# Ethereum & Smart Contracts

## A Visual Introduction

Standalone Mini-Lecture

---

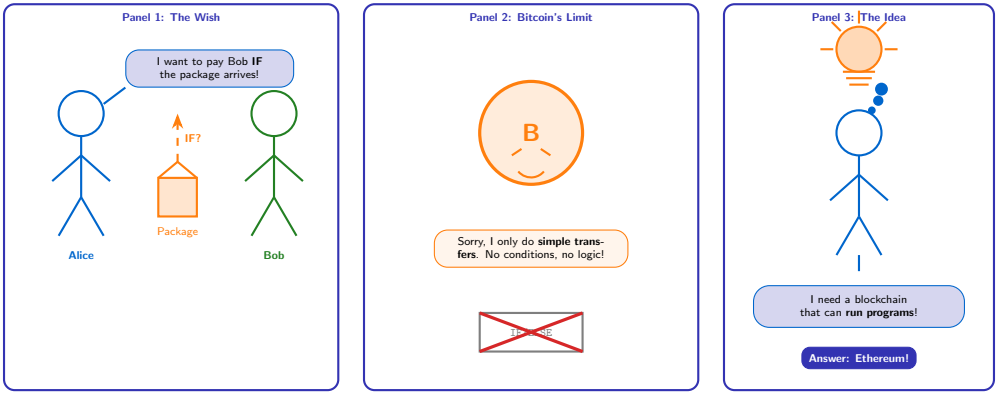
*"Code is law – until it's not"*

Prof. Dr. Joerg Osterrieder

University Lecture Series

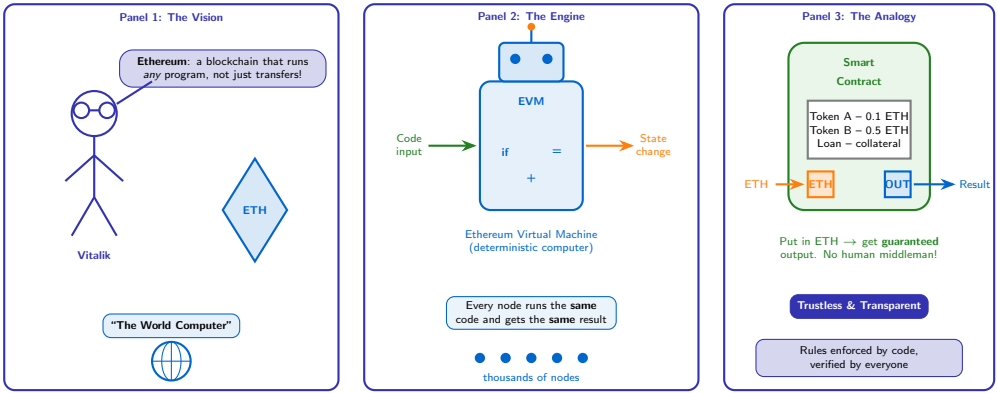
February 21, 2026

# The Programmable Money Problem




**Key insight: Bitcoin's scripting language is intentionally limited. Ethereum was designed from the ground up as a programmable blockchain.**

# Enter Ethereum: The World Computer

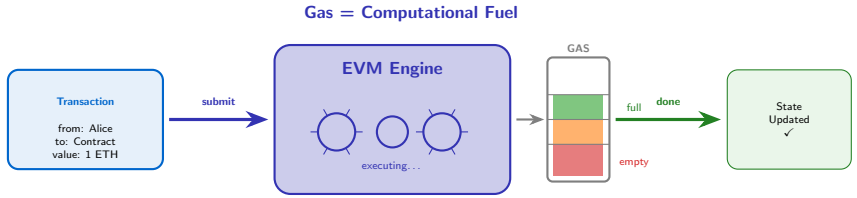



**Key insight: Ethereum extends blockchain from a payment ledger to a general-purpose computation platform — the "world computer."**

# Bitcoin vs Ethereum: Side-by-Side


**Key insight: Bitcoin optimizes for security and simplicity. Ethereum trades some simplicity for programmability — different goals, different designs.**

# How Gas Works: Fueling the World Computer

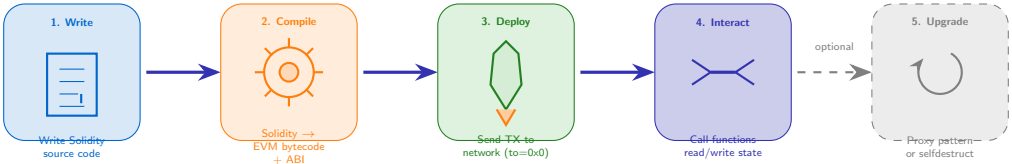



**Transaction Fee** = Gas Used × Gas Price (in Gwei) *1 Gwei* $= 10^{-9}$ *ETH*


**Key insight: Gas prevents spam and infinite loops. Users pay for computation; validators earn fees for processing. EIP-1559 (2021) added a base fee burn mechanism.**

# Smart Contract Lifecycle

## From Code to Blockchain: 5 Steps




**Bytecode:** Low-level instructions the EVM executes. Human-readable Solidity gets compiled down to opcodes like `PUSH`, `STORE`, `CALL`.

**ABI:** Application Binary Interface — the contract's "menu" that tells callers which functions exist and what parameters they accept.

**Immutability:** Once deployed, the code at that address *cannot* be changed. Upgrades require proxy patterns or deploying a new contract.

**Key insight: Deployment is a one-way door. Audit thoroughly before deploying — bugs in smart contracts can mean permanent loss of funds.**

# Your First Smart Contract

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.0;
3
4  contract SimpleStorage {
5      uint256 private storedValue;
6
7      event ValueChanged(uint256 newValue);
8
9      function set(uint256 _value) public {
10         storedValue = _value;
11         emit ValueChanged(_value);
12     }
13
14     function get() public view
15         returns (uint256)
16     {
17         return storedValue;
18     }
19 }
```

**Line 2 – pragma:** Specifies the compiler version. Prevents deploying with incompatible compilers.

**Line 4 – contract:** Like a "class" in OOP. Contains state variables and functions. Deployed as a single unit.

**Line 5 – state variable:** Permanently stored on the blockchain. Costs gas to modify, free to read.

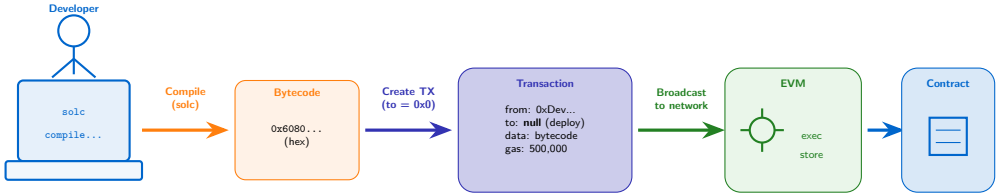
**Lines 9, 14 – functions:**
`set()`: writes state (costs gas)
`get()`: reads state (`view`, free)

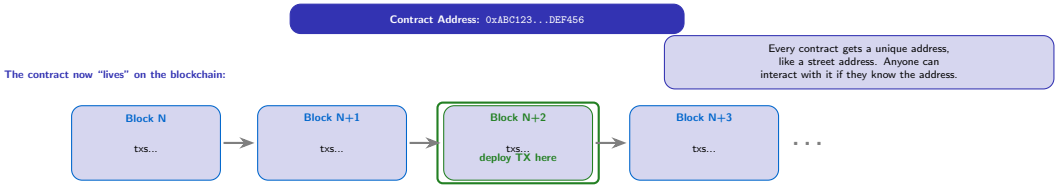
**Line 7 – event:** Logs data to the blockchain. Cheap storage for off-chain apps (e.g., front-ends) to listen to.

**18 lines of code = a complete contract deployed on Ethereum**

**Key insight: Smart contracts are surprisingly concise. This 18-line contract is a fully functional on-chain key-value store. Try it at remix.ethereum.org.**

# Deploying to Ethereum




**Key insight: Deploying is a special transaction with no recipient (to=null). The EVM executes the constructor, stores the bytecode, and assigns a permanent address.**

# Real-World Use Cases

## Ethereum Powers a New Economy

### Decentralized Finance

**DeFi**

Lending, borrowing, trading without banks or brokers.

Examples: Uniswap, Aave, Compound
TVL: $50B+ locked in protocols

### Non-Fungible Tokens

**NFT**

Provable digital ownership of art, music, collectibles.

Examples: CryptoPunks, BAYC
Standard: ERC-721

### DAOs

**DAO**

Community governance via token-weighted voting.

Examples: MakerDAO, Uniswap
Treasury managed by code


### Stablecoins

**Stable**

Price-stable tokens pegged to fiat currencies (e.g., USD).

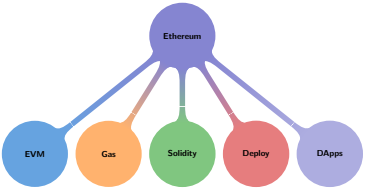
Examples: USDC, DAI, USDT
Market cap: $130B+

**Key insight: Ethereum's programmability enables entire financial systems to run autonomously as code. DeFi alone processes billions in daily volume.**

# Takeaways & What's Next

## Key Principles

- ✓ Ethereum = programmable blockchain
- ✓ Gas prevents abuse and pays validators
- ✓ Smart contracts are self-executing code
- ✓ Once deployed, contracts are (mostly) immutable
- ✓ Real-world impact: DeFi, NFTs, DAOs




## The Ethereum Stack

DApps (front-end)

Smart Contracts (Solidity)

EVM (execution layer)

Consensus (PoS validators)

P2P Network (nodes)

## What's Next

- ERC-20 tokens: build your own currency
- ERC-721: create an NFT collection
- DeFi deep dive: AMMs, lending, flash loans
- Security: common vulnerabilities and auditing

**Next lecture: Build your own ERC-20 token!**

*"In code we trust — but verify with audits."*