A12: Token Builder

Create Your Own ERC-20 Token

Prof. Joerg Osterrieder

Spring 2026

Assignment Overview

Learning Objectives

- Write a complete ERC-20 token in Solidity
- Deploy and test using Remix IDE
- Extend with a custom feature (burn, mint, or pause)
- Present your token and design decisions

Assignment Details

- Time: 60 minutes
- Format: Groups of 2–3 students
- Difficulty: Medium
- Points: 50 (+10 bonus)

Grading Breakdown

- Working contract: 25 pts
- Custom feature: 15 pts
- Presentation: 10 pts
- Bonus (2+ features): +10 pts

A12:

digital-ai-finance.github.io/crypto-economics/assignments/A12_token_builder/instructions.html

What is ERC-20?

The Token Standard

- ERC-20 = Ethereum Request for Comments #20
- Defines a **common interface** for fungible tokens
- Any wallet, DEX, or dApp that supports ERC-20 can work with *any* ERC-20 token
- Over 500,000 ERC-20 tokens deployed on Ethereum

Required Functions

- `totalSupply()`
- `balanceOf(address)`
- `transfer(to, amount)`
- `approve(spender, amount)`
- `transferFrom(from, to, amount)`
- `allowance(owner, spender)`

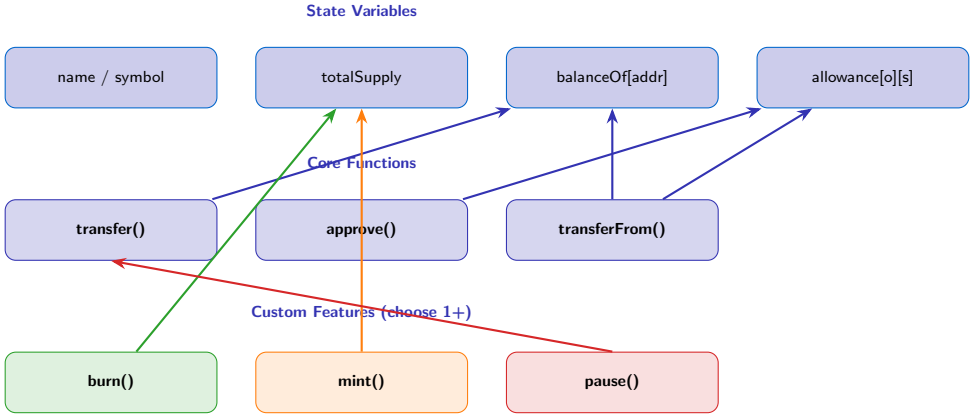
Required Events

- `Transfer(from, to, value)`
- `Approval(owner, spender, value)`

A12:

digital-ai-finance.github.io/crypto-economics/assignments/A12_token_builder/instructions.html

ERC-20 Token Architecture



How It Fits Together

Core functions read and write `balanceOf` and `allowance` mappings. Custom features extend the base token: burn reduces supply, mint increases it, pause blocks transfers.

Step 1: Setup & Starter Code

5 min

Steps

1. Open remix.ethereum.org
2. Create a new file: `MyToken.sol`
3. Copy the starter code from the assignment page
4. Read through the TODO comments

Starter Code Structure

- State variables: **provided**
- Events: **provided**
- Constructor: **TODO 1**
- transfer(): **TODO 2**
- approve(): **TODO 3**
- transferFrom(): **TODO 4**
- Custom feature: **TODO 5**

Remix Quick Guide

- **Compile:** Ctrl+S or Compiler tab
- **Deploy:** Deploy & Run tab
- **Environment:** Remix VM (Shanghai)
- **Test accounts:** 15 pre-funded accounts in dropdown

Key Tip

Compile after every function. Fix errors incrementally – don't write everything then compile.

A12:

digital-ai-finance.github.io/crypto-economics/assignments/A12_token_builder/instructions.html

Step 2: Build the Base Token

25 min

TODO 1: Constructor (4 lines)

1. Set `name` from parameter
2. Set `symbol` from parameter
3. Set `owner = msg.sender`
4. Set `totalSupply` and `balanceOf[msg.sender]`

TODO 2: transfer() (5 lines)

1. `require(balance >= amount)`
2. Subtract from sender
3. Add to recipient
4. `emit Transfer(...)`
5. `return true`

TODO 3: approve() (3 lines)

1. Set allowance mapping
2. `emit Approval(...)`
3. `return true`

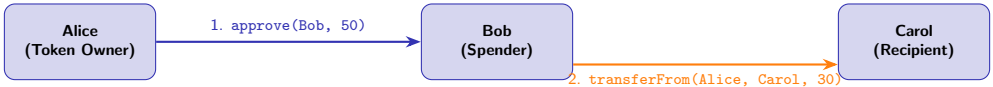
TODO 4: transferFrom() (7 lines)

1. Check balance of `_from`
2. Check allowance of caller
3. Update balances
4. **Decrease allowance** (most missed!)
5. Emit event + return

A12:

digital-ai-finance.github.io/crypto-economics/assignments/A12_token_builder/instructions.html

Understanding Approve + TransferFrom



Why This Matters

DEXes use this pattern: you `approve()` the DEX contract, then the DEX calls `transferFrom()` to move your tokens during a swap. Without this, smart contracts could never move tokens on your behalf.

A12:

digital-ai-finance.github.io/crypto-economics/assignments/A12_token_builder/instructions.html

Step 3: Deploy & Test

10 min

Deployment Checklist

1. Go to Deploy & Run Transactions panel
2. Environment: **Remix VM (Shanghai)**
3. Enter constructor args: name, symbol, initial supply (e.g., 1000000)
4. Click **Deploy**
5. Expand the deployed contract in the bottom panel

Test Checklist

- ☐ `balanceOf(deployer)` = total supply
- ☐ `transfer(acct2, 100)` succeeds
- ☐ Balances update correctly
- ☐ `transfer` with too much reverts
- ☐ `approve(acct2, 50)` works
- ☐ `transferFrom` from acct2 works
- ☐ Allowance decreases after transferFrom

Debugging Tip

If a transaction reverts, check the Remix console at the bottom for the error message. Common causes: wrong account selected, insufficient balance, or a typo in the require message string.

A12:

digital-ai-finance.github.io/crypto-economics/assignments/A12_token_builder/instructions.html

Step 4: Add a Custom Feature

15 min

Burn (Easiest)

- Destroys caller's tokens
- Reduces `totalSupply`
- No access control needed
- Real example: BNB quarterly burns

3 lines of logic + event

Mint (Medium)

- Creates new tokens
- Increases `totalSupply`
- **Must be owner-only!**
- Real example: USDT minting

4 lines of logic + event

Pause (Hardest)

- Emergency stop for transfers
- Uses a modifier
- Owner toggles on/off
- Real example: USDT pause

Modifier + 2 functions

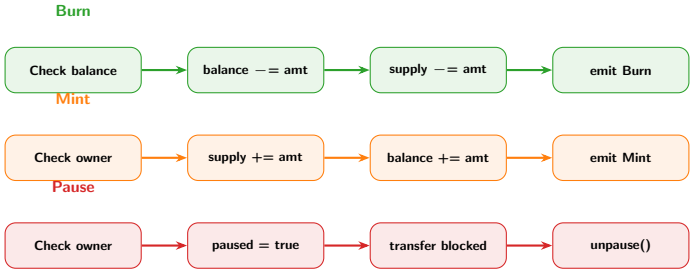
Bonus: +10 Points

Implement **two or more** features for bonus points (+5 per additional feature, max +10).

A12:

digital-ai-finance.github.io/crypto-economics/assignments/A12_token_builder/instructions.html

Custom Feature Flows



Key Insight: Access Control

Burn: No owner check needed – users burn their own tokens.
Mint & Pause: Must use `require(msg.sender == owner)` – without it, anyone could create tokens or freeze the contract.

A12:

digital-ai-finance.github.io/crypto-economics/assignments/A12_token_builder/instructions.html

Step 5: Presentation

5 min total (2 min/group)

Presentation Structure

1. **Token Identity:** Name, symbol, initial supply
2. **Feature Demo:** Show your custom feature working in Remix
3. **Design Decision:** Why did you choose this feature?
4. **Challenge:** What was the hardest part?

Deliverables

- ☐ Working ERC-20 contract (25 pts)
- ☐ Custom feature (15 pts)
- ☐ 2-minute presentation (10 pts)
- ☐ Written reflection (3–5 sentences)

Real-World Examples

- USDT: mint + burn + pause
- UNI: fixed supply (no mint)
- BNB: quarterly burn events

A12:

digital-ai-finance.github.io/crypto-economics/assignments/A12_token_builder/instructions.html

Get Started

Assignment Page

digital-ai-finance.github.io/crypto-economics/assignments/A12_token_builder/instructions.html

Starter Code

digital-ai-finance.github.io/crypto-economics/assignments/A12_token_builder/starter_code.html

All Assignments

digital-ai-finance.github.io/crypto-economics/assignments/index.html

Open Remix and start building your token!

You have 60 minutes: 5 min setup, 25 min build, 10 min test, 15 min feature, 5 min present.