理解 ERC20 的授权

对比 ETH / Token 转账

向 0xabcd… 转移 10 ETH， Tx:

```
{
    to: '0xabcd…',
    value: 10 * 10^18,
    data: "0x" // 也可以附加消息
}
```

如果 ETH 接收者是合约地址，receive() 会被调用

向 0xabcd… 转移 10 Token， Tx:

```
{
  to: '0x6874..', // Token 合约地址
  value: 0,
  data: "transfer(0xabcd…,10*10^18) ABI编码"
}
```

不管接收者是谁，都只有 Token 合约代码被执行

持有的余额是记录在 Token 合约内，不记录在用户地址下

练习题

- 部署自己的 Token（ERC20）
- 编写一个 TokenBank， 可以将 Token 存入 TokenBank:
  - 记录每个用户存入的 token 数量
  - 管理员/用户 可以提取所有的Token （withdraw 方法）。

https://github.com/lbc-team/Web3-BootCamp-Practice/blob/main/TokenBank_v1/TokenBank.sol

ERC20 – 授权

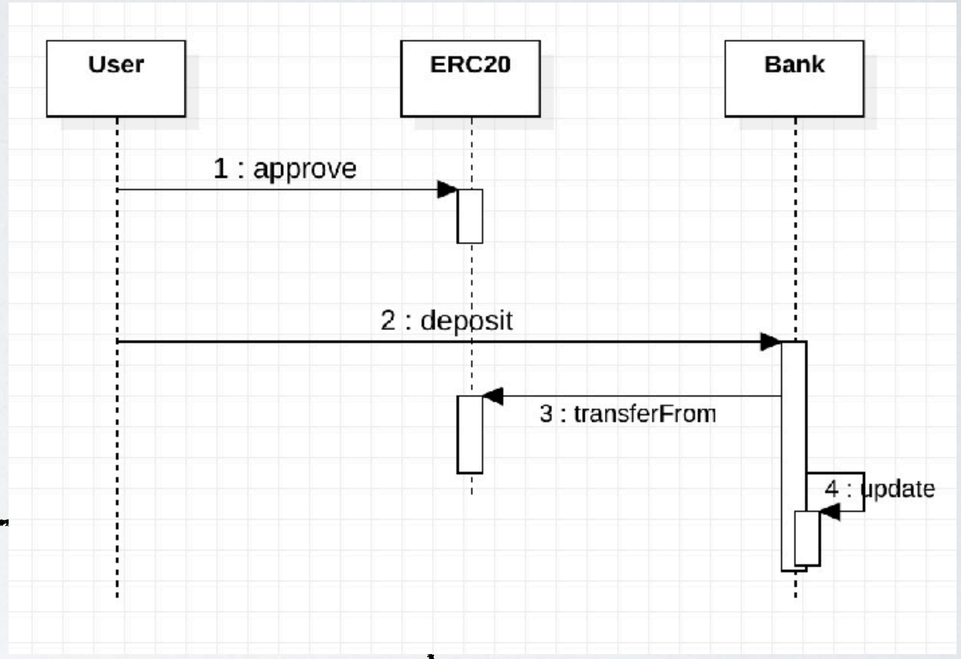
- 如何记录用户的存款量?
  - 直接调用 ERC20 合约向 TokenBank 合约地址转账不可行
- ERC20 还有 approve() / transferFrom()

```
function approve(address spender, uint256 value)  … {
    _allowances[msg.sender][spender] = value;
}

function transferFrom(address from, address to, uint256 value) … {
   _allowances[from][msg.sender] -= value;
   _transfer(from, to, value);
}
```

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol

ERC20 – 授权

- 分两笔交易：
  - 用户 A 调用 ERC20 的 Approve(tokenBank，数量);
  - 用户 A 调用 tokenBank 的 deposit(); 完成存款

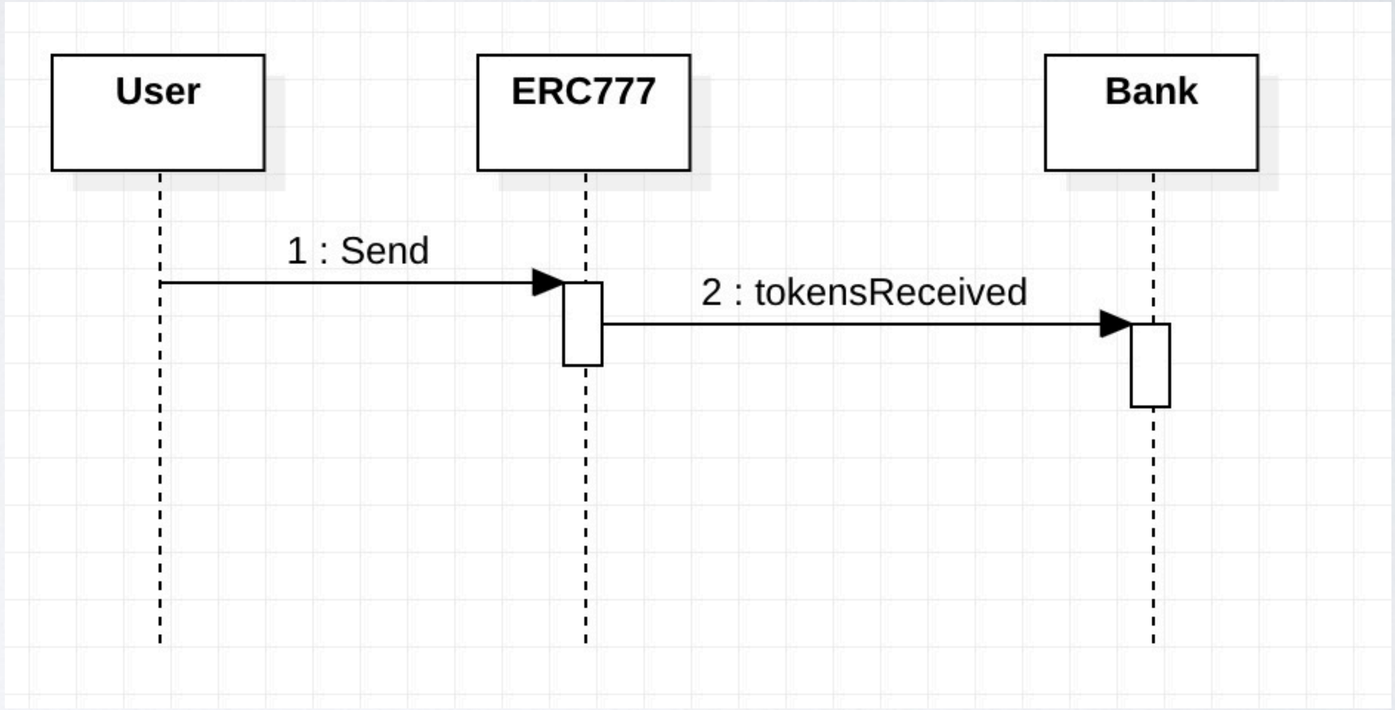



```
pragma solidity ^0.8.0;

contract TokenBank {
   mapping(address => uint) deposited;
   function deposit(uint amount) {
     IERC20(addr).transferFrom(msg.sender, address(this), amount);
     deposited[msg.sender] += amount;
   }
}
```

- 若要多次存款，如何减少授权次数?

ERC777

- ERC20 问题
  - 依靠：授权、授权、授权
  - 转账无法携带额外的信息。
  - 误转入合约被锁死。



- ERC777:
  - ERC777： send(to, value, data)
  - 防止误入合约被锁死。
  - 即便是普通地址也可以实现回调（如何实现的呢？）
  - ERC777 通过 全局注册表合约（ERC1820）的注册监听回调

ERC777

```
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC777/ERC777.sol";

contract MyERC777 is ERC777 {
    constructor() ERC777("MY777", "M777", new address[](0))
    {
        _mint(msg.sender, 1000 * 10 ** 18, "", "");
    }
}
```

源码：https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v4.9/contracts/token/ERC777/ERC777.sol

ERC777

```
pragma solidity ^0.8.0;

function send(from, to, amount …) {
  _callTokensToSend(operator, from, to, amount,…);

  _move(operator, from, to, amount, userData, operatorData);

  _callTokensReceived(operator, from, to, amount, userData,…);

}

// 检查是否有注册回调，有的话就调用一下
function _callTokensReceived( operator,   from,   to, amount) private {
   address implementer = _ERC1820_REGISTRY.getInterfaceImplementer(to,
_TOKENS_SENDER_INTERFACE_HASH);
        if (implementer != address(0)) {
            IERC777Sender(implementer).tokensToSend(operator, from, to, amount,..);
        }
    }
```

ERC1363

- Payable Token
- 3 对额外的函数
  - transferAndCall
  - transferFromAndCall
  - approveAndCall

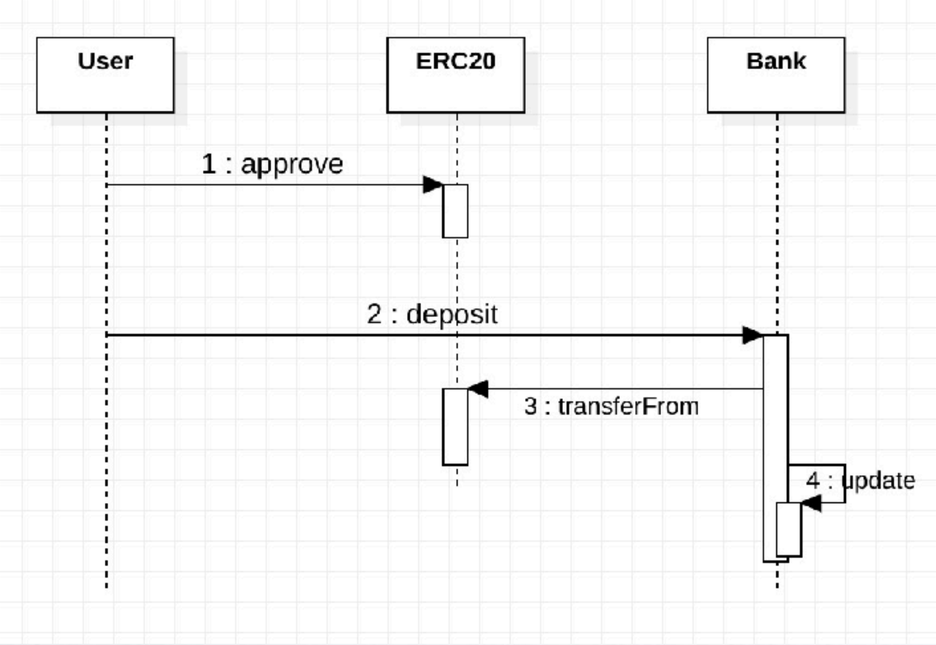
https://github.com/vittominacori/erc1363-payable-token

https://learnblockchain.cn/article/8549

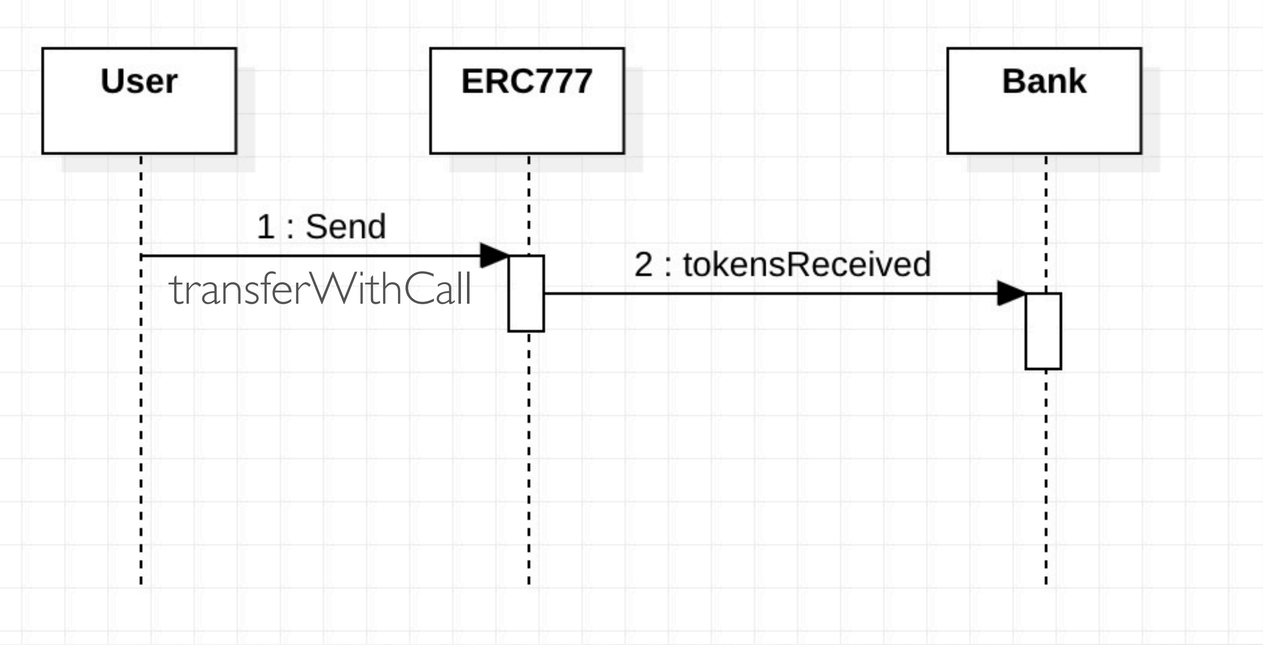
ERC20-Callback

```
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/utils/Address.sol";

function transferWithCallback(address to, uint256 amount) external returns (bool) {
        _transfer(msg.sender, to, amount);

        if (to.isContract()) {
            bool rv = TokenRecipient(to).tokensReceived(msg.sender, amount);
            require(rv, "No tokensReceived");
        }

        return true;
    }
```

Token 总结



ERC1363



ERC20 的一些坑

- 怎么办？ 加一个返回值判断么，然而有些 Token 没有返回值。
  - 如： USDT： https://etherscan.io/address/0xdac17f958d2ee523a2206206994597c13d831ec7#code
- 一些 Token 在实现时，转账失败没有回退，而是返回 false。
  - 导致，合约以为转账成功了，但却没有
  - 如：ZRX: https://etherscan.io/address/0xe41d2489571d322189246dafa5ebde1f4699f498#code
- 那该怎么办?
  - ERC20 转账应该总是使用 Openzeplin 的 SafeERC20 safeTransfer：
  - https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/utils/SafeERC20.sol

https://learnblockchain.cn/article/3074

练习题

- 发行一个具备回调功能的 ERC20 Token
- 扩展 TokenBank， 实现在转账回调中，将 Token 存入 TokenBank

https://decert.me/quests/4df553df-fbab-49c8-a05f-83256432c6af

交互任务

- 使用真实的 token 完成兑换操作（在 Uniswap 等 DEX 中）

https://decert.me/quests/65e9c4a1-a2ee-41ea-b8d7-7a5a1a945cbc