Solidity 进阶实战（二）

ETH 转账 · 合约交互 · Library · 小学生也能懂的版本

ETH Transfers（合约转帐）

一句话给小学生

从合约转 ETH，有三种方式，**每一种脾气都不一样**。

三种方法总览

- send()：给一点点油，失败不报错
- call()：给很多油，功能强但危险
- transfer()：简单安全，但油很少

send() 方法

send() 範例（含中文🅴解）

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.10;

contract SendEther {
    function sendEth(address payable _to) public payable {
        // 把收到的 ETH 原封不动转出去
        uint amountToSend = msg.value;

        // send 只给 2300 gas
        // 失败时不会 revert，只会回传 false
        bool sent = _to.send(amountToSend);
        require(sent == true, "ETH 转账失败");
    }
}
```

send 的问题

- gas 太少（只能发事件）
- 失败不会自动回滚
- 容易被忘记检查回传值

call() 方法

call() 範例（含中文🅵解）

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.10;

contract SendEther {
    function sendEth(address payable _to) public payable {
        // 把 ETH 转给对方
        uint amountToSend = msg.value;

        // call 会转 ETH，并且转发所有剩余 gas
        (bool success, bytes memory data) =
            _to.call{value: msg.value}("");

        require(success == true, "ETH 转账失败");
    }
}
```

call 的危险

- 会转发所有 gas
- 容易造成 Reentrancy（重入攻击）
- 一定要搭配防护机制

transfer() 方法

transfer() 範例（含中文🅵解）

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.10;

contract SendEther {
    function sendEth(address payable _to) public payable {
        // 直接使用 transfer 转帐
        // 失败会自动 revert
        _to.transfer(msg.value);
    }
}
```

为什么 transfer 很受欢迎？

- 失败会自动回滚
- gas 限制可防止重入攻击
- 语法简单，不易写错

三种转帐方法重点比较（考试会考）

正确答案

.send() 和.call() 会回传 bool，属于低阶函数；.transfer() 失败会直接 revert，属于高阶函数。

接收 ETH（receive / fallback）

一句话

EOA 自动收钱，**合约要写函数才能收钱**。

接收 ETH 範例（含中文🅵解）

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.10;

contract ReceiveEther {

    // 当 msg.data 为空时会被呼叫
    receive() external payable {}

    // 当 msg.data 不为空时会被呼叫
    fallback() external payable {}

    function getBalance() public view returns (uint) {
        // 查看合约余额
        return address(this).balance;
    }
}
```

呼叫其他合约（Interface）

小学生版

Interface = **别人合约的使用说明书**

Interface 範例（含中文🅴解）

```
interface MinimalERC20 {
    // 只声明我们要用的函数
    function balanceOf(address account)
        external view returns (uint256);
}

contract MyContract {
    MinimalERC20 externalContract;

    constructor(address _externalContract) {
        // 建立外部合约实例
        externalContract = MinimalERC20(_externalContract);
    }

    function mustHaveSomeBalance() public {
        uint balance =
            externalContract.balanceOf(msg.sender);

        require(balance > 0, "你没有代币");
    }
}
```

Import（分档案写合约）

一句话

Solidity 也可以「拆档案」。

Import 範例（含中文🅴解）

```
// Foo.sol
contract Foo {
    string public name = "Foo";
}

// Import.sol
import "./Foo.sol";

contract Import {
    Foo public foo = new Foo();

    function getFooName()
        public view returns (string memory)
    {
        return foo.name();
    }
}
```

Libraries（函式工具箱）

一句话

Library = **不能存钱的工具合约**

Library 特性

- 没有状态变量
- 不能转 ETH
- 只部署一次，节省 gas

SafeMath 範例（含中文🅵解）

```
library SafeMath {
    function add(uint x, uint y)
        internal pure returns (uint)
    {
        uint z = x + y;
        // 检查是否发生 overflow
        require(z >= x, "整数溢位");
        return z;
    }
}

contract TestSafeMath {
    function testAdd(uint x, uint y)
        public pure returns (uint)
    {
        return SafeMath.add(x, y);
    }
}
```

本章小学生总结

一页背完

- send / call / transfer 差别一定要会
- 合约收 ETH 要写 receive / fallback
- Interface 是合约说明书
- Library 是工具箱

教学重点：ETH 转帐是安全事故最多的地方，一定要懂原理。