Ethereum Interactive Lecture

BSc Blockchain, Crypto Economy & NFTs

Blockchain & Cryptocurrency Course

Spring 2026

This Lecture is Educational

This is not investment, legal, or tax advice.

Cryptocurrency markets are volatile and a meaningful share of retail investors lose money. If you choose to participate after this class, do so with money you can afford to lose, on a clean wallet you generated yourself, and after independent research.

See instructor_kit/security_brief.md.

Duty of care: the lecture leaves you more security-aware than you arrived, not less.

Today, in Six Acts

1. The shared doc (what Ethereum is)
2. Identity to write (accounts, signatures, the test-wallet warning)
3. The cost of writing (transactions, gas)
4. A cell that runs itself (smart contracts, EVM)
5. Reading the doc (Etherscan)
6. Build it ourselves (HelloWorld on Remix VM, optional Sepolia level-up)

Format. 60 minutes lecture, 30 minutes workshop. Two interactive checkpoints. One artifact: a deployed smart contract.

Tools you need. A browser tab. That is the entire toolchain.

First principles, honest about the work.

Act 1: Ethereum as a Shared, Signed, Append-Only Google Doc

We will treat Ethereum as a **shared, append-only Google Doc** where every edit must be signed and pays a small fee.

Imagine one Google Doc:

- Visible to everyone (*shared*)
- Every edit must be **signed** (you cannot pretend to be someone else)
- No one can **edit** a past line; you can only **append**
- Each character costs a tiny fee

That is Ethereum. The doc is the chain. The fees are gas. The signed edits are transactions. Cells with little scripts that run themselves are smart contracts.

[Diagram: a Google-Doc icon with three labeled tags above it: SHARED, SIGNED, APPEND-ONLY. Below the icon a row of three small rows with timestamps, each row a different sender, each row tagged with a tiny coin icon for fee.]

Vocab: shared = global, signed = cryptographically authenticated, append-only = immutable history.

The Four Pieces

Google Doc	**Ethereum**
Your Google identity	An **account** (a public/private key pair)
A signed edit you make	A **transaction**
A doc cell with a tiny script that runs itself	A **smart contract**
Per-character cost of writing	**Gas** (a fee paid in ETH)

Hold this table in your head. Every concept later is one of these four.

The analogy is your scaffolding; we will name where it breaks at slide 20.

Who can write in the doc?

Accounts. Signatures. The wallet warning.

Time check: minutes 12 to 22.

An Ethereum Account is a Key Pair

Public key (the address)

- Looks like `0xAbCd...1234`
- Anyone can see it
- Like your username

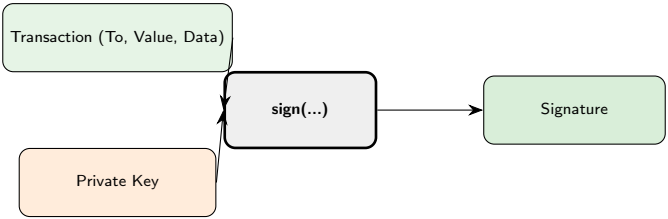
Private key (the secret)

- A 256-bit number you must never share
- Lets you SIGN transactions
- Like your password, except no reset button

Your wallet (e.g. MetaMask) is software that stores the private key locally and signs transactions on your behalf. It is NOT a bank account; the wallet does not hold ETH. The chain holds the ETH. The wallet holds the key that proves you own the chain entry.

Vocab: wallet = Geldborse (Schlusselverwaltung). It manages keys, not money.

What a Signature Proves



A signature proves: this transaction was authorized by the holder of the private key for this address. Anyone with the matching public key can verify the signature; nobody without the private key can reproduce it.

Without that proof, the network rejects the transaction.

This is the same mechanism that signs HTTPS certificates, SSH keys, and SWIFT messages.

TEST WALLET ONLY, NEVER REUSE THIS SEED

If you set up a wallet today, treat it as disposable.

[STOP] **Never fund this wallet with real ETH.**
[STOP] **Never import this seed into a real-money wallet.**
[STOP] **Never use the same seed for testnet and mainnet.**

For real-money use later, install a fresh wallet on a clean machine and generate a NEW seed.

See instructor_kit/security_brief.md.

The seed phrase is the master key. One leak = every chain compromised forever.

Interactive: Think, Then Pair (or Card)

Prompt (60 seconds thinking, 60 seconds pair, 60 seconds share):
What would happen if anyone in the world could pretend to be you and sign transactions from your account?

Alternative (no speaking required):
Write your answer on a card. Pass it to the instructor. We read three sample cards aloud anonymously.

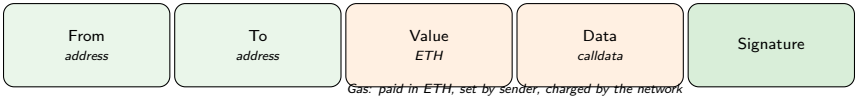
This is one of two interactive checkpoints. The other is the on-chain treasure hunt at slide 21.

Why does each edit cost a fee?

Transactions. Gas. The dollar number.

Time check: minutes 22 to 32.

Every Edit is a Transaction. Every Transaction Pays Gas.



Why charge gas at all?

- Computation is not free. Every node in the world re-runs your transaction.
- Without a fee, anyone could spam the network with infinite-loop transactions.
- Gas is also how validators (the nodes that order transactions into blocks) get paid.

Vocab: gas = Gas (Transaktionsgebuhr). nonce = laufende Transaktionsnummer.

Gas in Dollars: A Worked Example

A simple transfer (sending ETH from one account to another):

- Gas used: 21,000 (a fixed minimum)
- Gas price (typical, mainnet): 20 gwei
- ETH price (illustrative): $3,000

Fee $= 21{,}000 \times 20$ gwei $= 420{,}000$ gwei $= 0.00042$ ETH $\approx \$1.26$.

A contract deployment (like our HelloWorld) typically uses 100,000 to 200,000 gas. At the same prices, that is $6 to $12 on mainnet.

On Sepolia testnet, the ETH is fake. The fee is in fake ETH. The cost is zero in real money.

On Remix VM, the chain is in your tab. The fee is fake fake ETH. The cost is also zero.

One ETH $= 10^{18}$ *wei. One gwei* $= 10^{9}$ *wei. Numbers are illustrative; mainnet gas prices vary by orders of magnitude.*

90 seconds. Stand up. Look out the window.

Watch Me Deploy on a SHARED Chain (3 minutes)

- I will deploy the same HelloWorld contract you will deploy in 25 minutes.
- BUT I will deploy it on Sepolia, a chain shared by every student in this room and by everyone else in the world running an Ethereum testnet client.
- Then we open Etherscan and look at it together.
- **This is the moment that proves the analogy.** Everyone in this room (and outside it) sees the same line in the same doc.

Live page if available; pre-recorded video at instructor_kit/sepolia_anchor_assets/anchor_demo.mp4 is the source of truth.

Slide 16 is the diagram you watch alongside the demo.

Private vs Shared: Two Different Chains

Remix VM (your workshop)

Browser tab
private chain

[WARN] **Visible only to YOUR tab.**
Closing the tab erases the chain.

Sepolia (anchor demo)

Global testnet
shared chain

[OK] **Visible to the world.**
The chain is shared, the ETH is fake.

Honesty pattern. At every step of the workshop, ask which chain you are on. Same code can run on both, and the results LOOK the same; the social fact is different.

Mainnet is a third chain: shared, real ETH, real money. Out of scope today.

What is a smart contract, really?

The doc cell with a script.

Time check: minutes 38 to 48.

HelloWorld: Three Lines That Make a Contract

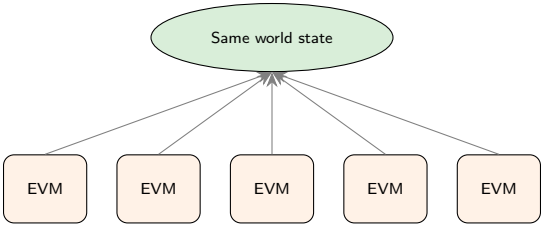
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;
contract HelloWorld {
    string public greeting = "Hello, Class";
}
```

Read it.

- One state variable (`greeting`), public, holds a string.
- Solidity auto-generates a getter for any `public` state variable. So `greeting()` is callable from outside without writing it.
- No constructor, no other functions. Three meaningful lines.

This is what every student deploys in the workshop.

The EVM: Why Every Node Agrees



deterministic + replicated = consensus

Two properties:

- **Deterministic.** The same input always produces the same output. No randomness, no system clocks, no file I/O inside contract execution.
- **Replicated.** Every node runs every transaction. They compare results. If they disagree, the dissenter is wrong.

This is why a contract is "a cell that runs itself". Nobody in particular runs it; every node does, and they agree.

Vocab: EVM = Ethereum Virtuelle Maschine.

Where the Google-Doc Analogy Breaks

	Google Doc	**Ethereum**
Identity per edit	Loose (login optional)	**Cryptographic signature required**
Edit a past line	Yes (paste over it)	**No, append-only**
Undo a transaction	Ctrl+Z	**No undo, ever**

The single most important difference. Ethereum is append-only. Once a transaction is mined, you can write a new transaction that updates state, but you cannot edit or delete the historical record. This is the property that makes the chain auditable forever, AND the property that makes mistakes expensive forever.

The analogy got us to slide 20. Now we know its limits.

Interactive: On-Chain Treasure Hunt

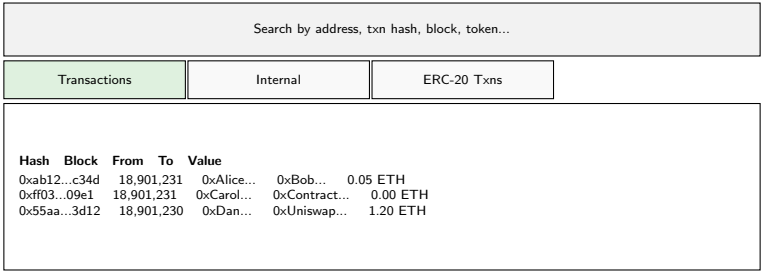
Open https://etherscan.io and search for block `46147`.

- Type `46147` into the top search bar; if Etherscan does not navigate directly, click the *Blocks* tab and enter `46147` there. Either route works.
- This is the block holding the very first non-coinbase transaction on Ethereum mainnet (year 2015).
- In your pair (or alone, your choice), find: the sender, the recipient, the value sent (in ETH), the gas paid.
- Why this tx and not the famous ENS first registration? The ENS one was used in week 3. We avoid repetition.

Hard reset checkpoint. If at this point your screen is still on the slides and not on Etherscan and Remix, the instructor projects Remix on the main screen and we rebuild together. No student gets left in the slides.

This block is older than this course. The append-only property has held for a decade.

Etherscan is a UI on the Shared Doc



Etherscan does not own the chain. It reads the chain and renders it as HTML. Other explorers (Blockscout, Beaconcha.in) read the same chain. If Etherscan is down, the chain is fine.

The companion notebook (take-home) reads the chain via web3.py: same chain, different UI.

Bridge: What You Are About to Build

1. Open Remix in a fresh tab: `https://remix.ethereum.org`
2. Paste `HelloWorld.sol` from the lab guide
3. Compile (button click; status turns to [OK] **compiled successfully**)
4. Deploy on Remix VM (the in-browser private chain)
5. Click `greeting` on the deployed contract; read `Hello, Class`

Time budget. 30 minutes for steps 1 through 5, plus an optional 5 minutes for the Sepolia level-up if you arrived with a funded MetaMask.

Stragglers' protocol. At minute 80 (i.e. 5 minutes before close), the TA pulls aside any student still without a deployed contract for a 5-minute focused walkthrough. Nobody leaves without an artifact.

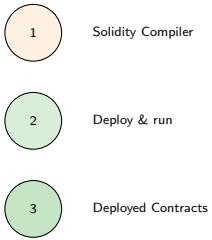
Lab guide: ../lab_guide.md. Pre-class prep was encouraged, not required.

Workshop: The Three Critical Clicks

1. Solidity Compiler panel -> [Compile]
   Status: green check + "compiled successfully"

2. Deploy & run transactions panel
   Environment: Remix VM (Cancun)
   -> [Deploy]
   Wait for green check in the bottom terminal

3. Deployed Contracts -> expand HelloWorld
   -> click [greeting]
   Output: string: greeting Hello, <your name>



The greeting string is editable: open HelloWorld.sol and change "Hello, Class" to "Hello, ¡your name¿" before clicking Compile.

Take-Home and Post-Quiz

Take-home notebook

- File: `notebook/ethereum_companion.ipynb`
- Open in Colab AFTER class.
- Reads your contract by address.
- Decodes the block-46147 historical tx.
- 2-minute instructor demo here, no student execution.

Post-quiz (5 minutes, 50 points)

- 5 multiple-choice questions on concepts.
- No question requires you to have a deployed contract address.
- Even if you missed the deploy, you can score 100%.
- File: `quizzes/ethereum_interactive_post_quiz.md`

Concept-based grading: nobody is penalised for an infrastructure failure they did not cause.

Recap, in Five Lines

1. Ethereum is a shared, signed, append-only Google Doc.
2. Identity to write = an account = a key pair. Wallet = key manager, not a bank.
3. Every edit is a transaction. Every transaction pays gas.
4. Smart contracts = doc cells that run themselves, replicated across thousands of nodes.
5. The chain is shared. Etherscan is just a UI on the shared chain.

What to never do. [STOP] **Never reuse a workshop seed for real money.** [STOP] **Never trust class infrastructure with anything you cannot afford to lose.**

security_brief.md is the canonical version of these warnings.

Appendix: The Road Ahead (Optional)

Two big topics we did not cover today:

- **Layer 2 (L2) networks.** Most real Ethereum activity in 2026 happens on rollups (Arbitrum, Optimism, Base, ZKsync). Same accounts, same Solidity, much cheaper gas. The mainnet you saw today is the settlement layer they anchor to.
- **Account abstraction (ERC-4337).** The "account = key pair" rule is being relaxed. Smart-contract-based accounts can have multi-factor authentication, social recovery, and sponsored gas. The wallet you generated today is still an EOA (Externally Owned Account); modern wallets increasingly are not.

Where these get covered. Module F of this course; the existing W04 (Create Your DAO) workshop touches on contract accounts.

The single optional slide of the lecture. Read for context; not on the post-quiz.

Questions?

If you take one thing home: the chain is shared, the wallet is a key, the seed is the master key, and a workshop seed is disposable.

Files for after class.

- Lab guide: `ethereum_interactive_lecture/lab_guide.md`
- Companion notebook: `ethereum_interactive_lecture/notebook/ethereum_companion.ipynb`
- Security brief: `ethereum_interactive_lecture/instructor_kit/security_brief.md`
- Post-quiz answer key (released after the quiz): `ethereum_interactive_lecture/quizzes/ethereum_interactive_post_quiz.md`