

Solidity: Programování pro EVM

Solidity je vysokoúrovňový, staticky typovaný jazyk určený k implementaci chytrých smluv na platformách podporujících **EVM (Ethereum Virtual Machine)**. Je to jazyk orientovaný na kontrakty, což znamená, že základní jednotkou kódu není třída, ale „Contract“.

1. Charakteristika jazyka

Solidity má několik unikátních vlastností, které ho odlišují od běžných jazyků:

- **Statické typování:** Typy proměnných (uint, address, string) musí být definovány při kompilaci.
- **Pojem „Address“:** Speciální datový typ pro uložení adresy peněženky nebo jiného kontraktu (např. "0x71C...").
- **Mapping:** Vysoce efektivní asociativní pole (klíč ⇒ hodnota), klíčové pro ukládání zůstatků uživatelů.
- **Modifikátory:** Speciální kousky kódu, které mění chování funkcí (např. pro kontrolu oprávnění).

2. Správa dat a paměti (Gas Optimization)

V Solidity je programátor zodpovědný za to, kolik výpočetních zdrojů aplikace spotřebuje. Každý zápis do blockchainu stojí **Gas**.

- **Storage:** Data uložená trvale v blockchainu. Velmi drahé (stovky tisíc gas jednotek).
- **Memory:** Dočasná data, která existují pouze během provádění funkce. Levnější.
- **Calldata:** Neupravitelná data předávaná funkcím jako argumenty. Nejlevnější varianta pro vstupy.

3. Příklad: Jednoduchý Smart Contract

Následující kód demonstruje základní syntaxi Solidity (verze 0.8.x):

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

contract PristupovySystem {
    address public majitel;
    mapping(address => bool) public povoleneAdresy;

    // Konstruktor se spustí pouze jednou při nasazení kontraktu
    constructor() {
```

```
    majitel = msg.sender; // Nastaví tvůrce jako majitele
}

// Modifikátor pro omezení přístupu
modifier jenMajitel() {
    require(msg.sender == majitel, "Nejste majitel!");
    _; // Zde se pokračuje v těle funkce
}

// Funkce pro přidání uživatele (může volat jen majitel)
function povolitVstup(address _uzivatel) public jenMajitel {
    povoleneAdresy[_uzivatel] = true;
}
}
```

4. Bezpečnostní mechanismy

Kvůli práci se skutečnými penězi obsahuje Solidity vestavěné mechanismy pro ošetření chyb:

- **require():** Slouží k validaci vstupů a podmínek. Pokud podmínka není splněna, transakce se zruší a vrátí zbývající Gas.
- **revert():** Okamžitě ukončí transakci s chybovou zprávou.
- **assert():** Používá se k testování vnitřních chyb a invariantů (stavů, které by nikdy neměly nastat).

5. Standardy a knihovny

Vývojáři v Solidity málokdy začínají „na zelené louce“. Používají prověřené standardy **ERC (Ethereum Request for Comments)**:

- **ERC-20:** Pro zaměnitelné tokeny (peníze).
- **ERC-721:** Pro nezaměnitelné tokeny (NFT).
- **OpenZeppelin:** Průmyslový standard pro knihovny kontraktů, které prošly bezpečnostními audity.

Související články:

- [Chytré smlouvy: Teorie](#)
- [Truffle a Ganache: Vývojové nástroje](#)
- [Ethereum: Platforma pro Solidity](#)

Tagy: it solidity programming smart-contracts ethereum evm development

From:

<http://serviceit.cz/> - **IT ENCYKLOPEDIE**

Permanent link:

http://serviceit.cz/doku.php?id=it:sw:solidity_evm

Last update: **2026/01/02 20:14**

