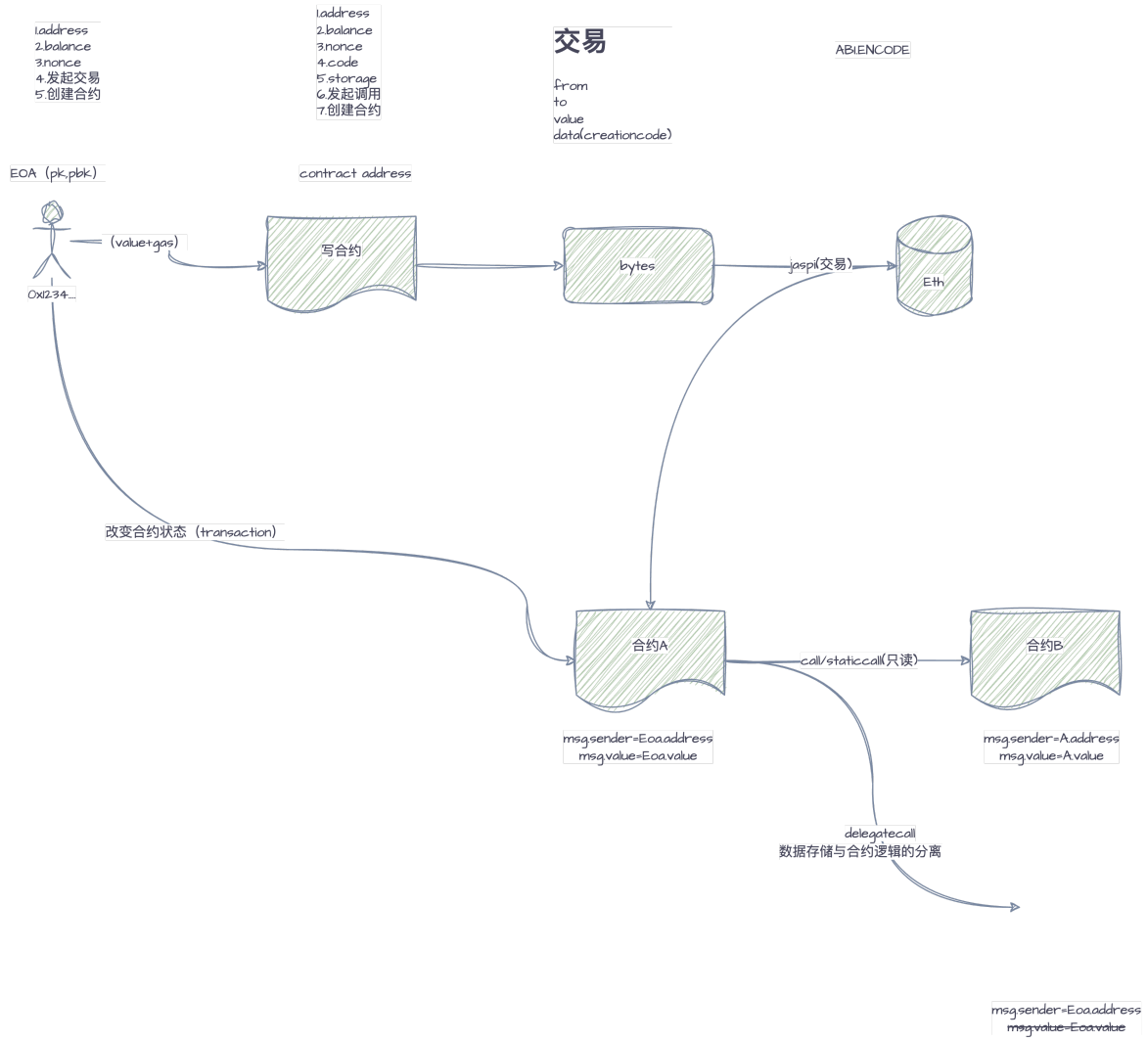
上半节

1、MSG的解释

（1）介绍MSG的概念



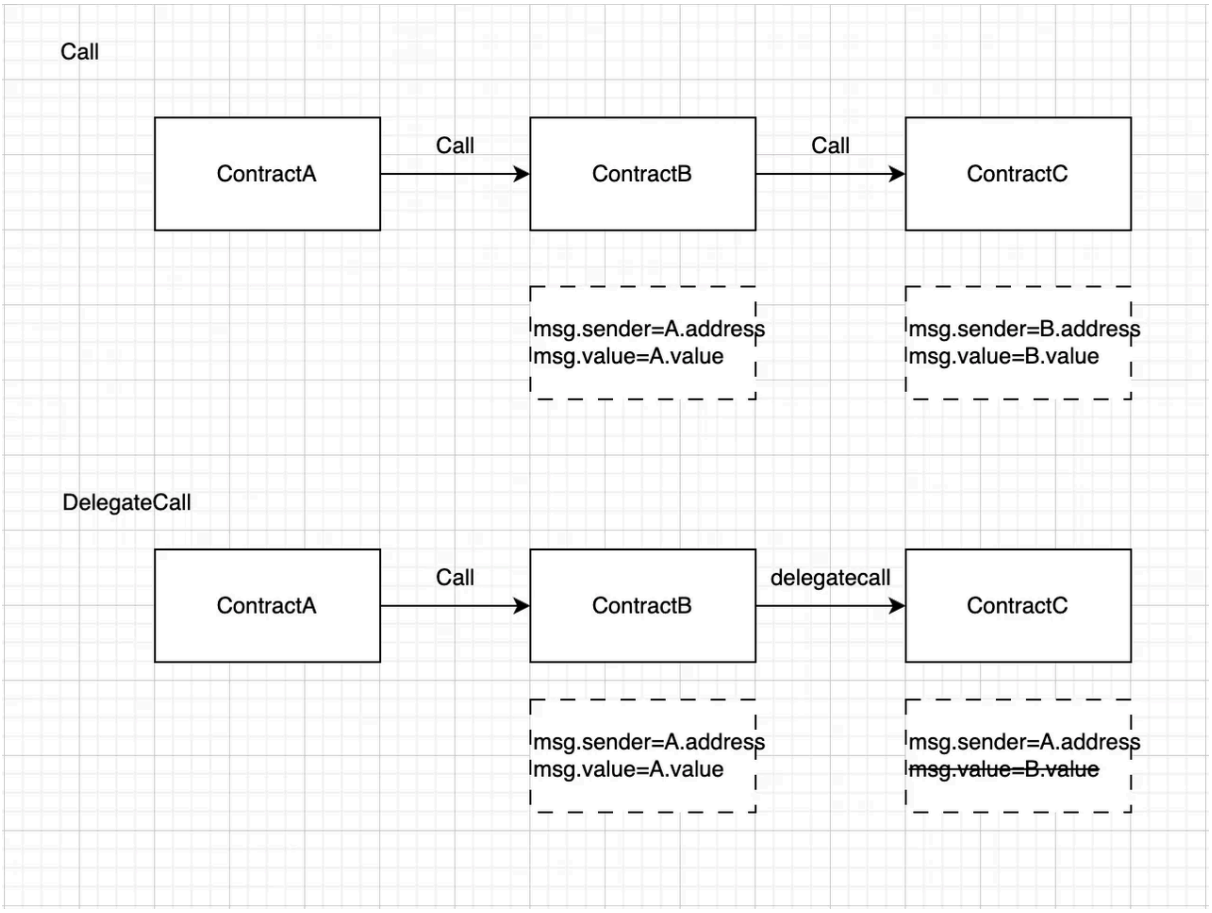
（2）REMIX演示EOA调用、CONTRACT调用和DELEGATE调用下面msg的不同

```
//SPDX-License-Identifier:MIT
pragma solidity 0.8.26;
import "hardhat/console.sol";

contract CallMsg{
```

```
// 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
// 100
// 0x7eecd3b6
// 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
//EOA→CallMsg
    function msgFrom() external  payable {
        console.logAddress(msg.sender);
        console.logUint(msg.value);
        console.logBytes(msg.data);
        console.logAddress(tx.origin);
        console.logAddress(address(this));
        //下面的内容再第二节课演示
        // console.logBytes4(CallMsg.msgFrom.selector);
        // console.logBytes(abi.encodeWithSignature("msgFrom()"));
        // console.logBytes4(bytes4(keccak256("msgFrom()")));
    }

}

// 0xb27A31f1b0AF2946B7F582768f03239b1eC07c2c
// 0
// 0x7eecd3b6
// 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
// EOA→CallContact→CallMsg
contract CallContract{

    CallMsg call;
    constructor(CallMsg _call){
        call=_call;
    }

    function callContract() external payable {
        call.msgFrom();
    }

}

// 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
```

```
// 100
// 0x7eecd3b6
// 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
// EOA→CallDelegate→CallMsg
contract CallDelegate{
    CallMsg call;
    constructor(CallMsg _call){
        call=_call;
    }

    function delegateCall() external payable{
       (bool _r,) =  address(call).delegatecall(abi.encodeWithSignature("msgF
rom()"));
       require(_r);
    }

}
```

（3）一张图总结



When a function is executed with a delegate call these values do not change.

- address(this)
- msg.sender
- msg.value

2、CALL/DELEGATECALL/STATICALL

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
import "https://github.com/NomicFoundation/hardhat/blob/main/packages/
hardhat-core/console.sol";

contract Called{
    uint256 public number;
```

```
    function inc() external {
        ++number;
    }

    function setN(uint256 _number) external returns(uint256){
        number=_number;
        return number;
    }

}
//EOA→CALLER→CALLED
contract Caller{
    uint256 public number;
    address to;
    constructor(address _to){
        to=_to;
    }


    function inc() external {
       (bool _r,bytes memory data)= to.call(abi.encodeWithSignature("inc
()"));
       require(_r,"execute fail");
       console.logBytes(data);
    }


    function setN(uint256 _number) external {
      (bool _r,bytes memory data)= to.call(abi.encodeWithSignature("setN(ui
nt256)",_number));
       require(_r,"execute fail");
       console.logBytes(data);
    }

    function delegateSetN(uint256 _number) external {
      (bool _r,bytes memory data)= to.delegatecall(abi.encodeWithSignature
("setN(uint256)",_number));
       require(_r,"execute fail");
```

```
        console.logBytes(data);
    }
     function staticSetN() view external {
      (bool _r,bytes memory data)= to.staticcall(abi.encodeWithSignature("nu
mber()"));
        require(_r,"execute fail");
        console.logBytes(data);
    }
}
```

3、SEND/TRANSFER/CALL的区别

Reentrancy Attack

（1）如何向一个地址发送以太 send/transfer/call

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
import "hardhat/console.sol";

contract Called{
    function balance() external view  returns(uint256){
        return address(this).balance;
    }
    receive() external payable {
        console.log(address(this));
    }
}


contract Caller{


    address payable  called;

    constructor(address payable to){
```

```
        called=to;
    }

    function sender() external payable {
        bool result=called.send(3);
        require(result);
    }

    function transfer() external payable{
        called.transfer(3);
    }

    function call() external payable{
       (bool result,bytes memory data)=called.call{value:3}("");
       require(result);
    }
}
```

🧪实验1: 如果called没有receive方法，使用send和transfer会成功么？

答案：send会成功，以太不退回；transfer会失败。

修复：called 增加receive让他能够接受以太

🧪实验2: 修改receieve方法，增加 console.log(address(this));会有什么结果

答案：send/transfer会失败，但是cal会成功

辨析：send/transfer只会transfer 2300个gas， cal会transfer所有

（2）三者的区别

- send返回值为bool，需要显式判断结果
- transfer如果转账失败会直接回滚
- send和transfer 会forward 2300个gas，如果下游操作超过2300个gas，转账失败
- call会forward所有的gas