulary

What this mini-course established:

1. Ethereum adds a Turing-complete EVM to Bitcoin's trust model, enabling programmable smart contracts.
2. Two account types – EOAs and Contract Accounts – together produce every Ethereum interaction.
3. The EVM is a 256-bit stack machine; every opcode costs gas, making abuse economically irrational.
4. Storage (SSTORE) costs more than 6 000 times arithmetic (ADD) – write to storage sparingly.
5. EIP-1559 burns the base fee, linking ETH supply to network demand and reducing issuance during high usage.
6. The Merge cut energy by 99.95%; the Surge, Verge, Purge, and Splurge scale throughput without sacrificing decentralisation.

Key vocabulary:

- **EVM** – Ethereum Virtual Machine; deterministic stack computer
- **EOA** – Externally Owned Account; private-key controlled
- **Smart contract** – code deployed at a contract address
- **Gas** – unit of EVM computational work
- **EIP-1559** – fee market reform; burn mechanism
- **MPT** – Merkle Patricia Trie; state database
- **The Merge** – PoW to PoS transition (Sep 2022)
- **L2 rollup** – off-chain execution, on-chain proof

Ethereum's architecture is a single coherent design: gas prices encode resource scarcity, the account model separates identity from logic, the Merkle trie makes state verifiable, and the roadmap continuously removes bottlenecks without breaking backward compatibility.

Source: Ethereum Yellow Paper; EIP-1559; ethereum.org/en/roadmap – BSc Blockchain Course L05.

Your Challenge

You now understand the machine. Time to use it.

Three design problems that require no code – only the concepts from this mini-course:

Problem A: Gas budget audit
A DeFi contract stores 10 user balances in individual storage slots. Estimate the gas cost of initialising all 10 slots. How would you redesign the storage layout to halve the cost?

Problem B: Fee market reasoning
The base fee is 80 Gwei. A user sets `maxFeePerGas` to 90 Gwei and `maxPriorityFeePerGas` to 2 Gwei. Will the transaction be included? What does the validator actually receive?

Problem C: Fork decision
A protocol bug has locked 500 ETH in a contract forever. State the arguments for and against a hard fork to recover the funds. Which Ethereum principle does each side invoke?

Scaffolded activity (30 min):

1. **Choose** one of the three problems above.
2. **Identify** which EVM concepts apply: gas table, fee formula, account model, or consensus principles.
3. **Draft** a 3–5 sentence answer that references at least one specific gas cost or EIP number from this mini-course.
4. **Share** your answer with a partner: can they find a flaw in your reasoning?
5. **Revisit** in L06 (Solidity): how does the language force you to handle the constraint you identified here?

Source: BSc Blockchain Course L05 – Ethereum and the EVM. (c) Joerg Osterrieder 2025.