ERC-20 Basics (Ethereum Tokens)

What ERC-20 is, what it requires, and why it matters

Introduction (Super Simple Idea)

Imagine you have a **game coin** or **points** that everyone can trade. If the coin rules are the same everywhere, then any game shop, trading booth, or wallet can understand it. **ERC-20** is like the **rulebook** that makes many Ethereum tokens work the same way, so apps and wallets can use them without needing special code for each token.

1. What is a Token? (In Kid-Friendly Words)

A **token** is a digital thing on Ethereum that can represent almost anything, like:

- points or reputation in an online community
- skills or items in a game
- money-like things (like a digital USD)
- parts of a company (like a share)
- an asset idea (like "1 ounce of gold" as a digital claim)

Important: A token is controlled by a **smart contract** (a small program on Ethereum) that keeps track of who owns how many tokens.

2. What does "ERC" mean?

ERC stands for **Ethereum Request for Comments**. You can think of it as:

- **A community-approved standard (a shared rule)**
- It tells developers: "If you build this kind of thing, follow these rules so everyone can understand it."

ERC-20 is one specific standard: it describes how a common type of token should behave.

3. What is ERC-20?

ERC-20 is the most common standard for **fungible tokens** on Ethereum.
Fungible means: **every unit is the same.**

- If you have 1 coin and I have 1 coin, they are equal.
- Like regular money: any $10 bill is worth $10.

So for an ERC-20 token:

- 1 token is always equal (in type) to any other 1 token of the same token.

Exam Check

Fungible means "all pieces are the same." Example: 1 ETH is the same as any other 1 ETH.

4. Why do we need ERC-20? (Why it matters)

ERC-20 is important because it makes tokens **interoperable** (easy to use together).
In simple words: If lots of tokens follow the same rules, then:

- wallets can show your token balances
- exchanges can swap tokens
- apps can accept tokens

They can do this **without writing special code for every single token.**

Quick Background

ERC-20 stands for **Ethereum Request for Comments 20**. It was proposed in **November 2015**.

5. What is REQUIRED for an ERC-20 token? (The rulebook)

An ERC-20 token is a **smart contract** that includes a standard set of functions (methods) and events.
Think of functions as buttons that apps can press. **Think of events as receipts** that the blockchain records.
If a token contract has the standard functions and events, many apps can talk to it in the same way.

6. The ERC-20 Functions (Explained for Beginners)

Here are the most important ERC-20 functions (what they do in plain English):

- `name()` → the token's full name (example: "Dai Stablecoin")
- `symbol()` → short name (example: "DAI")
- `decimals()` → how many tiny pieces the token can be divided into
- `totalSupply()` → how many tokens exist in total
- `balanceOf(address)` → how many tokens a person/account has
- `transfer(to, value)` → send tokens from **you** to someone else
- `approve(spender, value)` → allow another account/app to spend up to some amount
- `allowance(owner, spender)` → check how much spending is allowed
- `transferFrom(from, to, value)` → a spender sends tokens on your behalf (if approved)

Big idea: ERC-20 supports both **direct sending** and **permission-based spending**.

7. The “Approve + TransferFrom” Pattern (Simple Example)

Sometimes you want an app (like a trading app) to move tokens for you.
A common safe pattern is:

- Step 1: You call `approve(app, amount)`
  (“I allow this app to spend up to 10 tokens.”)
- Step 2: The app later calls `transferFrom(you, someone, amount)`
  (“The app spends those tokens according to the permission you gave.”)

Why use this? Many contracts expect this pattern for deposits and swaps.

8. ERC-20 Events (The Blockchain “Receipts”)

ERC-20 includes two main events:

- `Transfer(from, to, value)`
  Recorded whenever tokens move from one account to another.
- `Approval(owner, spender, value)`
  Recorded whenever an owner gives spending permission to a spender.

Events help wallets and apps track what happened on-chain.

Exam Check

Transfer event = a receipt for moving tokens.
Approval event = a receipt for giving spending permission.

9. Known Issue (Important): Tokens can get stuck in smart contracts

Problem (plain English): If you send ERC-20 tokens to a smart contract that is **not built to handle them**, those tokens can be **stuck forever**.
Why does this happen?

- ERC-20 uses `transfer` or `transferFrom` to move tokens.
- The token will move even if the receiving contract is “not paying attention.”
- ERC-20 does **not require** a special “I received tokens” callback message.
- So a contract may not know what to do with incoming tokens, and no one can easily get them back.

Simple picture: It’s like mailing a package to a building with **no mailbox**. The package arrives, but nobody can pick it up.

10. How to Reduce Accidental Token Loss (Practical Safety Tips)

You can't fully remove this risk for ERC-20, but you can reduce it:

- **Double-check addresses.** A common mistake is sending tokens to the **token contract address itself**.
- Developers sometimes add a rule to block sending tokens to the token contract itself, like: `require(_to != address(this));`
- For deposits into apps, many systems prefer the **approve + transferFrom** pattern.
- If you write your own contract, assume tokens can arrive by accident and add a **recovery function** to withdraw stuck tokens (if appropriate).
- Some newer standards (examples: ERC-223, ERC-1363) were designed to help with this "token reception" problem.

User rule of thumb: If you are not 100% sure an address can receive tokens safely, **do not send**.

11. ERC-20 in One Minute (Elementary-Level Summary)

- A **token** is a digital thing on Ethereum that can represent value or points.
- **ERC-20** is the **rulebook** that tells tokens how to act so wallets/apps can understand them.
- ERC-20 tokens are **fungible**: each token is the same as another token.
- ERC-20 requires common "buttons" (functions) like: `balanceOf`, `transfer`, `approve`.
- **Warning:** tokens can get stuck if you send them to the wrong smart contract address.

Final Checklist (Before You Use an ERC-20 Token)

- I understand ERC-20 tokens are **fungible** (all the same).
- I know `transfer` sends tokens directly.
- I know `approve` gives spending permission and `transferFrom` spends using that permission.
- I will **double-check** addresses so tokens do not get stuck.

Tip: If you're learning, use a test network or a development wallet so mistakes don't cost real funds.