Selling a token

From a price tag to automated market makers · Session 13 · 15 October 2026

Where we left off

- **ERC-20** is an agreed interface: any wallet, explorer or contract can handle any token.
- On Monday you created **12 tokens** from addresses in the class sheet.
- Two of them are held by **every** other address in the sheet.

Suppose your token is useful: a café accepts it. Can you **sell** it, for ether?

The simplest shop

A contract that holds your tokens and hands them out for ether, at a price you set.

```
contract TokenSale {
    IERC20 public immutable token;
    address public immutable seller;
    uint256 public price;                  // wei for one whole token

    constructor(IERC20 _token, uint256 _price) {
        token = _token; seller = msg.sender; price = _price;
    }
    function buy() external payable {
        uint256 amount = msg.value * 1e18 / price;
        require(token.transfer(msg.sender, amount), "transfer failed");
    }
    function withdraw() external { … }     // seller only: ether and unsold tokens
}
```

Challenge: buy from the shop

In class: a shop selling **20 CLG**, at 0.001 ETH each.

1. Open the shop on Blockscout: 0x73fA…5fB9, *Write contract*, connect.
2. Call `buy()`, with the ether you pay in *Value*.
3. The price is in **wei** per whole token: 0.001 ETH = `1000000000000000` wei = 10^{15}.

To open a shop of your own: the full code, ready to copy, is at cryptoassets2026.bruschi.dev/13/token-sale.

What goes wrong with a fixed price

What happens	Consequence
The price is too low	the first buyer takes everything
The price is too high	nobody buys
The café starts accepting the token	demand grows, the price does not move
A buyer wants to sell back	the shop only sells

Every correction is a transaction by the seller, who must keep **watching**.

Splitting the sale

Sell the same 400 tokens in four **orders**, at increasing prices:

Order	Tokens	Price (ETH per token)
1	100	0.0011
2	100	0.0014
3	100	0.0018
4	100	0.0024

The first buyers pay less, later ones more: the price **rises with demand**.

An order book

Buyers post orders too. The **book** lists them all, best prices first:

Bids: will buy	Price	Asks: will sell
	0.0014	100
	0.0011	100
150	0.0010	
300	0.0008	

- The gap between best ask and best bid is the **spread**.
- A trade happens when a new order **crosses** the book: a buy at 0.0011 or more.

Live: the ETH order book on Hyperliquid

Reading a live book

Price ($)	Size (ETH)	Total (ETH)
2383.0	267.61	679.10
2382.9	82.76	411.49
2382.8	89.52	328.73
2382.7	69.85	239.21
2382.6	91.60	169.36
2382.5	22.85	77.76
2382.4	8.53	54.92
2382.3	46.39	46.39
spread 0.1		
2382.2	149.56	149.56
2382.1	52.17	201.72
2382.0	89.22	290.94
2381.9	158.49	449.43
2381.8	246.30	695.73

A market buy of 500 ETH

takes every ask from 2,382.3 to 2,382.9, 411.5 ETH, then 88.5 of the 267.6 ETH offered at 2,383.0.

Average price **2,382.73**: 0.02% above the mid price, 2,382.25.

the part the buy takes

— best ask − best bid: 0.1 on 2,382, **0.004%**

Bars: the running total

how much you can buy, or sell, without going past that price

Who keeps the book alive

A **market maker** keeps orders on both sides, and earns the spread.

- When news arrives, every resting order is at the **old** price.
- Whoever learns first buys the cheap asks, or sells to the high bids, at the market maker’s expense.
- So orders must be **updated constantly**: cancelled and placed again, all day.

An order book on a blockchain

Every order, change and cancellation is a **transaction**:

- it pays a **fee**, even if it never trades;
- it waits for a **block**, about 12 seconds on Ethereum;
- it is **visible** before it is included, and others can act first.

EtherDelta (2016) settled every trade on Ethereum, and kept the signed orders on its own server. Most order books today run **off-chain**, or on chains built for speed.

EtherDelta on Blockscout

Why EtherDelta faded

- **Off-chain weak points**: the order server lagged or went down at peak times. In December 2017 a DNS hijack put up a fake site that stole private keys.
- **Gas for every step**: deposit, trade, withdraw. Two takers on one order: one fails, and still pays. Pending trades could be front-run.
- **Ghost orders**: listed in the book, no longer funded.
- **The law**: in November 2018 the SEC found its founder had run an **unregistered securities exchange**.
- **Better designs**: 0x for signed orders; from 2018, AMMs, with no one online to quote.

A market that sets its own price

What if the contract held **both** assets, and quoted a price from what it holds?

- Nobody has to post or update orders.
- After every trade the price **moves by itself**.
- Anyone can trade with it, at any time, for any amount.

An **automated market maker** (AMM): the market maker is a contract.

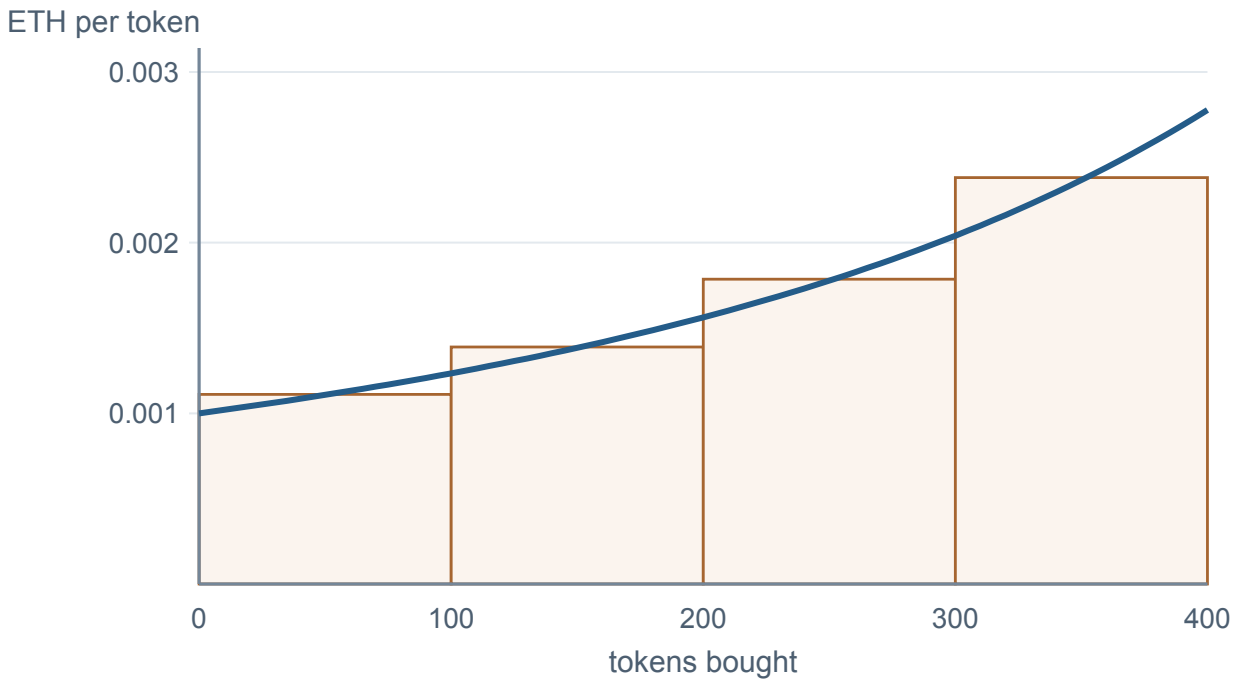
Constant product: x · y = k

A **pool** holds 1,000 tokens (x) and 1 ETH (y): k = 1,000. Price = y / x = 0.001.

A trade must leave **x · y unchanged**. Buy tokens 100 at a time:

Tokens left	ETH in the pool	You pay	Price after
1,000	1.000		0.00100
900	1.111	0.111	0.00123
800	1.250	0.139	0.00156
700	1.429	0.179	0.00204
600	1.667	0.238	0.00278

An order book that updates itself



Order book
four sell orders,
100 tokens each,
updated by hand

Pool
1,000 tokens + 1 ETH,
$x \cdot y = 1{,}000$: the price
moves after every trade

The pool is a staircase with infinitely many steps, and no one has to maintain it.

x · y = k, live

The Uniswap V2 pool between **USDC** and **WETH** (ether as an ERC-20), on Ethereum:

1. *Contract* → *Read* → `getReserves()`: `reserve0` is USDC (6 decimals), `reserve1` is WETH (18 decimals).
2. Price of one ETH = (reserve0 / 10^{6}) ÷ (reserve1 / 10^{18}).
3. Compare with the order book on Hyperliquid: arbitrage keeps them close.
4. *Logs*: every trade leaves a `Swap` and a `Sync` event, and two `Transfer`s.

   The USDC/WETH pool on Blockscout

Uniswap

Version	Year	What it added
V1	2018	ETH–token pools, $x \cdot y = k$
V2	2020	pools between **any** two ERC-20 tokens
V3	2021	liquidity concentrated in a chosen **price range**
V4	2025	all pools in one contract; **hooks**: custom code in a pool

A few hundred lines of public code, and among the largest exchanges in crypto.

Live: Uniswap Explore · DefiLlama: Uniswap

A pool for a new token

A token made for the pool, in the OpenZeppelin Wizard: **MisteriousToken** (MIS).

- **0 decimals**: it moves only in whole units. **100** in all, none can be minted later.
- *Votes*: we will use it to vote.

On Uniswap, a **V4** pool MIS/ETH, with all 100 MIS and 0.1 ETH: a starting price of 0.001 ETH per MIS.

MIS on Blockscout

Challenge: buy MIS

1. **app.uniswap.org**, connect MetaMask; Settings → **Testnet mode**, network Sepolia.
2. *Swap*: ETH → MIS. Paste its **contract address**:
   `0x43855fEfAbd5D73754d32404A5a34e54EDC895f8`.
3. The interface warns that the token is unknown: anyone can create a token with any name.
4. Swap. A failed attempt counts too.

   There are only 100 MIS: every purchase raises the price for the next buyer.

Challenge: a pool for your token

On Uniswap: *Positions* → *New position*.

1. Pair: **ETH** and the token you created on Monday.
2. Fee tier, and the **starting price**: you set it, by the amounts you deposit.
3. Range: **full range**, as in V2.
4. Deposit: you are the pool's first liquidity provider.

Then anyone can buy and sell your token, at any time, with no one keeping a book.

Next session

- What happened in our shop, and in the MIS pool.
- Large trades move the price: **price impact** and **slippage**.
- Where the reserves come from: **liquidity providers**, fees, arbitrage.
- Why a pool needs an **approval** before it can take your tokens.
- SushiSwap copies Uniswap, Uniswap answers with a token: **UNI**, and **governance**.