Ethereum & Smart Contracts: Programmable Money and Its Consequences

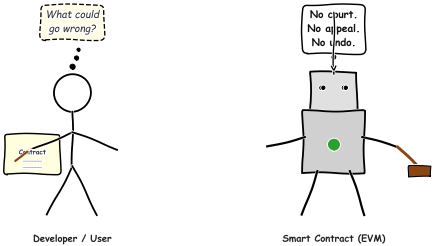
Four Tensions of Unstoppable Code

Prof. Dr. J. Osterrieder

BSc Blockchain, Crypto Economy & NFTs

Spring 2026

[Cartoon] What If a Contract Could Enforce Itself?



You sign a contract. A robot reads it, locks the money, and executes automatically.

No lawyer. No court. No appeal.

What could go wrong?

The robot cannot read intent, only code. It cannot forgive mistakes. It cannot call a judge.

And once deployed, nobody can turn it off.

This lecture explores what happens when code becomes law — and when the code has bugs.

Learning Objectives

By the end of this lecture, you will be able to:

1. Explain how Ethereum's account-based model differs from Bitcoin's UTXO model
2. Describe the gas mechanism and how EIP-1559 determines what users pay
3. Apply the ERC-20 token standard to trace an approve-transferFrom workflow
4. Compare ERC-20, ERC-721, and ERC-1155 on fungibility, batch operations, and use cases
5. Evaluate whether a proposed smart contract is safe, cost-effective, and appropriate

No prerequisites beyond Blockchain Foundations.

These objectives span Bloom levels from Understand to Evaluate.

Why Wasn't Bitcoin Enough?

Bitcoin's limitation:

Bitcoin can transfer value, but it cannot express conditions.

You cannot write: "Send 10 BTC **if** temperature exceeds 30 degrees."

Bitcoin's scripting language is intentionally limited — simple transfers, multi-signature wallets, time locks. Nothing more.

This was a deliberate design choice: fewer features mean fewer bugs.

Ethereum's answer:

Add a virtual computer to the ledger. Now the blockchain can *compute*, not just record.

Analogy

Bitcoin is a **calculator**: it adds and subtracts balances.

Ethereum is a **smartphone**: it runs arbitrary applications.

Same underlying technology, vastly different capabilities — and vastly different risks.

Bitcoin's Script language is intentionally limited to prevent bugs. Ethereum made the opposite choice.

What Happens When You Send Money to a Program?

The Vending Machine Thought Experiment

You walk up to a vending machine. Insert coins. Press a button. Get a drink.

Simple. Predictable. Mechanical.

Now imagine a vending machine that holds **millions of dollars**, runs **24 hours a day, 365 days a year**, and **nobody can unplug it**. Nobody owns it. Nobody maintains it. Nobody can issue a refund.

If the machine has a bug, it keeps running — with your money inside.

That is a smart contract.

When was the last time you trusted a machine with your money? An ATM? A parking meter? What made you trust it?

A smart contract is a vending machine that nobody owns, nobody can unplug, and nobody can refund.

What is a Smart Contract, Really?

Definition

A **smart contract** is a program stored on a blockchain that executes automatically when predefined conditions are met.

Three key properties:

1. **Self-executing:** No human intervention needed after deployment. The code runs when triggered.
2. **Immutable:** Once deployed, the code cannot be changed. No patches, no updates, no hotfixes.
3. **Transparent:** Anyone can read the code and verify what it does. No hidden clauses.

Analogy: A vending machine is a physical smart contract — insert coins (input), receive drink (output), no negotiation possible.

The term was coined by Nick Szabo in 1994, two decades before Ethereum existed.

How is Ethereum Different From a Normal Computer?

Normal computer:

- Runs locally on one machine
- Crashes? Reboot it
- Bug found? Patch it
- Too slow? Upgrade hardware
- Free to run computations

You control everything. You fix everything.

Ethereum Virtual Machine (EVM):

- Runs on 100,000+ computers simultaneously
- Cannot crash (no single point of failure)
- Bugs live forever (immutable code)
- Every instruction costs money (gas)
- Nobody controls it

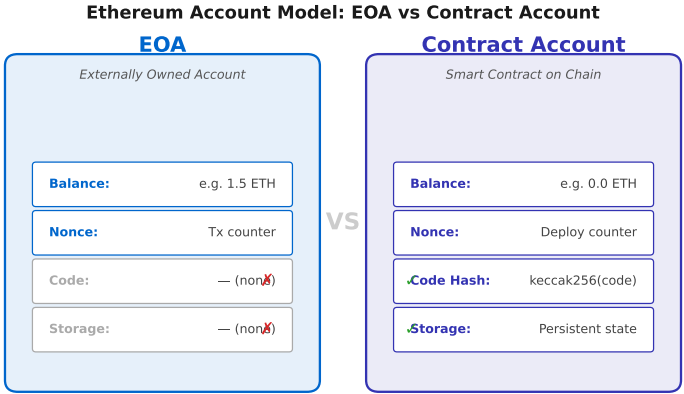
Nobody controls it. Nobody fixes it.

Key Insight

The EVM is the world's most expensive computer — and that expense IS the security.

Redundancy is the price of trustlessness: 100,000 copies ensure no single point of failure.

Two Kinds of Accounts



- **What you see:** Two account types with different control mechanisms
- **Key pattern:** EOAs initiate transactions; contracts only respond when called
- **Takeaway:** Every action on Ethereum starts with a human pressing "send"

Contract accounts cannot initiate transactions — they wait to be called, like a vending machine waiting for coins.

Bitcoin vs Ethereum: Two Philosophies

Bitcoin vs Ethereum: Six Dimensions

Dimension	B Bitcoin (BTC)	E Ethereum (ETH)
Scripting	Limited (Script)	Turing-Complete (Solidity)
State Model	UTXO	Account-Based
Purpose	Value Transfer	Programmable Logic
Throughput	~ 7 TPS	~ 15 TPS
Launch Year	2009	2015
Consensus	Proof of Work	Proof of Stake

- **What you see:** Two systems compared on six dimensions
- **Key pattern:** Ethereum gained programmability by accepting more complexity and risk
- **Takeaway:** Bitcoin stores value; Ethereum computes with value

These are complementary systems, not competitors — each optimizes for different properties.

Follow One Smart Contract From Code to Execution

Five steps from idea to running contract:

1. **Write** the contract in Solidity (a programming language)
2. **Compile** it into EVM bytecode (machine instructions)
3. **Deploy** it by sending a special transaction to Ethereum
4. **Address** is assigned — the contract now lives on-chain
5. **Call** functions by sending transactions to that address

After step 4, the contract exists as long as Ethereum exists.

Worked Example — Concert Tickets:

Alice writes a ticket contract:

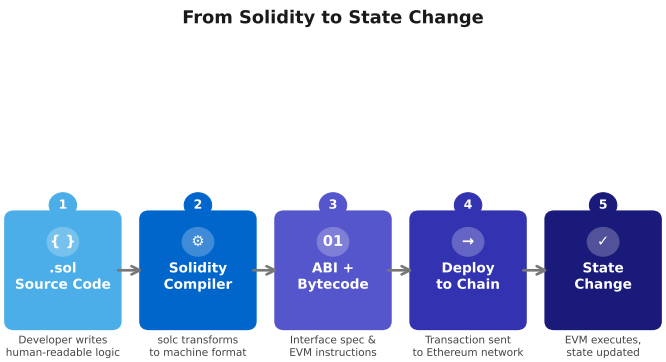
- Price: 1 ETH per ticket
- Supply: 100 tickets
- Rule: one ticket per address

Bob sends 1 ETH to the contract address. The contract checks the rules, accepts the payment, and issues Bob a ticket token.

No box office. No middleman. No scalpers (one per address).

Deployment is permanent — once a contract has an address, it exists as long as Ethereum exists.

How Does Solidity Become Machine Code?



- **What you see:** A five-step pipeline from human-readable code to on-chain execution
- **Key pattern:** The ABI is the menu; the bytecode is the kitchen
- **Takeaway:** Compilation is a one-way street; once deployed, you cannot "uncompile" and edit

The ABI (Application Binary Interface) is what wallets and websites use to interact with contracts.

Worked Example: A Simple Token Contract

Pseudocode (not Solidity — the real language has more syntax, but the logic is identical):

```
Contract MyToken:
  total_supply = 1,000,000
  balances = mapping(address -> number)

  function transfer(from, to, amount):
    check:  balances[from] >= amount
    balances[from] -= amount
    balances[to] += amount
    emit TransferEvent(from, to, amount)
```

Worked numerical example:

Before:
Alice = 100 tokens, Bob = 0 tokens

Call: `transfer(Alice, Bob, 30)`

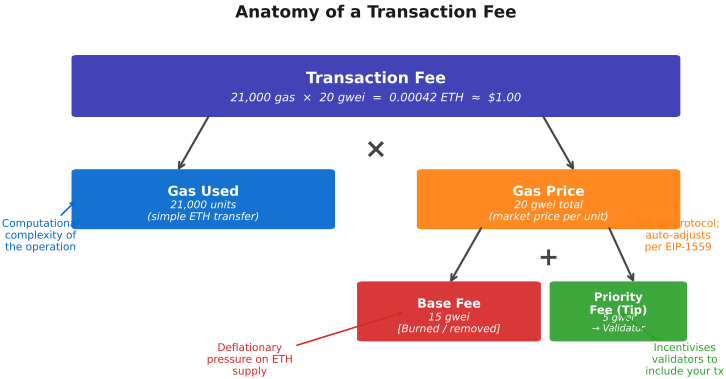
Check: 100 ¿= 30? Yes.

After:
Alice = 70 tokens, Bob = 30 tokens

Event emitted:
"Alice sent 30 to Bob"

This is pseudocode, not Solidity. The real language has more syntax, but the logic is identical.

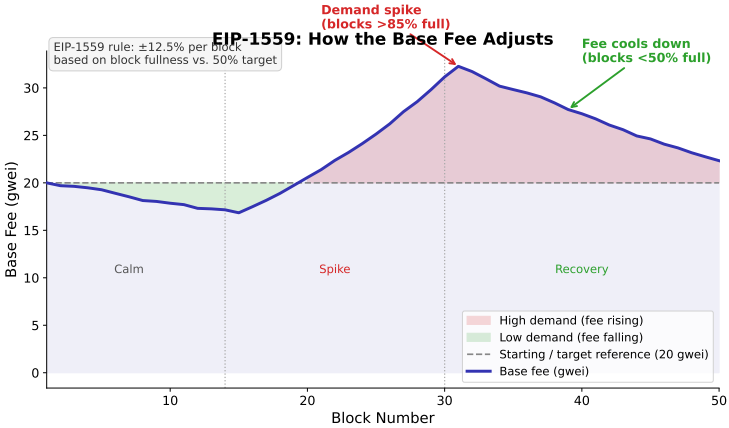
What Does It Actually Cost?



- **What you see:** A breakdown of a transaction fee into its components
- **Key pattern:** The base fee adjusts automatically based on demand — more users means higher fees
- **Takeaway:** Gas is not a bug — it is the price of decentralized computation

Without gas, an infinite loop would freeze every node on the network forever.

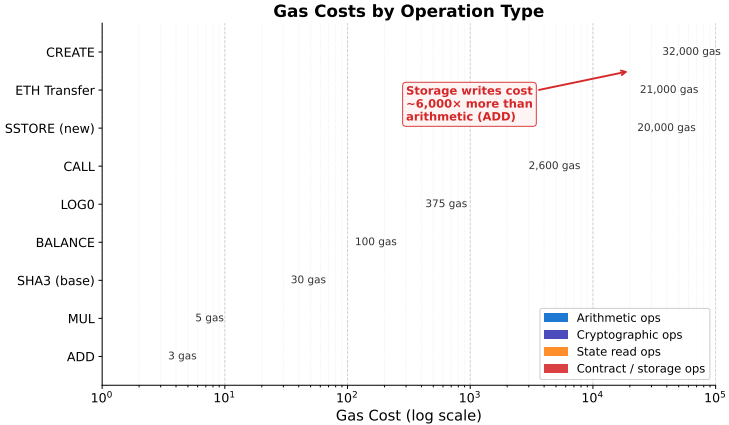
How Does EIP-1559 Set the Price?



- **What you see:** A time series of base fee rising when demand exceeds capacity and falling when demand drops
- **Key pattern:** The protocol acts like a thermostat — it adjusts the price to target 50% block utilization
- **Takeaway:** EIP-1559 replaced a blind auction with a predictable pricing mechanism

Before EIP-1559 (Aug 2021), users had to guess the right gas price — overpaying or getting stuck.

Which Operations Cost the Most Gas?



- **What you see:** Gas costs varying by 4 orders of magnitude across operation types
- **Key pattern:** Writing to storage (SSTORE) costs 6,000x more than arithmetic (ADD) because storage persists forever
- **Takeaway:** Smart contract optimization is about minimizing storage writes, not arithmetic

Worked example: A simple transfer costs 21,000 gas. At 20 gwei: 21,000 × 20 = 420,000 gwei = 0.00042 ETH (about $1.00 at $2,400/ETH).

Storage is expensive because every node must store it forever. Computation is cheap because it is temporary.

What is an ERC-20 Token?

ERC-20 is a standard interface — 6 functions:

1. `totalSupply()` — how many tokens exist
2. `balanceOf(addr)` — how many tokens an address holds
3. `transfer(to, amount)` — send your tokens
4. `approve(spender, amount)` — allow someone to spend
5. `allowance(owner, spender)` — check approval amount
6. `transferFrom(from, to, amount)` — spend approved tokens

Any contract implementing these 6 functions is "ERC-20 compatible."

Why standardize?

Because wallets, exchanges, and other contracts can interact with **any** ERC-20 token without custom code.

One standard, 500,000+ tokens.

Token vs Coin

A **coin** has its own blockchain (BTC, ETH).

A **token** lives as a smart contract ON someone else's blockchain (USDC on Ethereum).

ERC stands for Ethereum Request for Comments — the community standards process.

Worked Example: The Approve-TransferFrom Dance

Scenario: Alice wants to swap 100 USDC for DAI on a decentralized exchange (DEX).

Step-by-step:

1. **Alice calls** `approve(DEX, 100)` on the USDC contract
   Translation: "DEX, you may spend up to 100 of my USDC."
2. **Alice calls** `swap()` on the DEX contract
   Translation: "DEX, execute my trade now."
3. **DEX calls** `transferFrom(Alice, DEX, 100)` on the USDC contract
   Translation: "USDC contract, move 100 from Alice to me." The USDC contract checks the approval and transfers.
4. **DEX calls** `transfer(Alice, 98)` on the DAI contract
   Translation: "DAI contract, send 98 DAI to Alice" (minus 2% fee).

Why two steps? The approve step is the safety mechanism — without it, the DEX cannot touch Alice's tokens. She controls the permission.

The two-step pattern (approve then transferFrom) prevents contracts from spending tokens without permission.

Why is Immutability Both a Feature and a Threat?

Feature:

- Nobody can censor transactions
- Nobody can freeze your funds
- Nobody can reverse a payment
- The rules are public and permanent
- "Code is law" — predictable, neutral

No government, company, or hacker can alter a deployed contract.

Threat:

- Nobody can fix a bug after deployment
- Nobody can patch a vulnerability
- Nobody can undo a theft
- Nobody can update outdated logic
- "Code is law" — even bad code

2016: The DAO — $60M drained through a single bug in 3 hours.

Central Tension

Immutability means your best code and your worst bugs both live forever.

This is the central tension of smart contracts: the same property that makes them powerful makes them dangerous.

What is a Reentrancy Attack?

The pattern:

1. Contract A calls Contract B (sends money)
2. Contract B calls *back into* Contract A before A finishes
3. Contract A has not updated its records yet
4. Contract A sends money *again*
5. Repeat until drained

The contract sends money before recording that it sent money.

ATM analogy:

You withdraw $100. The ATM dispenses cash but has not updated your balance yet. You press "withdraw" again before it finishes. You get another $100.

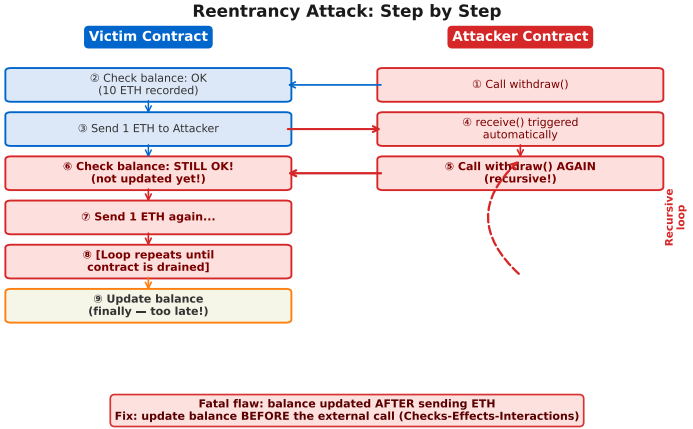
The fix — Checks-Effects-Interactions:

1. **Check** conditions (balance sufficient?)
2. **Effect** state changes (update balance)
3. **Interact** with external contracts (send money)

Update your records BEFORE sending money.

The checks-effects-interactions pattern: first check conditions, then update state, then interact with external contracts.

How Does Reentrancy Actually Work?



- **What you see:** A recursive loop where the attacker re-enters the vulnerable function before it updates state
- **Key pattern:** The contract sends money before recording that it sent money — the classic ordering error
- **Takeaway:** In smart contracts, the order of operations is a security decision, not a style preference

Every major reentrancy exploit followed the same pattern: send first, update later.

Most Common Smart Contract Vulnerabilities

By frequency:

1. Reentrancy (recursive callbacks)
2. Access control (missing permission checks)
3. Oracle manipulation (fake price data)
4. Integer overflow (arithmetic wrap-around)

Most exploits target logic errors — simple mistakes that auditors miss.

By severity (dollars lost):

1. Flash loan attacks ($200M+ per incident)
2. Reentrancy ($60M–$150M per incident)
3. Oracle manipulation ($50M–$100M)
4. Front-running / MEV (ongoing extraction)

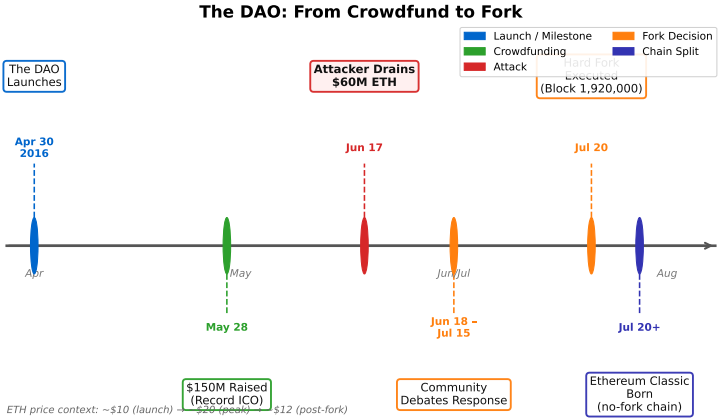
$3.8 billion lost to exploits in 2022 alone.

Key Insight

Most exploits target logic errors, not cryptographic weaknesses — the math is fine, the code is wrong.

Smart contract security is a software engineering problem, not a cryptography problem.

The DAO: The Hack That Split Ethereum



- **What you see:** A 3-month timeline from the largest crowdfund in history to the most controversial fork
- **Key pattern:** The community chose to rewrite history — violating immutability to save $60M
- **Takeaway:** "Code is law" lasted until the losses were big enough

The fork created two Ethereums: Ethereum (forked) and Ethereum Classic (original chain).

Fungible vs Non-Fungible vs Multi-Token

Token Standards: ERC-20 vs ERC-721 vs ERC-1155

- **What you see:** Three token standards compared on four dimensions
- **Key pattern:** ERC-1155 combines properties of both ERC-20 and ERC-721 in a single contract
- **Takeaway:** The standard you choose determines what your token CAN and CANNOT do

Most gaming platforms now use ERC-1155 because it handles both currencies and items in one contract.

How Much Value is Locked in Smart Contracts Today?

Smart contract value (Jan 2026):

- DeFi total value locked: $120B+
- Stablecoins market cap: $200B+
- NFT trading volume (2024): $15B+
- DAO treasuries: $25B+

Combined: over $360 billion controlled by autonomous programs.

For perspective:

More value is controlled by smart contracts than is held by most national banks.

The GDP of Finland is approximately $300 billion.

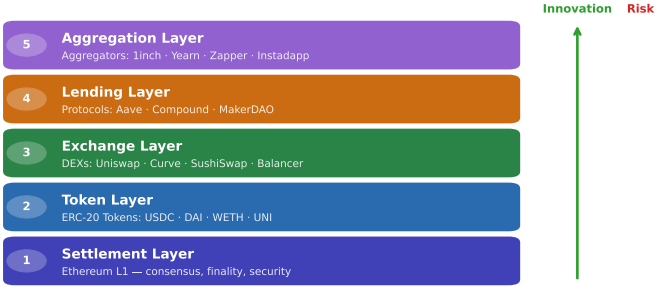
Smart contracts now manage more value than the entire economic output of many countries — with zero employees, zero offices, zero business hours.

These programs never sleep, never take holidays, and never ask for a raise.

Data: DefiLlama, CoinGecko (Jan 2026). These numbers change daily.

Why Do People Call DeFi “Money Legos”?

DeFi Composability: Money Legos



- **What you see:** Five layers stacked, each using contracts from the layer below
- **Key pattern:** Each layer is permissionless — anyone can build on top without asking
- **Takeaway:** Composability is both DeFi's superpower (rapid innovation) and its greatest risk (cascading failures)

No API keys, no partnerships, no approval needed — just call the contract.

Who Benefits and Who Gets Hurt?

Winners:

- **Developers** — permissionless innovation, global market from day one
- **Unbanked populations** — 1.4 billion people gain financial access
- **Innovators** — build financial products without banking licenses
- **Transparency advocates** — every transaction is publicly auditable

Losers:

- **Users who lose funds** — bugs, scams, lost keys are permanent
- **Regulators** — code moves faster than law
- **Non-technical users** — complexity creates a steep learning curve
- **Exploit victims** — $3.8B lost in 2022 alone

Smart contracts do not discriminate: they are equally accessible to innovators and scammers.

Programmable money democratizes finance AND democratizes financial risk.

Should Smart Contracts Replace Traditional Contracts?

What smart contracts do well:

- Escrow (hold money until conditions met)
- Token issuance (create and distribute)
- Governance (transparent voting)
- Automated market making (24/7 trading)

Mechanical, deterministic, binary decisions.

What they cannot do:

- Interpret ambiguity ("reasonable effort")
- Access off-chain data without oracles
- Be updated after deployment
- Handle disputes or exceptions

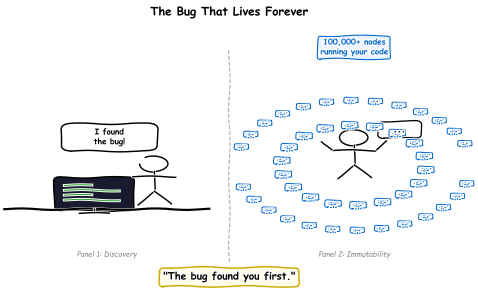
Judgment, nuance, context, forgiveness.

Key Insight

Smart contracts complement legal contracts — they do not replace them.

The sweet spot: automate the mechanical parts, leave judgment to humans.

[Cartoon] The Bug That Lives Forever



We started with a question: *what if contracts could enforce themselves?*

Ethereum's answer: powerful, transparent, unstoppable code.

The cost? Every bug is also unstoppable.

The four tensions:

1. Programmability vs complexity
2. Immutability vs fixability
3. Transparency vs exploitability
4. Composability vs systemic risk

These are not bugs — they are fundamental design trade-offs.

From self-executing promises to self-executing bugs — the four tensions of Ethereum.

Key Takeaways

Five ideas to remember from today:

1. **Programmability:** Ethereum turned a ledger into a computer — code becomes law
2. **Cost:** Gas is the price of decentralized trust — every instruction has a fee
3. **Standards:** ERC-20, ERC-721, and ERC-1155 enable interoperable tokens across the ecosystem
4. **Security:** Immutable code means immutable bugs — auditing is not optional
5. **Composability:** Money legos enable rapid innovation AND create systemic risk

Decision Heuristic:

$$\text{Smart Contract Value} = \text{Automation Benefit} \,/\, (\text{Gas Cost} + \text{Bug Risk} + \text{Complexity})$$

If bug risk or complexity is high relative to automation benefit, use a traditional system instead.

These five takeaways map directly to the four tensions introduced at the start.

Summary and What Comes Next

Today's four tensions:

- Programmability vs complexity
- Immutability vs fixability
- Transparency vs exploitability
- Composability vs systemic risk

These are not bugs — they are fundamental design trade-offs that every Ethereum application must navigate.

Five questions to evaluate ANY smart contract:

1. Has it been audited by an independent security firm?
2. Is there an upgrade path (proxy pattern, governance)?
3. Who holds the admin keys — a multisig or a single person?
4. Does it depend on external oracles? How reliable are they?
5. What is the worst-case loss if the contract is exploited?

Coming next in the full course:

- **L13** — Ethereum Architecture
- **L14** — Gas Mechanics (deep dive)
- **L15** — Solidity Fundamentals
- **L17** — ERC-20 Token Standard
- **L18** — NFTs (ERC-721/1155)
- **L19** — Token Lifecycle
- **L43** — Smart Contract Security

Apply these five questions to the next DeFi protocol you encounter.

Quiz: Questions 1–5

Q1. Which of the following best defines a smart contract?
A) A legal agreement signed digitally B) A program stored on a blockchain that executes automatically when conditions are met
C) An encrypted message between two parties D) A database query that returns financial data

Quiz: Questions 1–5

Q1. Which of the following best defines a smart contract?
A) A legal agreement signed digitally B) A program stored on a blockchain that executes automatically when conditions are met
C) An encrypted message between two parties D) A database query that returns financial data
Answer: B – A smart contract is code on-chain that self-executes; it is not a legal document or encrypted message.

Q2. What distinguishes an Externally Owned Account (EOA) from a contract account on Ethereum?
A) EOAs store code; contract accounts do not
B) Contract accounts can initiate transactions; EOAs cannot
C) EOAs are controlled by private keys; contract accounts are controlled by code
D) There is no difference — both are identical

Quiz: Questions 1–5

Q1. Which of the following best defines a smart contract?
A) A legal agreement signed digitally B) A program stored on a blockchain that executes automatically when conditions are met
C) An encrypted message between two parties D) A database query that returns financial data
Answer: B – A smart contract is code on-chain that self-executes; it is not a legal document or encrypted message.

Q2. What distinguishes an Externally Owned Account (EOA) from a contract account on Ethereum?
A) EOAs store code; contract accounts do not
B) Contract accounts can initiate transactions; EOAs cannot
C) EOAs are controlled by private keys; contract accounts are controlled by code
D) There is no difference — both are identical
Answer: C – EOAs are controlled by humans via private keys; contract accounts execute their stored code when called.

Q3. What is the key difference between Bitcoin and Ethereum?
A) Bitcoin is decentralized; Ethereum is centralized
B) Bitcoin transfers value; Ethereum can also execute arbitrary programs
C) Ethereum is faster than Bitcoin in all cases
D) Bitcoin uses cryptography; Ethereum does not

Quiz: Questions 1–5

Q1. Which of the following best defines a smart contract?
A) A legal agreement signed digitally B) A program stored on a blockchain that executes automatically when conditions are met
C) An encrypted message between two parties D) A database query that returns financial data
Answer: B – A smart contract is code on-chain that self-executes; it is not a legal document or encrypted message.

Q2. What distinguishes an Externally Owned Account (EOA) from a contract account on Ethereum?
A) EOAs store code; contract accounts do not
B) Contract accounts can initiate transactions; EOAs cannot
C) EOAs are controlled by private keys; contract accounts are controlled by code
D) There is no difference — both are identical
Answer: C – EOAs are controlled by humans via private keys; contract accounts execute their stored code when called.

Q3. What is the key difference between Bitcoin and Ethereum?
A) Bitcoin is decentralized; Ethereum is centralized
B) Bitcoin transfers value; Ethereum can also execute arbitrary programs
C) Ethereum is faster than Bitcoin in all cases
D) Bitcoin uses cryptography; Ethereum does not
Answer: B – Ethereum added a Turing-complete virtual machine, enabling programmable logic beyond simple transfers.

Q4. A token is different from a coin because:
A) Tokens are more valuable than coins
B) Coins have their own blockchain; tokens live as smart contracts on another blockchain
C) Tokens can only be used for NFTs
D) There is no meaningful difference

Quiz: Questions 1–5

Q1. Which of the following best defines a smart contract?
A) A legal agreement signed digitally B) A program stored on a blockchain that executes automatically when conditions are met
C) An encrypted message between two parties D) A database query that returns financial data
Answer: B – A smart contract is code on-chain that self-executes; it is not a legal document or encrypted message.

Q2. What distinguishes an Externally Owned Account (EOA) from a contract account on Ethereum?
A) EOAs store code; contract accounts do not
B) Contract accounts can initiate transactions; EOAs cannot
C) EOAs are controlled by private keys; contract accounts are controlled by code
D) There is no difference — both are identical
Answer: C – EOAs are controlled by humans via private keys; contract accounts execute their stored code when called.

Q3. What is the key difference between Bitcoin and Ethereum?
A) Bitcoin is decentralized; Ethereum is centralized
B) Bitcoin transfers value; Ethereum can also execute arbitrary programs
C) Ethereum is faster than Bitcoin in all cases
D) Bitcoin uses cryptography; Ethereum does not
Answer: B – Ethereum added a Turing-complete virtual machine, enabling programmable logic beyond simple transfers.

Q4. A token is different from a coin because:
A) Tokens are more valuable than coins
B) Coins have their own blockchain; tokens live as smart contracts on another blockchain
C) Tokens can only be used for NFTs
D) There is no meaningful difference
Answer: B – BTC and ETH are coins (native to their chains); USDC and UNI are tokens (smart contracts on Ethereum).

Q5. A simple ETH transfer uses 21,000 gas. If the gas price is 30 gwei, what is the transaction fee?
A) 21,000 gwei B) 630,000 gwei C) 21,000 ETH D) 630,000 ETH

Quiz: Questions 1–5

Q1. Which of the following best defines a smart contract?
A) A legal agreement signed digitally B) A program stored on a blockchain that executes automatically when conditions are met
C) An encrypted message between two parties D) A database query that returns financial data
Answer: B – A smart contract is code on-chain that self-executes; it is not a legal document or encrypted message.

Q2. What distinguishes an Externally Owned Account (EOA) from a contract account on Ethereum?
A) EOAs store code; contract accounts do not
B) Contract accounts can initiate transactions; EOAs cannot
C) EOAs are controlled by private keys; contract accounts are controlled by code
D) There is no difference — both are identical
Answer: C – EOAs are controlled by humans via private keys; contract accounts execute their stored code when called.

Q3. What is the key difference between Bitcoin and Ethereum?
A) Bitcoin is decentralized; Ethereum is centralized
B) Bitcoin transfers value; Ethereum can also execute arbitrary programs
C) Ethereum is faster than Bitcoin in all cases
D) Bitcoin uses cryptography; Ethereum does not
Answer: B – Ethereum added a Turing-complete virtual machine, enabling programmable logic beyond simple transfers.

Q4. A token is different from a coin because:
A) Tokens are more valuable than coins
B) Coins have their own blockchain; tokens live as smart contracts on another blockchain
C) Tokens can only be used for NFTs
D) There is no meaningful difference
Answer: B – BTC and ETH are coins (native to their chains); USDC and UNI are tokens (smart contracts on Ethereum).

Q5. A simple ETH transfer uses 21,000 gas. If the gas price is 30 gwei, what is the transaction fee?
A) 21,000 gwei B) 630,000 gwei C) 21,000 ETH D) 630,000 ETH
Answer: B – Fee $= 21{,}000$ gas $\times$ 30 gwei/gas $= 630{,}000$ gwei $= 0.00063$ ETH.

Quiz: Questions 6–10

Q6. When Ethereum blocks are more than 50% full, what happens to the base fee under EIP-1559?
A) It stays the same B) It increases
C) It decreases D) It is set to zero

Quiz: Questions 6–10

Q6. When Ethereum blocks are more than 50% full, what happens to the base fee under EIP-1559?
A) It stays the same B) It increases
C) It decreases D) It is set to zero
Answer: B – EIP-1559 increases the base fee when blocks exceed 50% capacity, acting like a thermostat.

Q7. In the approve-transferFrom pattern, what must happen BEFORE a DEX can move Alice's tokens?
A) The DEX must pay Alice a fee
B) Alice must call approve() to grant the DEX permission
C) Alice must transfer tokens directly to the DEX first
D) The DEX must register with a regulatory authority

Quiz: Questions 6–10

Q6. When Ethereum blocks are more than 50% full, what happens to the base fee under EIP-1559?
A) It stays the same B) It increases
C) It decreases D) It is set to zero
Answer: B – EIP-1559 increases the base fee when blocks exceed 50% capacity, acting like a thermostat.

Q7. In the approve-transferFrom pattern, what must happen BEFORE a DEX can move Alice's tokens?
A) The DEX must pay Alice a fee
B) Alice must call approve() to grant the DEX permission
C) Alice must transfer tokens directly to the DEX first
D) The DEX must register with a regulatory authority
Answer: B – Without an explicit approve() call, the DEX has no permission to move Alice's tokens.

Q8. Alice has 100 tokens. She calls transfer(Bob, 40). What are their balances after?
A) Alice = 100, Bob = 40 B) Alice = 60, Bob = 40
C) Alice = 40, Bob = 60 D) Alice = 0, Bob = 100

Quiz: Questions 6–10

Q6. When Ethereum blocks are more than 50% full, what happens to the base fee under EIP-1559?
A) It stays the same B) It increases
C) It decreases D) It is set to zero
Answer: B – EIP-1559 increases the base fee when blocks exceed 50% capacity, acting like a thermostat.

Q7. In the approve-transferFrom pattern, what must happen BEFORE a DEX can move Alice's tokens?
A) The DEX must pay Alice a fee
B) Alice must call approve() to grant the DEX permission
C) Alice must transfer tokens directly to the DEX first
D) The DEX must register with a regulatory authority
Answer: B – Without an explicit approve() call, the DEX has no permission to move Alice's tokens.

Q8. Alice has 100 tokens. She calls transfer(Bob, 40). What are their balances after?
A) Alice = 100, Bob = 40 B) Alice = 60, Bob = 40
C) Alice = 40, Bob = 60 D) Alice = 0, Bob = 100
Answer: B – Transfer subtracts from sender (100 - 40 = 60) and adds to receiver (0 + 40 = 40).

Q9. A game needs both in-game currency (fungible) and unique weapons (non-fungible) in one contract. Which standard?
A) ERC-20 B) ERC-721 C) ERC-1155 D) ERC-777

Quiz: Questions 6–10

Q6. When Ethereum blocks are more than 50% full, what happens to the base fee under EIP-1559?
A) It stays the same B) It increases
C) It decreases D) It is set to zero
Answer: B – EIP-1559 increases the base fee when blocks exceed 50% capacity, acting like a thermostat.

Q7. In the approve-transferFrom pattern, what must happen BEFORE a DEX can move Alice's tokens?
A) The DEX must pay Alice a fee
B) Alice must call approve() to grant the DEX permission
C) Alice must transfer tokens directly to the DEX first
D) The DEX must register with a regulatory authority
Answer: B – Without an explicit approve() call, the DEX has no permission to move Alice's tokens.

Q8. Alice has 100 tokens. She calls transfer(Bob, 40). What are their balances after?
A) Alice = 100, Bob = 40 B) Alice = 60, Bob = 40
C) Alice = 40, Bob = 60 D) Alice = 0, Bob = 100
Answer: B – Transfer subtracts from sender (100 - 40 = 60) and adds to receiver (0 + 40 = 40).

Q9. A game needs both in-game currency (fungible) and unique weapons (non-fungible) in one contract. Which standard?
A) ERC-20 B) ERC-721 C) ERC-1155 D) ERC-777
Answer: C – ERC-1155 supports both fungible and non-fungible tokens in a single contract with batch operations.

Q10. Why does the SSTORE operation cost far more gas than the ADD operation?
A) SSTORE requires more complex math
B) SSTORE writes to persistent storage that every node must keep forever
C) ADD is a premium operation reserved for validators
D) There is no difference in gas cost

Quiz: Questions 6–10

Q6. When Ethereum blocks are more than 50% full, what happens to the base fee under EIP-1559?
A) It stays the same B) It increases
C) It decreases D) It is set to zero
Answer: B – EIP-1559 increases the base fee when blocks exceed 50% capacity, acting like a thermostat.

Q7. In the approve-transferFrom pattern, what must happen BEFORE a DEX can move Alice's tokens?
A) The DEX must pay Alice a fee
B) Alice must call approve() to grant the DEX permission
C) Alice must transfer tokens directly to the DEX first
D) The DEX must register with a regulatory authority
Answer: B – Without an explicit approve() call, the DEX has no permission to move Alice's tokens.

Q8. Alice has 100 tokens. She calls transfer(Bob, 40). What are their balances after?
A) Alice = 100, Bob = 40 B) Alice = 60, Bob = 40
C) Alice = 40, Bob = 60 D) Alice = 0, Bob = 100
Answer: B – Transfer subtracts from sender (100 - 40 = 60) and adds to receiver (0 + 40 = 40).

Q9. A game needs both in-game currency (fungible) and unique weapons (non-fungible) in one contract. Which standard?
A) ERC-20 B) ERC-721 C) ERC-1155 D) ERC-777
Answer: C – ERC-1155 supports both fungible and non-fungible tokens in a single contract with batch operations.

Q10. Why does the SSTORE operation cost far more gas than the ADD operation?
A) SSTORE requires more complex math
B) SSTORE writes to persistent storage that every node must keep forever
C) ADD is a premium operation reserved for validators
D) There is no difference in gas cost
Answer: B – Storage persists on all nodes indefinitely; arithmetic is temporary and discarded after execution.

Quiz: Questions 11–15

Q11. What is the correct order for deploying a smart contract?
A) Deploy, Write, Compile, Test B) Write, Compile, Deploy, Call
C) Compile, Write, Call, Deploy D) Call, Deploy, Compile, Write

Quiz: Questions 11–15

Q11. What is the correct order for deploying a smart contract?
A) Deploy, Write, Compile, Test B) Write, Compile, Deploy, Call
C) Compile, Write, Call, Deploy D) Call, Deploy, Compile, Write
Answer: B – Write source code, compile to bytecode, deploy to blockchain, then call functions at the contract address.

Q12. What does the ABI (Application Binary Interface) provide?
A) The private keys of the contract owner
B) A description of the contract's functions so external software can interact with it
C) The Ethereum network's IP addresses
D) A backup copy of the contract's source code

Quiz: Questions 11–15

Q11. What is the correct order for deploying a smart contract?
A) Deploy, Write, Compile, Test B) Write, Compile, Deploy, Call
C) Compile, Write, Call, Deploy D) Call, Deploy, Compile, Write
Answer: B – Write source code, compile to bytecode, deploy to blockchain, then call functions at the contract address.

Q12. What does the ABI (Application Binary Interface) provide?
A) The private keys of the contract owner
B) A description of the contract's functions so external software can interact with it
C) The Ethereum network's IP addresses
D) A backup copy of the contract's source code
Answer: B – The ABI is like a menu: it tells wallets and dApps what functions exist and how to call them.

Q13. Why does a reentrancy attack succeed?
A) The attacker breaks the cryptographic hash
B) The contract sends funds before updating its internal balance record
C) The attacker gains access to the contract owner's private key
D) The Ethereum network goes offline during the transaction

Quiz: Questions 11–15

Q11. What is the correct order for deploying a smart contract?
A) Deploy, Write, Compile, Test B) Write, Compile, Deploy, Call
C) Compile, Write, Call, Deploy D) Call, Deploy, Compile, Write
Answer: B – Write source code, compile to bytecode, deploy to blockchain, then call functions at the contract address.

Q12. What does the ABI (Application Binary Interface) provide?
A) The private keys of the contract owner
B) A description of the contract's functions so external software can interact with it
C) The Ethereum network's IP addresses
D) A backup copy of the contract's source code
Answer: B – The ABI is like a menu: it tells wallets and dApps what functions exist and how to call them.

Q13. Why does a reentrancy attack succeed?
A) The attacker breaks the cryptographic hash
B) The contract sends funds before updating its internal balance record
C) The attacker gains access to the contract owner's private key
D) The Ethereum network goes offline during the transaction
Answer: B – Reentrancy exploits the ordering error: send first, update later allows recursive withdrawal.

Q14. The Ethereum community hard-forked after The DAO hack. Which principle did this violate?
A) Transparency B) Decentralization
C) Immutability D) Interoperability

Quiz: Questions 11–15

Q11. What is the correct order for deploying a smart contract?
A) Deploy, Write, Compile, Test B) Write, Compile, Deploy, Call
C) Compile, Write, Call, Deploy D) Call, Deploy, Compile, Write
Answer: B – Write source code, compile to bytecode, deploy to blockchain, then call functions at the contract address.

Q12. What does the ABI (Application Binary Interface) provide?
A) The private keys of the contract owner
B) A description of the contract's functions so external software can interact with it
C) The Ethereum network's IP addresses
D) A backup copy of the contract's source code
Answer: B – The ABI is like a menu: it tells wallets and dApps what functions exist and how to call them.

Q13. Why does a reentrancy attack succeed?
A) The attacker breaks the cryptographic hash
B) The contract sends funds before updating its internal balance record
C) The attacker gains access to the contract owner's private key
D) The Ethereum network goes offline during the transaction
Answer: B – Reentrancy exploits the ordering error: send first, update later allows recursive withdrawal.

Q14. The Ethereum community hard-forked after The DAO hack. Which principle did this violate?
A) Transparency B) Decentralization
C) Immutability D) Interoperability
Answer: C – The fork reversed transactions on an "immutable" ledger, breaking the "code is law" principle.

Q15. DeFi composability means protocols can build on each other. What is the primary risk?
A) Higher transaction fees B) Cascading failures when one protocol breaks
C) Slower block times D) Increased electricity consumption

Quiz: Questions 11–15

Q11. What is the correct order for deploying a smart contract?
A) Deploy, Write, Compile, Test B) Write, Compile, Deploy, Call
C) Compile, Write, Call, Deploy D) Call, Deploy, Compile, Write
Answer: B – Write source code, compile to bytecode, deploy to blockchain, then call functions at the contract address.

Q12. What does the ABI (Application Binary Interface) provide?
A) The private keys of the contract owner
B) A description of the contract's functions so external software can interact with it
C) The Ethereum network's IP addresses
D) A backup copy of the contract's source code
Answer: B – The ABI is like a menu: it tells wallets and dApps what functions exist and how to call them.

Q13. Why does a reentrancy attack succeed?
A) The attacker breaks the cryptographic hash
B) The contract sends funds before updating its internal balance record
C) The attacker gains access to the contract owner's private key
D) The Ethereum network goes offline during the transaction
Answer: B – Reentrancy exploits the ordering error: send first, update later allows recursive withdrawal.

Q14. The Ethereum community hard-forked after The DAO hack. Which principle did this violate?
A) Transparency B) Decentralization
C) Immutability D) Interoperability
Answer: C – The fork reversed transactions on an "immutable" ledger, breaking the "code is law" principle.

Q15. DeFi composability means protocols can build on each other. What is the primary risk?
A) Higher transaction fees B) Cascading failures when one protocol breaks
C) Slower block times D) Increased electricity consumption
Answer: B – Composability creates dependencies; a bug in one layer can cascade through the entire stack.

Quiz: Questions 16–20

Q16. Bitcoin uses UTXO (unspent transaction outputs); Ethereum uses an account model. What trade-off does this represent?
A) UTXO is faster; accounts are slower B) UTXO enables better privacy but is harder to program; accounts are easier to program but less private
C) There is no practical difference D) Accounts use less storage than UTXO

Quiz: Questions 16–20

Q16. Bitcoin uses UTXO (unspent transaction outputs); Ethereum uses an account model. What trade-off does this represent?
A) UTXO is faster; accounts are slower B) UTXO enables better privacy but is harder to program; accounts are easier to program but less private
C) There is no practical difference D) Accounts use less storage than UTXO
Answer: B – UTXO tracks individual coins (better privacy); accounts track balances (easier for smart contracts).

Q17. A developer discovers a critical bug in their deployed contract. What can they do?
A) Edit the contract code directly on-chain
B) Nothing — immutable code cannot be changed; they can only deploy a new version and migrate users
C) Ask Ethereum validators to patch the contract
D) Revert the blockchain to before deployment

Quiz: Questions 16–20

Q16. Bitcoin uses UTXO (unspent transaction outputs); Ethereum uses an account model. What trade-off does this represent?
A) UTXO is faster; accounts are slower B) UTXO enables better privacy but is harder to program; accounts are easier to program but less private
C) There is no practical difference D) Accounts use less storage than UTXO
Answer: B – UTXO tracks individual coins (better privacy); accounts track balances (easier for smart contracts).

Q17. A developer discovers a critical bug in their deployed contract. What can they do?
A) Edit the contract code directly on-chain
B) Nothing — immutable code cannot be changed; they can only deploy a new version and migrate users
C) Ask Ethereum validators to patch the contract
D) Revert the blockchain to before deployment
Answer: B – Deployed contracts are immutable. The only option is deploying a new contract and migrating state.

Q18. DeFi protocols hold over $120B in total value locked. What does this number represent?
A) The total revenue earned by DeFi developers
B) The value of assets deposited into smart contracts that could be lost if those contracts fail
C) The market cap of all Ethereum tokens
D) The amount of electricity consumed by DeFi

Quiz: Questions 16–20

Q16. Bitcoin uses UTXO (unspent transaction outputs); Ethereum uses an account model. What trade-off does this represent?
A) UTXO is faster; accounts are slower B) UTXO enables better privacy but is harder to program; accounts are easier to program but less private
C) There is no practical difference D) Accounts use less storage than UTXO
Answer: B – UTXO tracks individual coins (better privacy); accounts track balances (easier for smart contracts).

Q17. A developer discovers a critical bug in their deployed contract. What can they do?
A) Edit the contract code directly on-chain
B) Nothing — immutable code cannot be changed; they can only deploy a new version and migrate users
C) Ask Ethereum validators to patch the contract
D) Revert the blockchain to before deployment
Answer: B – Deployed contracts are immutable. The only option is deploying a new contract and migrating state.

Q18. DeFi protocols hold over $120B in total value locked. What does this number represent?
A) The total revenue earned by DeFi developers
B) The value of assets deposited into smart contracts that could be lost if those contracts fail
C) The market cap of all Ethereum tokens
D) The amount of electricity consumed by DeFi
Answer: B – TVL measures assets at risk in smart contracts — it represents both opportunity and exposure.

Q19. A freelancer wants to use a smart contract for escrow: client deposits payment, freelancer delivers work, payment releases. Is this appropriate?
A) No — smart contracts cannot handle any financial transactions
B) Yes — but only if "delivery" can be verified on-chain (e.g., by an oracle or mutual confirmation)
C) Yes — the contract can judge work quality automatically
D) No — escrow always requires a human intermediary

Quiz: Questions 16–20

Q16. Bitcoin uses UTXO (unspent transaction outputs); Ethereum uses an account model. What trade-off does this represent?
A) UTXO is faster; accounts are slower B) UTXO enables better privacy but is harder to program; accounts are easier to program but less private
C) There is no practical difference D) Accounts use less storage than UTXO
Answer: B – UTXO tracks individual coins (better privacy); accounts track balances (easier for smart contracts).

Q17. A developer discovers a critical bug in their deployed contract. What can they do?
A) Edit the contract code directly on-chain
B) Nothing — immutable code cannot be changed; they can only deploy a new version and migrate users
C) Ask Ethereum validators to patch the contract
D) Revert the blockchain to before deployment
Answer: B – Deployed contracts are immutable. The only option is deploying a new contract and migrating state.

Q18. DeFi protocols hold over $120B in total value locked. What does this number represent?
A) The total revenue earned by DeFi developers
B) The value of assets deposited into smart contracts that could be lost if those contracts fail
C) The market cap of all Ethereum tokens
D) The amount of electricity consumed by DeFi
Answer: B – TVL measures assets at risk in smart contracts — it represents both opportunity and exposure.

Q19. A freelancer wants to use a smart contract for escrow: client deposits payment, freelancer delivers work, payment releases. Is this appropriate?
A) No — smart contracts cannot handle any financial transactions
B) Yes — but only if "delivery" can be verified on-chain (e.g., by an oracle or mutual confirmation)
C) Yes — the contract can judge work quality automatically
D) No — escrow always requires a human intermediary
Answer: B – Smart contracts excel at escrow but cannot judge subjective quality; an oracle or mutual sign-off is needed.

Q20. You are evaluating a new DeFi protocol. It has not been audited, a single developer holds the admin keys, and it depends on one price oracle. Should you deposit funds?
A) Yes — high risk means high reward
B) No — it fails three of the five evaluation criteria (no audit, single admin, single oracle)
C) Yes — if the TVL is above $1 billion, it must be safe
D) It depends only on the expected interest rate

Quiz: Questions 16–20

Q16. Bitcoin uses UTXO (unspent transaction outputs); Ethereum uses an account model. What trade-off does this represent?
A) UTXO is faster; accounts are slower B) UTXO enables better privacy but is harder to program; accounts are easier to program but less private
C) There is no practical difference D) Accounts use less storage than UTXO
Answer: B – UTXO tracks individual coins (better privacy); accounts track balances (easier for smart contracts).

Q17. A developer discovers a critical bug in their deployed contract. What can they do?
A) Edit the contract code directly on-chain
B) Nothing — immutable code cannot be changed; they can only deploy a new version and migrate users
C) Ask Ethereum validators to patch the contract
D) Revert the blockchain to before deployment
Answer: B – Deployed contracts are immutable. The only option is deploying a new contract and migrating state.

Q18. DeFi protocols hold over $120B in total value locked. What does this number represent?
A) The total revenue earned by DeFi developers
B) The value of assets deposited into smart contracts that could be lost if those contracts fail
C) The market cap of all Ethereum tokens
D) The amount of electricity consumed by DeFi
Answer: B – TVL measures assets at risk in smart contracts — it represents both opportunity and exposure.

Q19. A freelancer wants to use a smart contract for escrow: client deposits payment, freelancer delivers work, payment releases. Is this appropriate?
A) No — smart contracts cannot handle any financial transactions
B) Yes — but only if "delivery" can be verified on-chain (e.g., by an oracle or mutual confirmation)
C) Yes — the contract can judge work quality automatically
D) No — escrow always requires a human intermediary
Answer: B – Smart contracts excel at escrow but cannot judge subjective quality; an oracle or mutual sign-off is needed.

Q20. You are evaluating a new DeFi protocol. It has not been audited, a single developer holds the admin keys, and it depends on one price oracle. Should you deposit funds?
A) Yes — high risk means high reward
B) No — it fails three of the five evaluation criteria (no audit, single admin, single oracle)
C) Yes — if the TVL is above $1 billion, it must be safe
D) It depends only on the expected interest rate
Answer: B – No audit, centralized admin keys, and a single oracle dependency are three critical red flags.

Glossary (A–D)

ABI — Application Binary Interface; a JSON description of a contract's functions that lets wallets and dApps call it.
approve — ERC-20 function granting a third party (e.g. a DEX) permission to spend up to a specified number of your tokens.
bytecode — Low-level machine instructions compiled from Solidity that the EVM executes on every node.
composability — The ability of DeFi protocols to interact like building blocks—enabling powerful combinations but also cascading risk.

DAO — Decentralized Autonomous Organization; a smart-contract-governed entity where token holders vote on decisions.
DeFi — Decentralized Finance; financial services (lending, trading, insurance) built on smart contracts without traditional intermediaries.
DEX — Decentralized Exchange; a protocol enabling peer-to-peer token swaps via smart contracts (e.g. Uniswap).

Glossary (E)

EIP-1559 — Fee reform introducing a burned base fee that adjusts with demand, plus an optional tip to validators.
EOA — Externally Owned Account; a human-controlled Ethereum address managed by a private key (vs. a contract account).
ERC-20 — Standard interface for fungible tokens, defining functions like `transfer`, `approve`, and `transferFrom`.
ERC-721 — Standard for non-fungible tokens; each token has a unique ID and distinct ownership.
ERC-1155 — Multi-token standard supporting both fungible and non-fungible tokens in one contract with batch operations.
EVM — Ethereum Virtual Machine; the sandboxed runtime executing smart-contract bytecode identically on every node.

Glossary (F–N)

flash loan — An uncollateralized loan borrowed and repaid within one transaction; used in arbitrage and exploits.
fungible — Interchangeable; each unit is identical in value (e.g. one ETH equals any other ETH).
gas — Unit measuring computational effort of EVM operations; users pay gas fees to compensate validators.
gwei — Denomination of ETH equal to 10^9 wei; commonly used for gas prices.

hard fork — A backward-incompatible protocol upgrade; used after The DAO hack to reverse stolen funds, creating Ethereum Classic.
MEV — Maximal Extractable Value; profit validators or searchers capture by reordering, inserting, or censoring transactions.
non-fungible — Unique and not interchangeable; each token represents a distinct asset (e.g. a specific digital artwork).

Glossary (O–U)

oracle — Service feeding off-chain data (prices, weather) into smart contracts, which cannot access external data natively.
proxy — Upgradeable contract pattern where a proxy delegates calls to a replaceable implementation contract.
reentrancy — Vulnerability where an external call re-enters the caller before state updates complete, enabling repeated withdrawals.
smart contract — A program stored on a blockchain that executes automatically when predefined conditions are met.
Solidity — Primary high-level language for writing EVM-compatible smart contracts.
transferFrom — ERC-20 function allowing an approved spender to move tokens on behalf of the owner.
TVL — Total Value Locked; the aggregate value of crypto assets deposited in DeFi smart contracts.
UTXO — Unspent Transaction Output; Bitcoin's accounting model where each "coin" is an unspent output from a prior transaction.