Ethereum & Smart Contracts

Pre-Class Discovery Handout

Lesson 03 · Complete before class · 25–30 minutes

Activity 1: Explore Ethereum

10 min

Visit `https://etherscan.io` and find any smart contract. Answer:

1. What is the contract's address?
2. How many transactions has it processed?
3. What is the contract's ETH balance?
4. Can you identify what the contract does from its name or verified source code?

Bonus: Find a transaction to this contract. What was the gas used? What was the gas price in Gwei?

Activity 2: Gas Price Calculator

5 min

Calculate the transaction fee for each scenario:

Operation	Gas Used	Gas Price (Gwei)	Fee (ETH)
ETH Transfer	21,000	30	_____
ERC-20 Transfer	65,000	30	_____
Uniswap Swap	150,000	50	_____
NFT Mint	250,000	50	_____
Contract Deploy	1,500,000	20	_____

Hint: Fee = Gas Used × Gas Price. 1 Gwei = 10^{-9} ETH.

Activity 3: Read a Smart Contract

10 min

Read this Solidity contract and answer the questions:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract SimpleVault {
   mapping(address => uint256) public balances;
   function deposit() public payable {
      balances[msg.sender] += msg.value;
   }
   function withdraw(uint256 amount) public {
      require(balances[msg.sender] >= amount);
      balances[msg.sender] -= amount;
      payable(msg.sender).transfer(amount);
   }
}
```

1. What is the state variable and its type?
2. What does `msg.sender` represent?
3. What does `msg.value` represent?
4. What happens if you try to withdraw more than your balance?
5. Can you spot a potential security vulnerability? (Hint: think about the order of operations)

Activity 4: Smart Contract vs Traditional App

5 min

Compare a traditional ride-sharing app (Uber) with a decentralized version (DAO-based):

Feature	Uber (Centralized)	Ride DAO (Decentralized)
Who sets prices?	Company	_____
Payment processing	Credit card + Uber	_____
Driver verification	Background check	_____
Revenue distribution	Uber takes 25%	_____
Rule changes	Uber decides	_____

Key Terms

Term	Definition
EVM	Ethereum Virtual Machine. The sandboxed runtime environment that executes smart contract bytecode on every Ethereum node.
Gas	The unit measuring computational effort required to execute operations on the EVM. Every instruction costs a defined amount of gas.
Wei	The smallest denomination of ether. $1\,\text{ETH} = 10^{18}\,\text{Wei}$. Used for precise on-chain arithmetic.
Gwei	Gigawei. $1\,\text{Gwei} = 10^{9}\,\text{Wei}$. The conventional unit for expressing gas prices.
Solidity	The most widely used high-level programming language for writing Ethereum smart contracts, syntactically similar to JavaScript/C++.
ABI	Application Binary Interface. A JSON specification describing a contract's functions and events, enabling external callers to encode and decode calls.
Bytecode	The low-level EVM instructions produced by compiling Solidity source code. This is what is actually stored on-chain and executed.
EOA	Externally Owned Account. An Ethereum account controlled by a private key (a human wallet), as opposed to a contract account.
Smart Contract	Self-executing code stored on the blockchain that automatically enforces agreement terms without a trusted intermediary.
Nonce	A per-account transaction counter. Each new transaction must increment the nonce, preventing replay attacks and ordering transactions.