# 以太坊智能合约开发入门

Michael@Celer Network

2021/07/29

# 区块链1.0 - 比特币

## 比特币：一种点对点的电子现金系统

## Bitcoin: A Peer-to-Peer Electronic Cash System


原文作者：中本聪（Satoshi Nakamoto）

翻译：Bitcoinblogger.com 独家赞助

作者邮箱：Satoshin@gmx.com

www.bitcoin.org



**[摘要]**：本文提出了一种完全通过点对点技术实现的电子现金系统，它使得在线支付能够直接由一方发起并支付给另外一方，中间不需要通过任何的金融机构。虽然数字签名（Digital signatures）部分解决了这个问题，但是如果仍然需要第三方的支持才能防止双重支付（double-spending）的话，那么这种系统也就失去了存在的价值。我们(we)在此提出一种解决方案，使现金系统在点对点的环境下运行，并防止双重支付问题。该网络通过随机散列（hashing）对全部交易加上时间戳（timestamps），将它们合并入一个不断延伸的基于随机散列的工作量证明（proof-of-work）的链条作为交易记录，除非重新完成全部的工作量证明，形成的交易记录将不可更改。最长的链条不仅将作为被观察到的事件序列（sequence）的证明，而且被看做是来自 CPU 计算能力最大的池（pool）。只要大多数的 CPU 计算能力都没有打算合作起来对全网进行攻击，那么诚实的节点将会生成最长的、超过攻击者的链条。这个系统本身需要的基础设施非常少。信息尽最大努力在全网传播即可，节点(nodes)可以随时离开和重新加入网络，并将最长的工作量证明链条作为在该节点离线期间发生的交易的证明。


# 区块链1.0 - 比特币

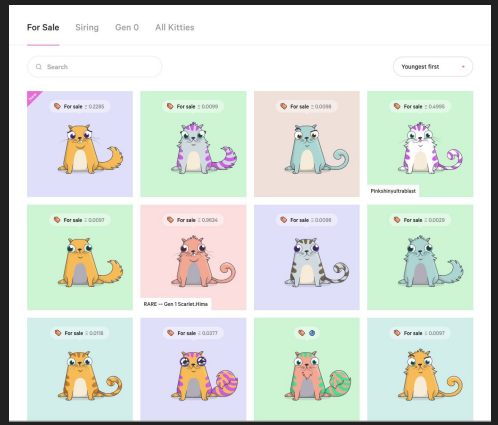
- 中本聪共识机制 - 去中心化
- Proof of Work (PoW) 工作量证明 - 不可伪造
- 哈希算法和不对称加密 - 安全可靠
- 总量恒定 - 稀缺、保值
- 公开、透明、可溯源、不可篡改
- 一定程度的匿名性
- 相对低廉的手续费
- UTXO模型 - 类似纸币

# 区块链2.0 - 以太坊

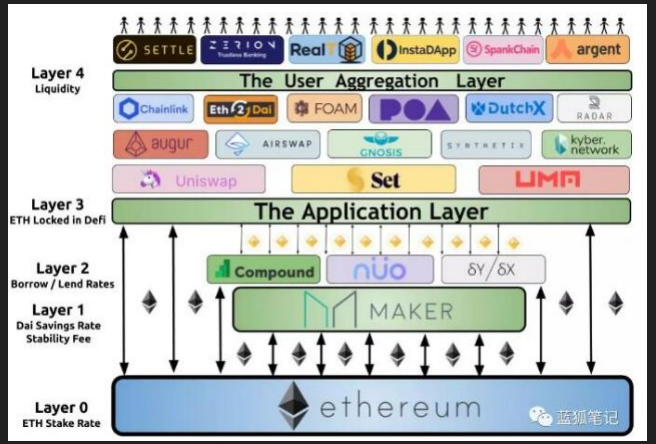
2013年，Vitalik Buterin (V神) 等人发明Ethereum
**支持可编程智能合约**，实现去中心化通用计算

```
/**
 * @title ERC20 interface
 * @dev see https://github.com/ethereum/EIPs/issues/20
 */
interface IERC20 {
  function totalSupply() external view returns (uint256);

  function balanceOf(address who) external view returns (uint256);

  function allowance(address owner, address spender)
    external view returns (uint256);

  function transfer(address to, uint256 value) external returns (bool);

  function approve(address spender, uint256 value)
    external returns (bool);

  function transferFrom(address from, address to, uint256 value)
    external returns (bool);

  event Transfer(
    address indexed from,
    address indexed to,
    uint256 value
  );

  event Approval(
    address indexed owner,
    address indexed spender,
    uint256 value
  );
}
```

ERC20通证接口
发行资产



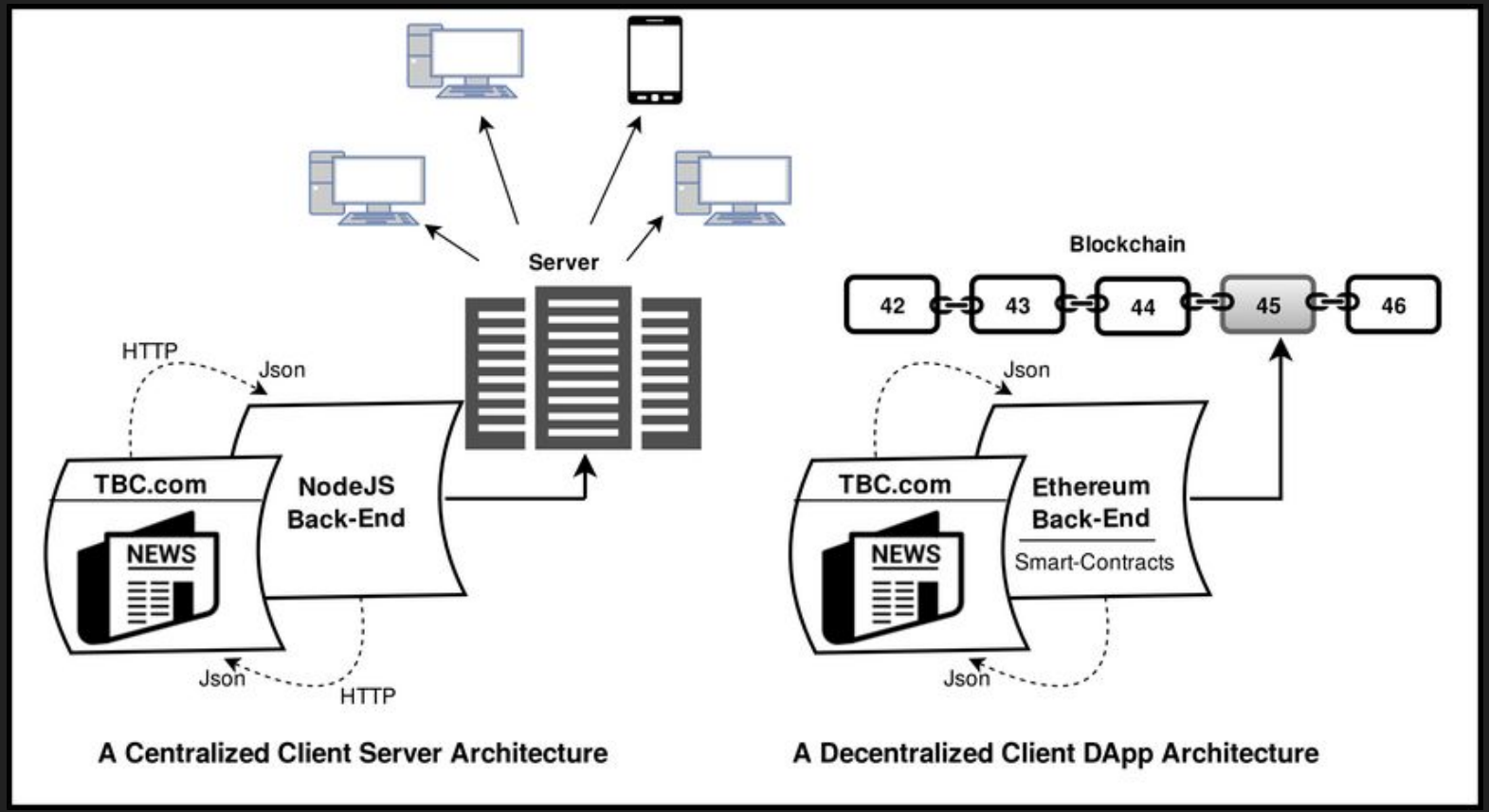

ERC721非同质化通证(NFT)接口
数字艺术作品、元宇宙




Decentralized Finance (DeFi)
去中心化金融

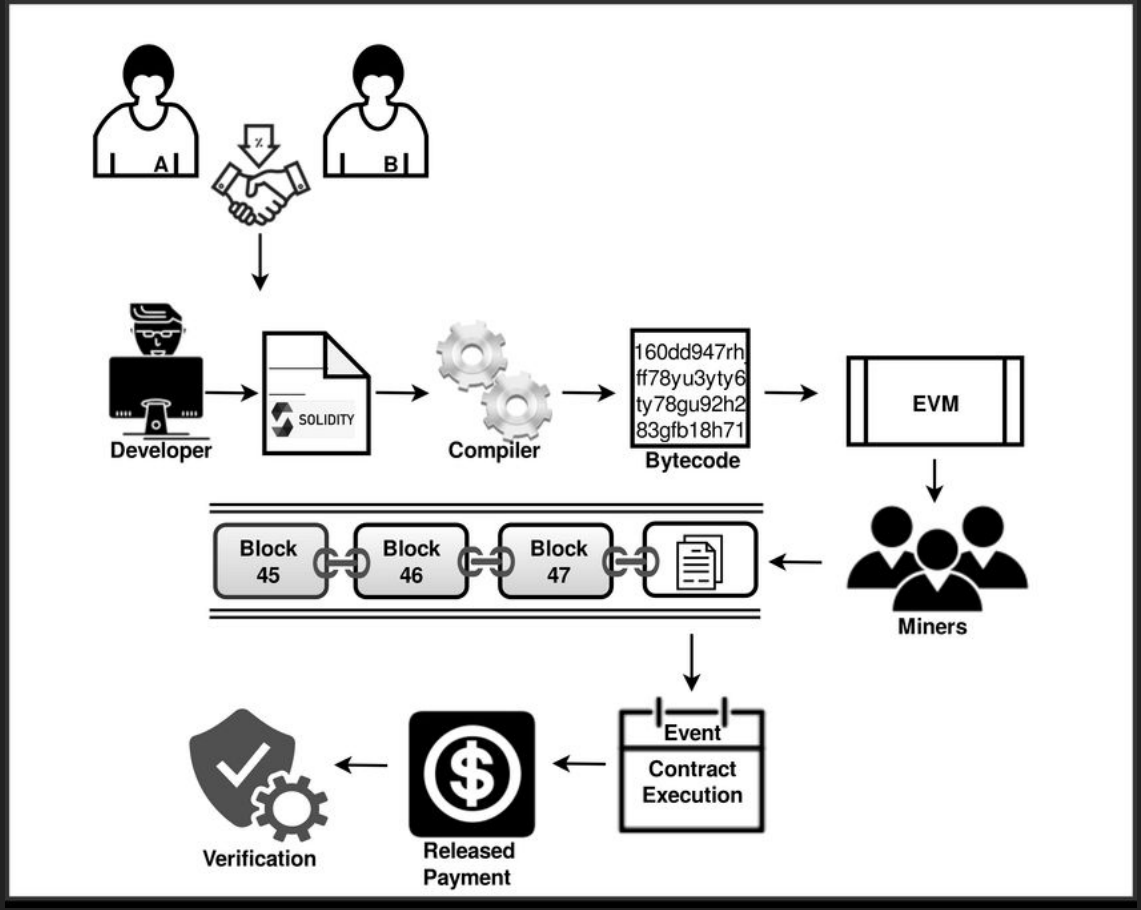
# 区块链2.0 - 以太坊

- 可编程的智能合约、图灵完备的虚拟机(EVM)和开发语言 (Solidity)
- EVM类比JavaScript VM，Solidity类比JavaScript
- 除了资产以外的更多数据和状态上链
- 2017 ICO与通证经济的爆发
- 以太坊竞争者 (EOS, TRX, NEO....)，公链之争
- EVM一统江湖的趋势
- Decentralized Finance (DeFi)，去中心化金融
- Non-Fungible Token (NFT)，数字艺术收藏品、链上游戏和元宇宙 (Metaverse)




传统web app与dApp架构

Source: Smart Contract: Attacks and Protections

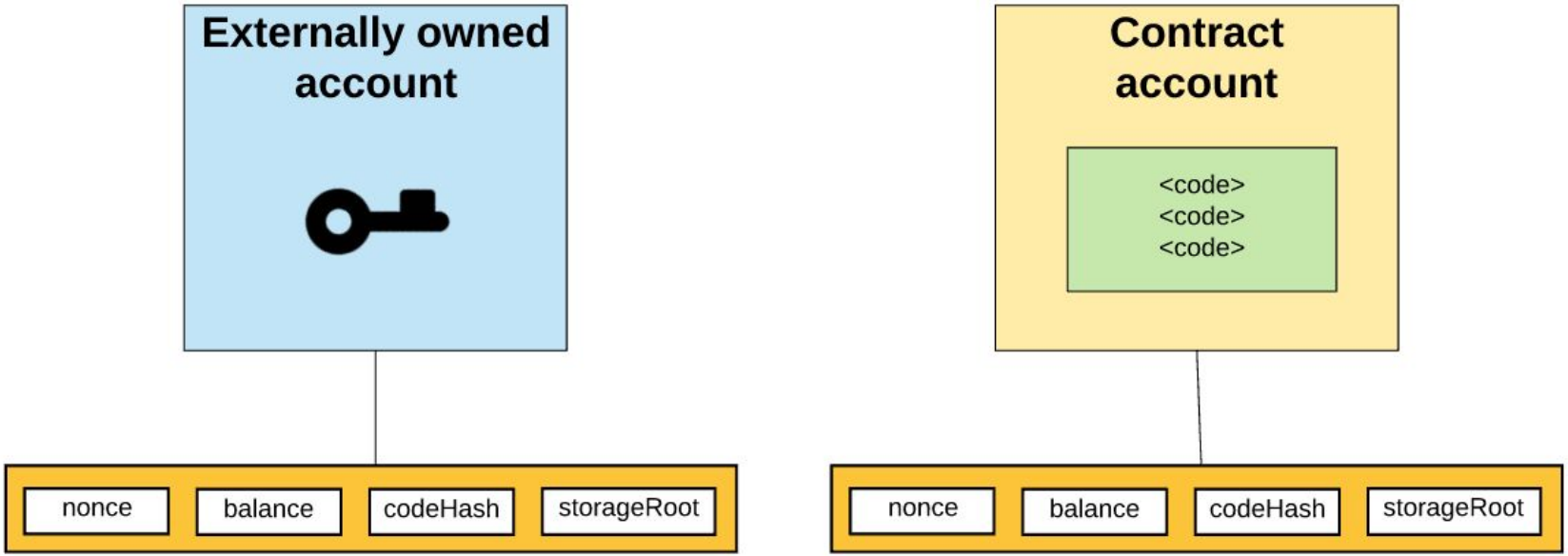



智能合约执行过程

Source: Smart Contract: Attacks and Protections

# 智能合约虚拟机：EVM

- 基于堆栈的虚拟机 (stack machine)
- 基于账户 (Account-based) 的模型，而不是比特币的UTXO
- 两类账户: 外部账户 (EOA) 和合约账户
- 默克尔树 (Merkle Tree) 与状态根
- 256位字长
- 每个字节码有自己的Gas消耗，每个区块有Gas限制
  - Tx fee = gasUsed * gasPrice




两类以太坊账户: 外部账户(EOA)和合约账户

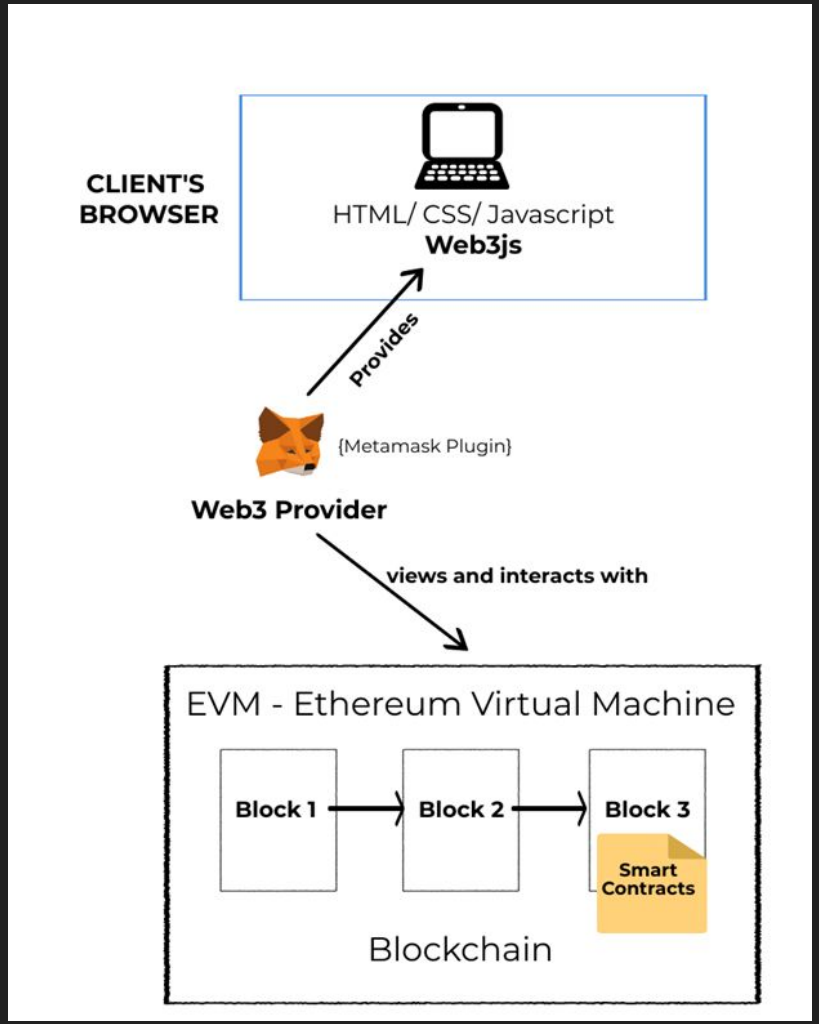
# 智能合约开发语言: Solidity

- 官方文档
- 命令式 (Imperative)，强类型
- 稍类似TypeScript
- Contract, Interface与Library
- Application Binary Interface (ABI)
- 继承和组合
- 事件 (Events)
- 存储类型：
  - calldata: 合约调用时传入的参数
  - stack: 本地的简单变量
  - memory: 一次合约调用中使用的存储
  - storage: 保存在以太坊状态中，成本高

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.5.0 <0.9.0;

library Balances {
    function move(mapping(address => uint256) storage balances, address from, address to, uint
        require(balances[from] >= amount);
        require(balances[to] + amount >= balances[to]);
        balances[from] -= amount;
        balances[to] += amount;
    }
}

contract Token {
    mapping(address => uint256) balances;
    using Balances for *;
    mapping(address => mapping (address => uint256)) allowed;

    event Transfer(address from, address to, uint amount);
    event Approval(address owner, address spender, uint amount);

    function transfer(address to, uint amount) public returns (bool success) {
        balances.move(msg.sender, to, amount);
        emit Transfer(msg.sender, to, amount);
        return true;

    }

    function transferFrom(address from, address to, uint amount) public returns (bool success)
        require(allowed[from][msg.sender] >= amount);
        allowed[from][msg.sender] -= amount;
        balances.move(from, to, amount);
        emit Transfer(from, to, amount);
        return true;
    }

    function approve(address spender, uint tokens) public returns (bool success) {
        require(allowed[msg.sender][spender] == 0, "");
        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);
        return true;
    }

    function balanceOf(address tokenOwner) public view returns (uint balance) {
        return balances[tokenOwner];
    }
}
```

# dApp组件 : Web3 / Ethereum Provider

- 负责管理用户私钥
- 发送交易和签名
- 调用JSON-RPC服务与区块链交互
- 桌面: MetaMask插件
- 移动端
  - imToken
  - Trust Wallet
  - Math Wallet
  - ...



# 智能合约开发工具

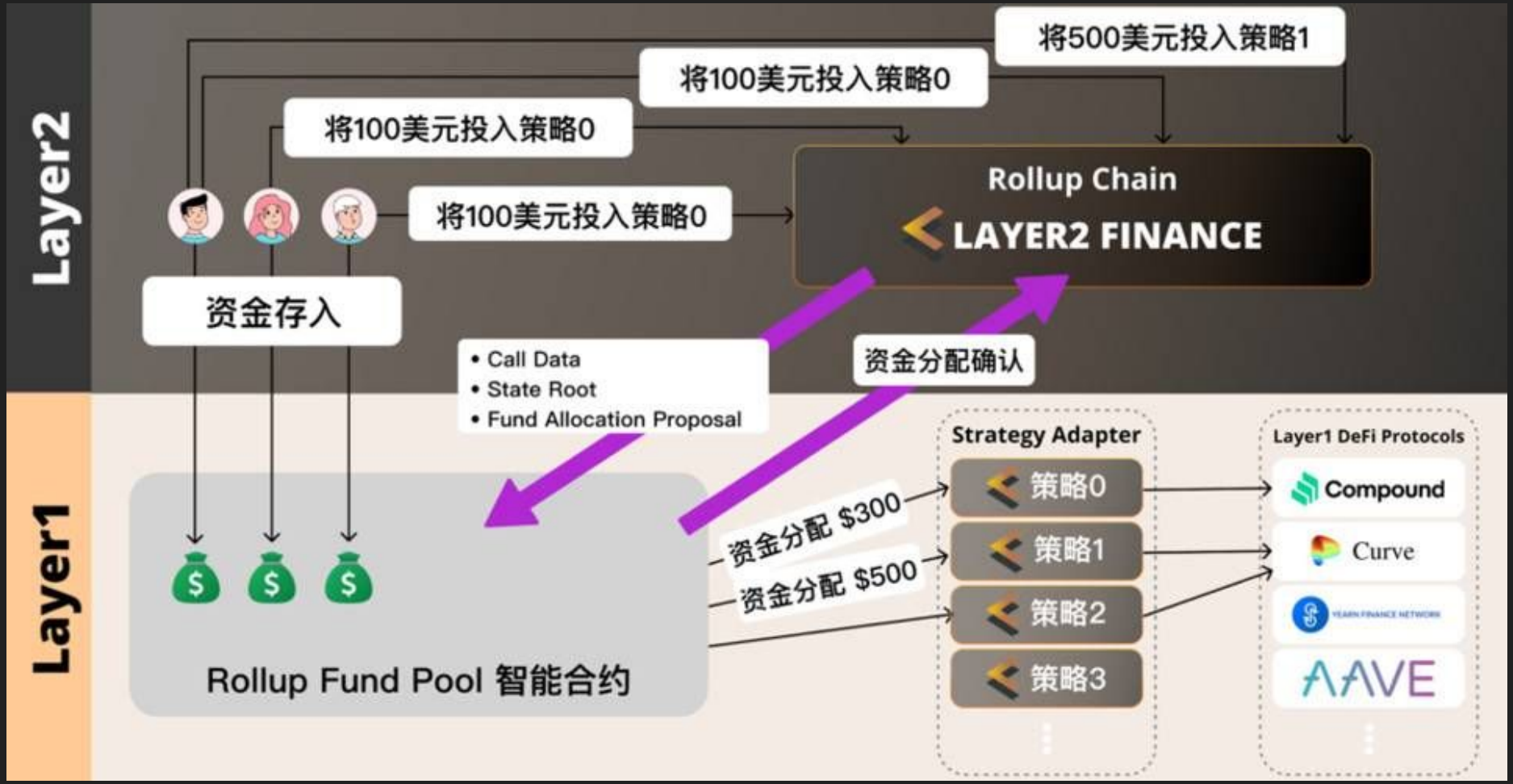
- Remix IDE (个人不使用)
- VSCode + Solidity plugin (推荐)
- 编译器: solc
- Hardhat: 智能合约编译和部署管理器 (强烈推荐)
  - 官方文档
  - 完整教程
- Waffle: 智能合约测试框架 (强烈推荐)
  - 官方文档
- Truffle (不推荐)

# dApp开发框架

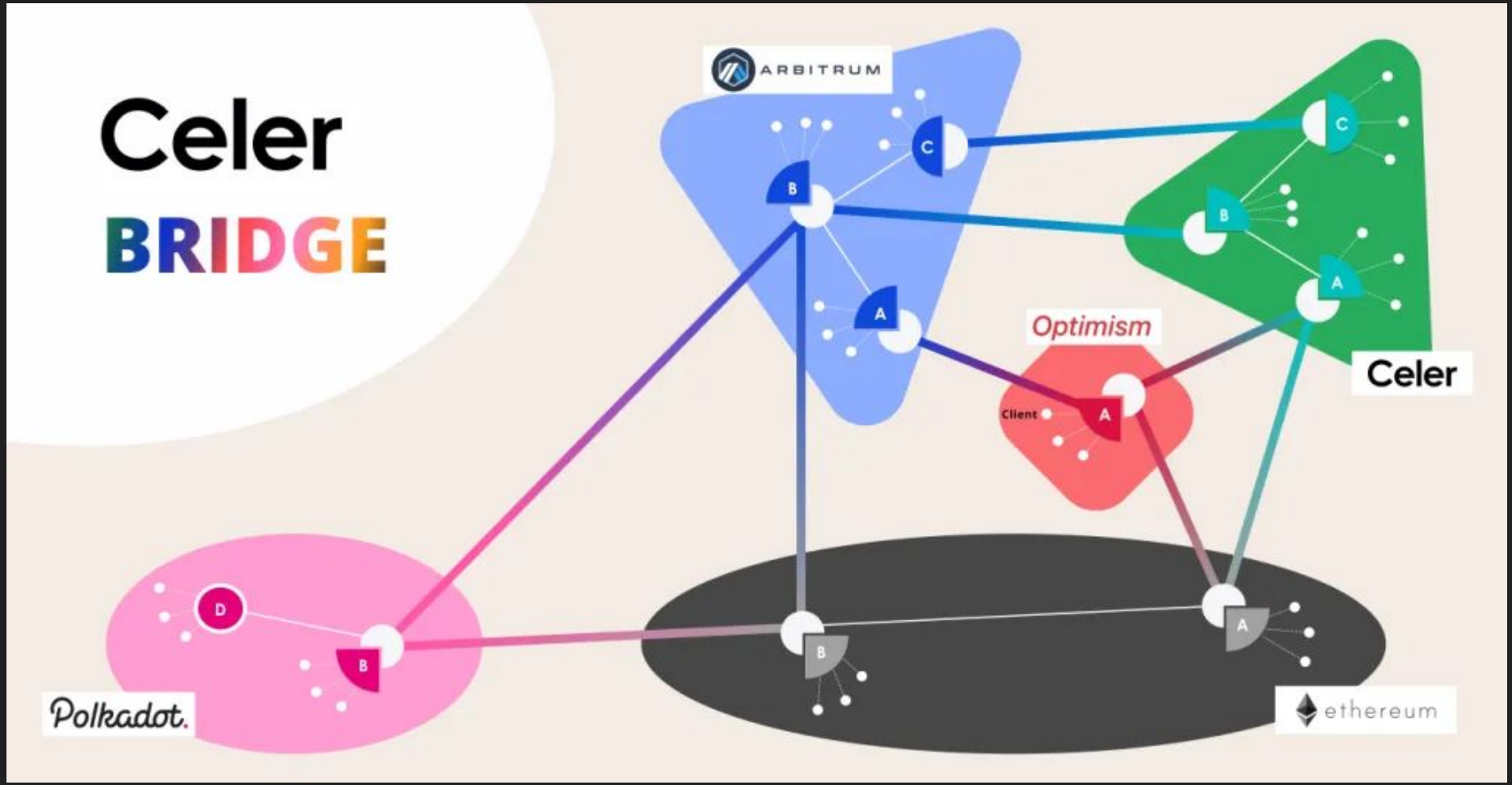
- Ethers.js 管理Ethereum provider和以太坊交互 (强烈推荐)
  - 发送交易
  - 调用合约
  - 监听链上事件
  - 用私钥签名
- web3.js (不推荐)
- TypeChain 用于生成强类型的合约接口文件 (强烈推荐)
- 其他工具:
  - ESLint检查代码风格
  - Prettier格式化代码，有插件可用于 Solidity
- 完整开发模版

# Celer Network相关项目

- 专注于二层扩容和跨链技术
- cChannel: 以太坊上的第一个通用状态通道网络
- SGN: 基于Cosmos SDK的侧链，提供数据可用性
- Layer2.Finance: 基于optimistic rollup，Layer2上的DeFi门户与聚合器
- cBridge: 基于哈希时间锁 (HTLC) 的原子交换，简单好用的跨链桥




Layer2.Finance


Celer
BRIDGE
ARBITRUM
C
B
A
Optimism
Client
A
C
B
A
Celer
D
B
Polkadot.
B
A
ethereum

cBridge