区块链技术与应用

计算机科学与技术学院　李京

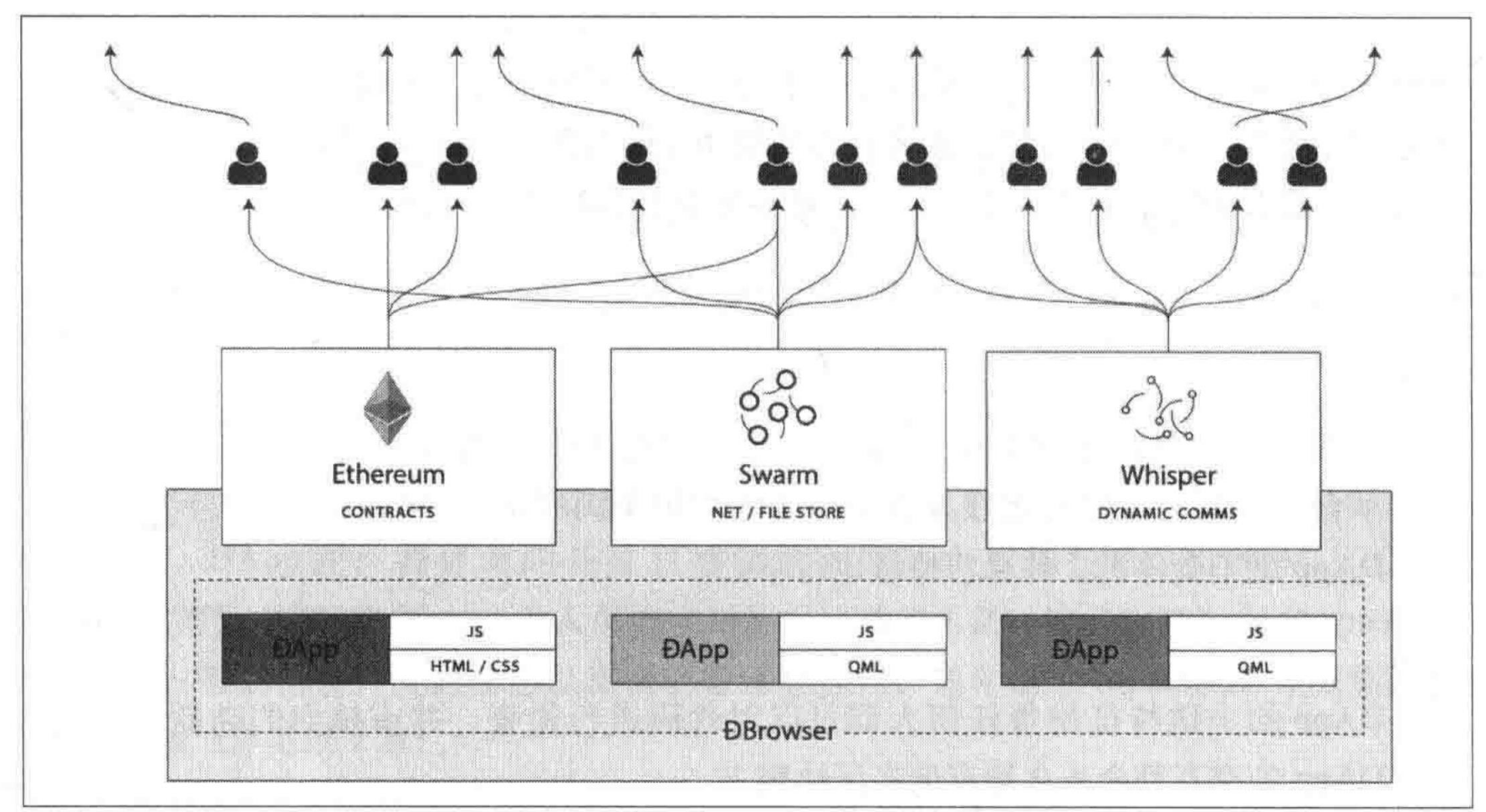

09章 去中心化应用

目 录

9.1 去中心化应用DApp

- 去中心化应用（ Decentralized Application，DApp)：一个大部分或者全部去中心化的应用，去中心化的含义是任何一个节点都不会控制其他节点的工作。
- DApp通过智能合约实现应用控制逻辑和支付方法的去中心化。区块链2.0通常可理解为一个通用的智能合约平台，在其上支撑DApp的运行。
- DApp的优势在于区块链所特有的数据完备、不可篡改且有价值传递功能，在实际使用中体现在可信的数据共享、安全保障的交易、行业生产关系变更、减少运维成本等方面。DApp的优点：
  - 弹性：只要区块链平台仍在运行，DApp永不停止。
  - 透明：DApp上链，任何人可检查业务代码，并审核其功能。并且交互过程也会永久存在链上。
  - 不可篡改：上链的智能合约代码无法修改。
  - 无需授权：只要用户能够访问区块链节点，就可以与DApp进行交互，而不会受到任何集中式控制力量的干扰。

Web3 DApp



web3 是指使用智能合约和 **P2P** 技术的去中心化网络

去中心化应用DApp

- 一个DApp应用一般包括：
  - 后端软件（应用逻辑）
  - 前端软件
  - 数据存储
  - 消息通信
  - 域名解析

这些组件中后端软件一般是去中心化的，通常业务逻辑是以智能合约的形式存储在区块链上，其他组件可能是中心化的，也可能是去中心化的。

去中心化应用DApp

DApp—后端（智能合约）

智能合约被用来存储应用的业务逻辑（程序代码）和相关的状态信息，可看做一个常规的应用程序的服务端组件。以太坊的智能合约允许构建出复杂的架构，网络中的智能合约可以相互调用并传递数据，以及随时读写自己的状态变量。由于智能合约的执行非常昂贵，例如以太坊智能合约计算需要消耗gas，因此其复杂性受限于区块gas的限制。

智能合约架构设计的主要考虑因素：

- 一旦合约代码被部署，就无法对其进行修改。
- DApp的大小，一个体量非常大的合约在部署和使用的时候会花费非常多的gas。

去中心化应用DApp

DApp—前端（Web用户界面）

DApp客户端界面开发只需要基础的Web前端技术。与DApp的交互，如签名消息、发送交易、以及密钥管理等，通常通过MetaMask等浏览器扩展工具完成，MetaMask 是一个开源的以太坊钱包。

前端通常通过Web3.js JavaScript库连接到以太坊，该库与前端资源捆绑到一起，并由Web服务器提供给浏览器。

去中心化应用DApp

DApp—数据存储

由于昂贵的gas费用以及当前较低的区块gas上限，智能合约不太适合存储和处理大量数据，因此大多数DApp利用链下数据存储服务。数据存储平台可以是中心化的，或者点对点形式（IPFS、Swarm）的。

去中心化的点对点存储系统是存储和分发大型静态资产如图片、视频以及客户端应用程序其余部分的理想选择。

- 星际文件系统IPFS：是一个去中心化的、内容可寻址的存储系统，可以在P2P网络中分配存储的对象。内容的每一小块都被哈希处理，并且这个哈希值可以标识文件，通过哈希值在IPFS的任何节点可以将文件取回。
- Swarm：是以太坊基金会创建，是另一个内容可寻址，类似于IPFS的P2P存储系统。

去中心化应用DApp

DApp—去中心化消息通信协议

任何应用都会包含的重要组件是进程之间的通信。这意味着不同的应用之间、相同应用的不同实例之间、以及同一应用的不同用户之间可以交换信息。通信可以依赖中心化的服务器进行，但也有许多基于P2P网络的替代方案。

- Whisper：是一种加密通讯协议，允许节点间安全地直接发送信息，帮助信息发送者和信息接受者屏蔽掉多余的第三方。

去中心化应用DApp

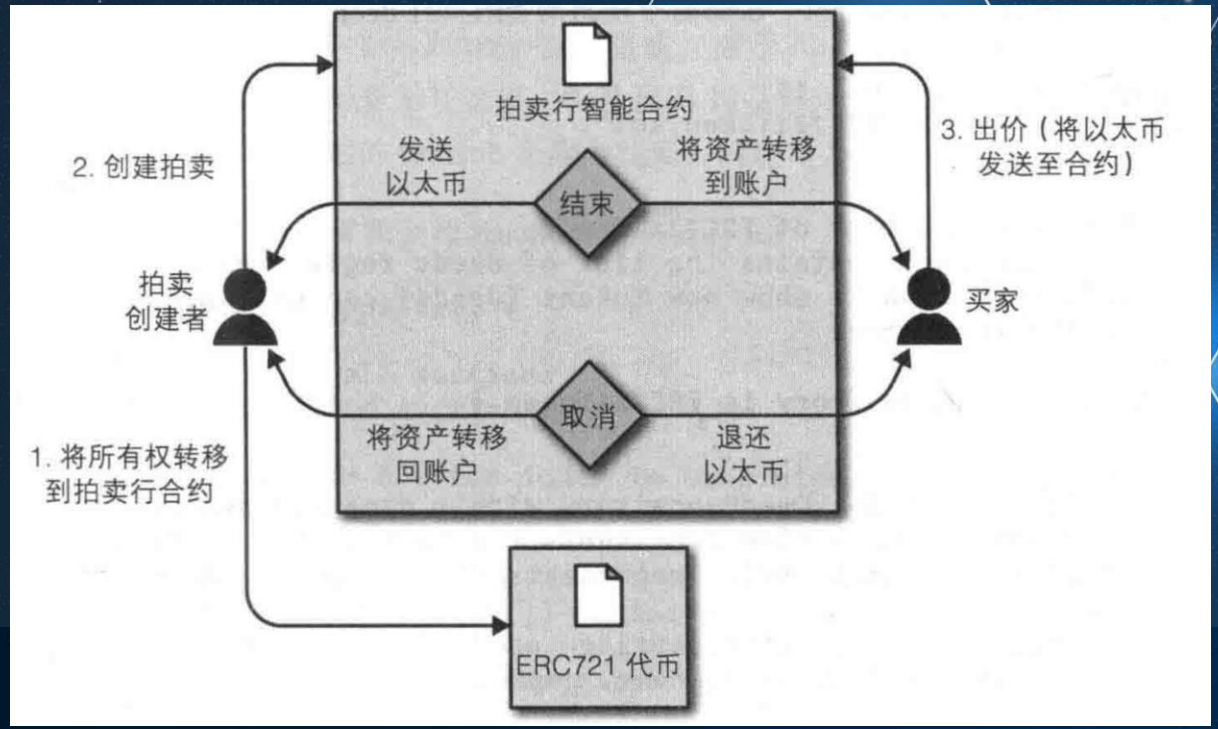
DApp—名称服务

以太坊名称服务（ENS）用一种去中心化的方式允许用户在浏览器中使用人类可读的名称，并可以将其解析为对应的IP地址或者其他标志符。

ENS不仅仅是一个智能合约，它本身是一个基础的DApp，提供去中心化的名称。此外，ENS在名称的注册、管理和拍卖方面受到了很多DApp的支持。ENS展示了多个DApp之间是如何协同工作的：它是为其他DApp服务的DApp，受到DApp的生态系统的支持，嵌入到其他的DApp中。

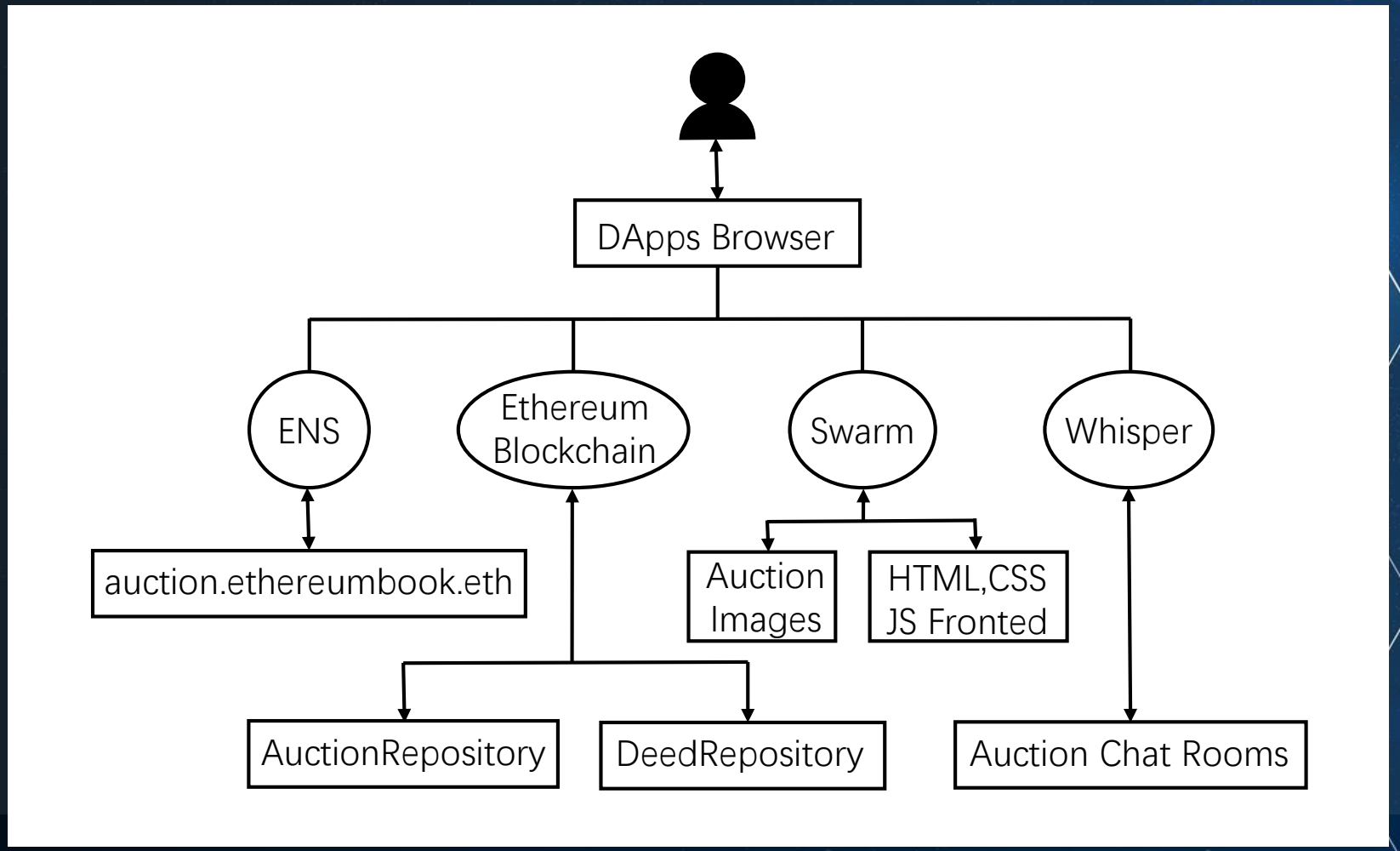
9.2 DApp示例：拍卖DApp

- 拍卖DApp允许用户登记一个“deed”代币，这个代币代表着一些特定的资产，例如房屋、汽车和商标等，一旦代币被登记到拍卖DApp，代币的所有权就被转移到拍卖DApp中，允许它被“挂牌待售”。拍卖DApp列出每一个已经登记的代币，并允许其他用户出价购买。在每次拍卖期间，用户可以加入专门为这次拍卖创建的聊天室。一旦拍卖完成，deed代币的所有权将会被转移给拍卖的获胜者（对应着实物所有权的转移）。



DApp示例：拍卖DApp

拍卖DApp架构



后端业务逻辑：

- DeedRepository实现不可替代性“deed”代币的智能合约
- AuctionRepository实现拍卖的智能合约

前端界面：

- 使用Vue/Vuetify JS框架的前端

数据存储：

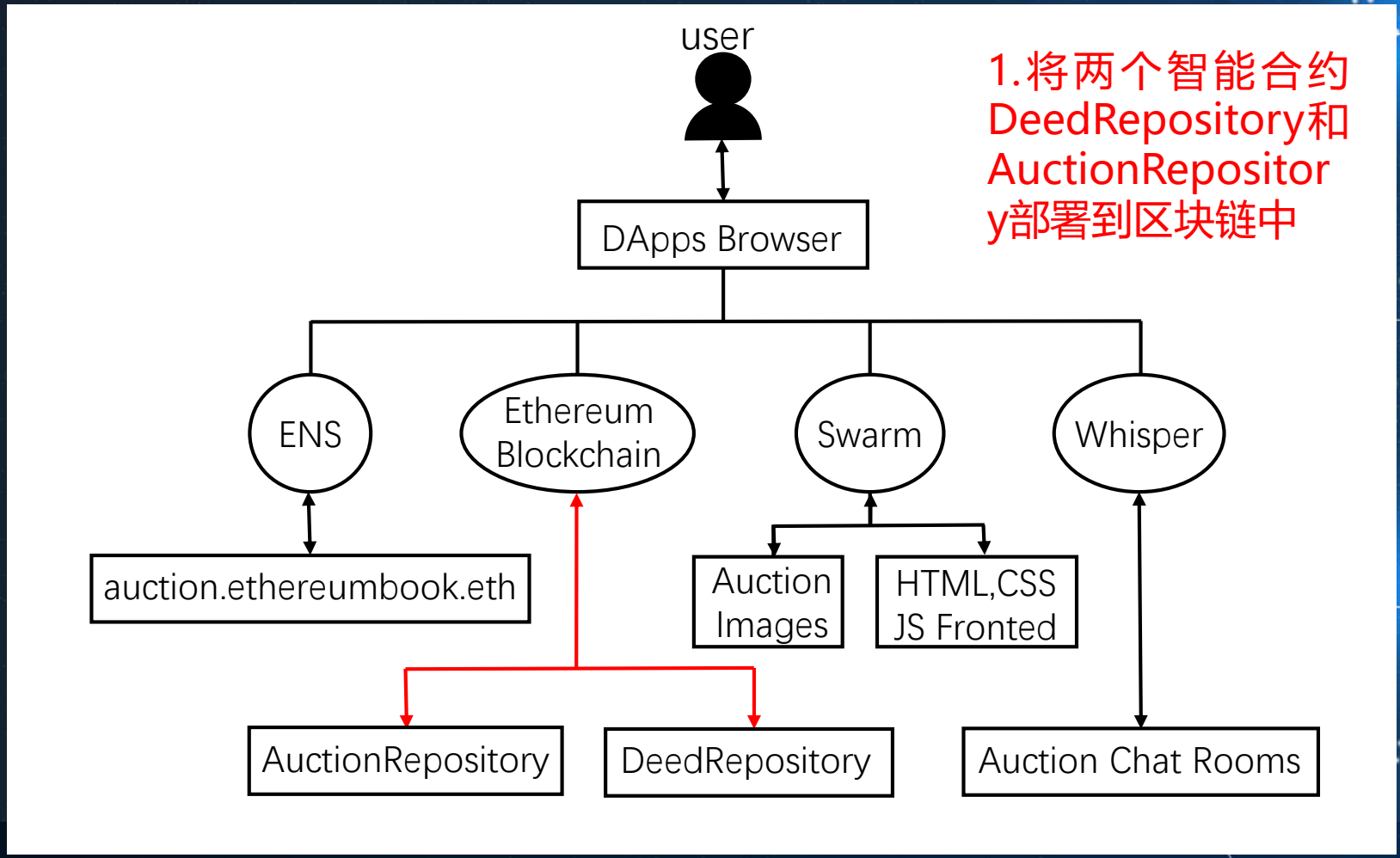
- Swarm客户端存储图片等资源

消息通讯：

- Whisper客户端为所有参与者就每一笔拍卖创建一个聊天室

DApp示例：拍卖DApp

拍卖DApp后端业务逻辑实现



DApp示例：拍卖DApp

- 后端DeedRepository智能合约

代码 12-1：DeedRepository.sol：在拍卖中使用的 ERC721 deed 代币

```
pragma solidity ^0.4.17;
import "./ERC721/ERC721Token.sol";

/**
 * @title Repository of ERC721 Deeds
 * This contract contains the list of deeds registered by users.
 * This is a demo to show how tokens (deeds) can be minted and added
 * to the repository.
 */
contract DeedRepository is ERC721Token {

    /**
    * @dev Created a DeedRepository with a name and symbol
    * @param _name string represents the name of the repository
    * @param _symbol string represents the symbol of the repository
    */
    function DeedRepository(string _name, string _symbol)
        public ERC721Token(_name, _symbol) {}

    /**
    * @dev Public function to register a new deed
    * @dev Call the ERC721Token minter
    * @param _tokenId uint256 represents a specific deed
    * @param _uri string containing metadata/uri
    */
    function registerDeed(uint256 _tokenId, string _uri) public {
        _mint(msg.sender, _tokenId);
        addDeedMetadata(_tokenId, _uri);
        emit DeedRegistered(msg.sender, _tokenId);
    }

    /**
    * @dev Public function to add metadata to a deed
    * @param _tokenId represents a specific deed
    * @param _uri text which describes the characteristics of a given deed
    * @return whether the deed metadata was added to the repository
    */
    function addDeedMetadata(uint256 _tokenId, string _uri) public
        returns(bool){
        _setTokenURI(_tokenId, _uri);
        return true;
    }

    /**
    * @dev Event is triggered if deed/token is registered
    * @param _by address of the registrar
    * @param _tokenId uint256 represents a specific deed
    */
    event DeedRegistered(address _by, uint256 _tokenId);
}
```

拍卖DApp使用DeedRepository合约来为每次拍卖登记和跟踪相应的代币。

DApp示例：拍卖DApp

- 后端AuctionRepository智能合约

代码 12-2：拍卖 DApp 的智能合约 AuctionRepository.sol

```
contract AuctionRepository {

    // Array with all auctions
    Auction[] public auctions;

    // Mapping from auction index to user bids
    mapping(uint256 => Bid[]) public auctionBids;

    // Mapping from owner to a list of owned auctions
    mapping(address => uint[]) public auctionOwner;

    // Bid struct to hold bidder and amount
    struct Bid {
        address from;
        uint256 amount;
    }

    // Auction struct which holds all the required info
    struct Auction {
        string name;
        uint256 blockDeadline;
        uint256 startPrice;
        string metadata;
        uint256 deedId;
        address deedRepositoryAddress;
        address owner;
        bool active;
        bool finalized;
    }
```

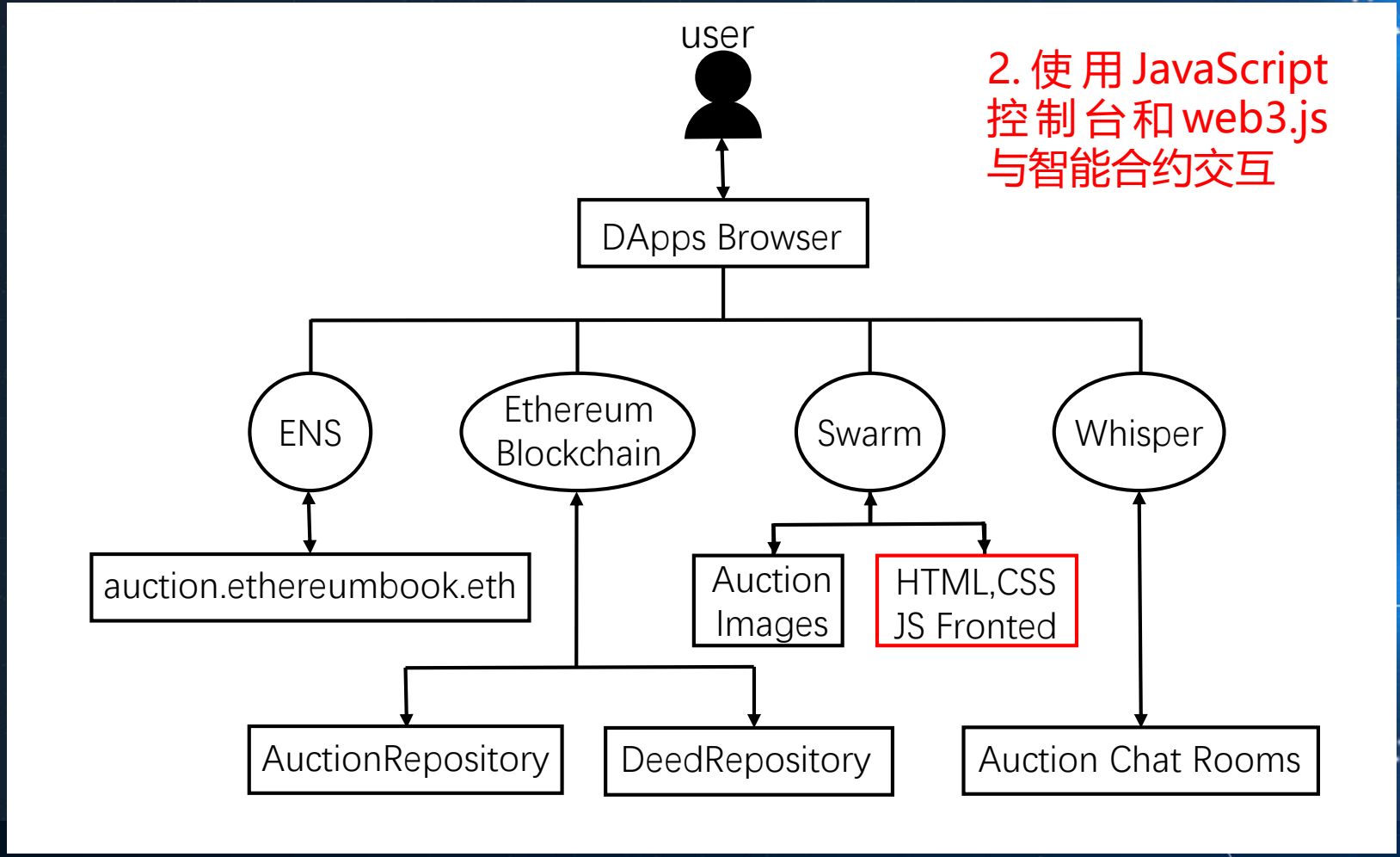
AuctionRepository 合约通过以下函数管理所有的拍卖：

```
getCount()
getBidsCount(uint _auctionId)
getAuctionsOf(address _owner)
getCurrentBid(uint _auctionId)
getAuctionsCountOfOwner(address _owner)
getAuctionById(uint _auctionId)
createAuction(address _deedRepositoryAddress, uint256 _deedId,
              string _auctionTitle, string _metadata, uint256 _startPrice,
              uint _blockDeadline)
approveAndTransfer(address _from, address _to, address _deedRepository-
    Address,
                   uint256 _deedId)
cancelAuction(uint _auctionId)
finalizeAuction(uint _auctionId)
bidOnAuction(uint _auctionId)
```

完整的代码的网址为https://bit.ly/2IaOo9i。

DApp示例：拍卖DApp

拍卖DApp前端实现



2. 使用JavaScript控制台和web3.js与智能合约交互

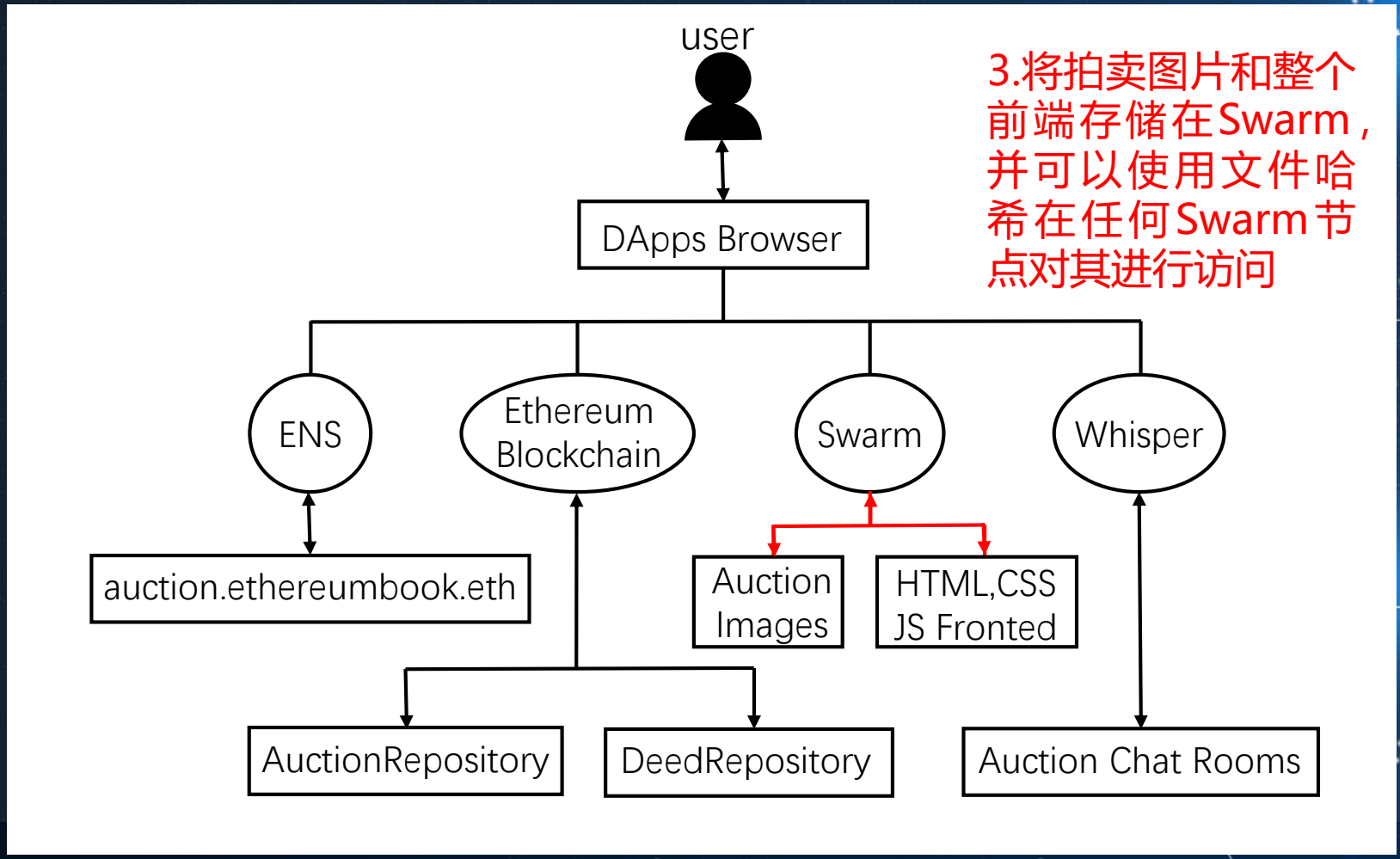
DApp示例：拍卖DApp

拍卖DApp前端界面展示

user

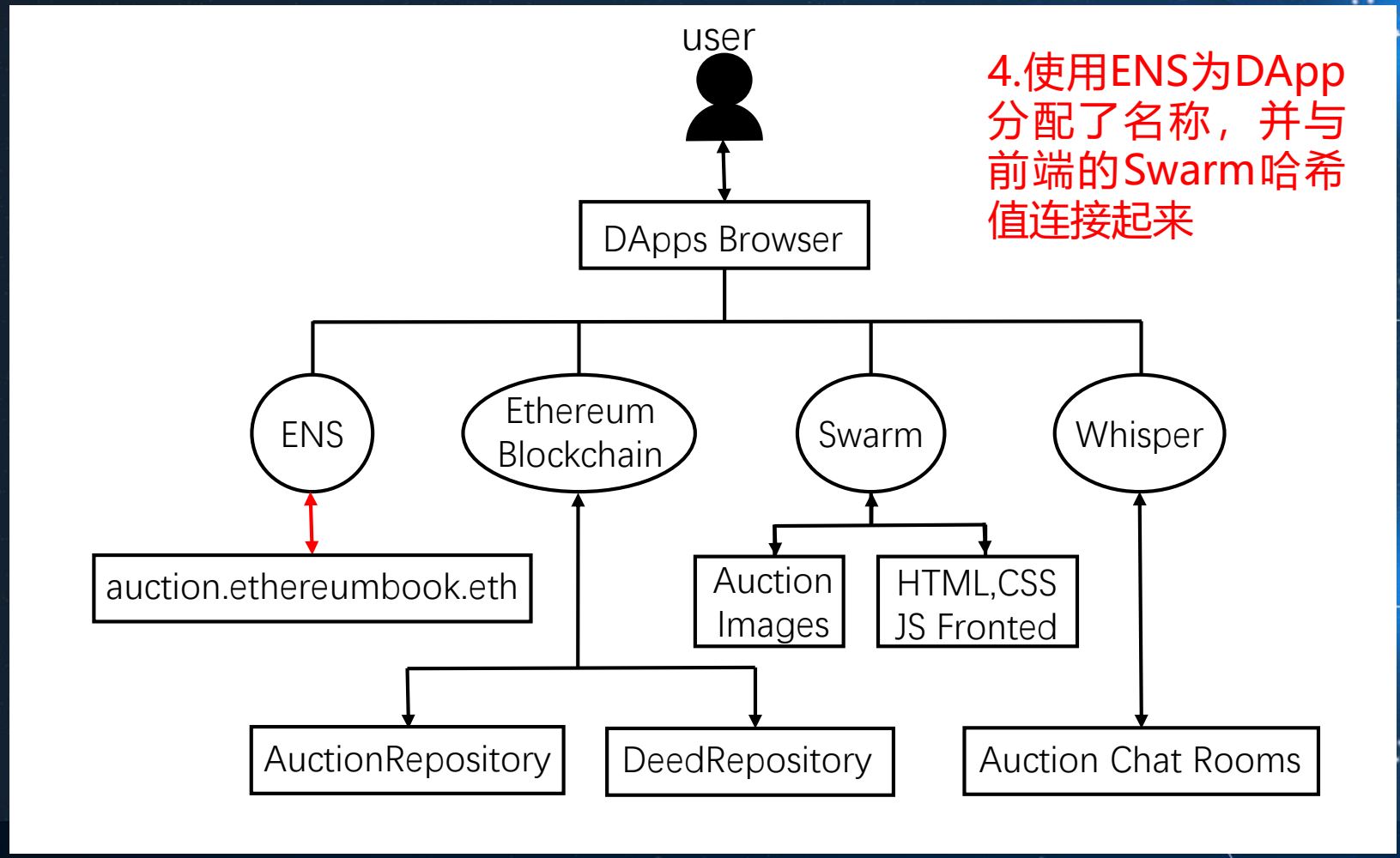
DApp示例：拍卖DApp

拍卖DApp数据存储：去中心化的文件存储DApp Swarm



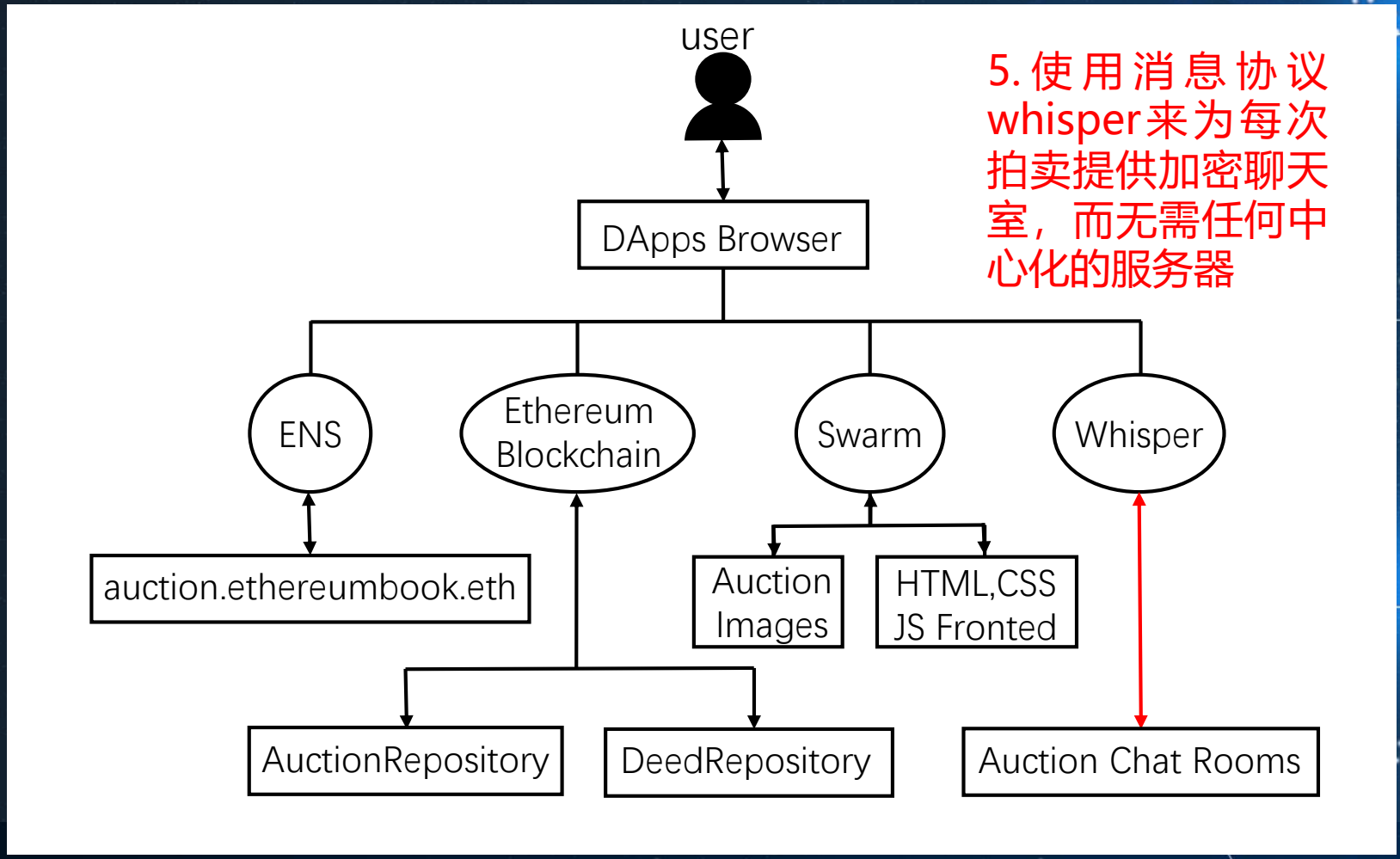
DApp示例：拍卖DApp

拍卖DApp去中心化域名解析



DApp示例：拍卖DApp

拍卖DApp聊天室实现：使用去中心化DApp whisper实现

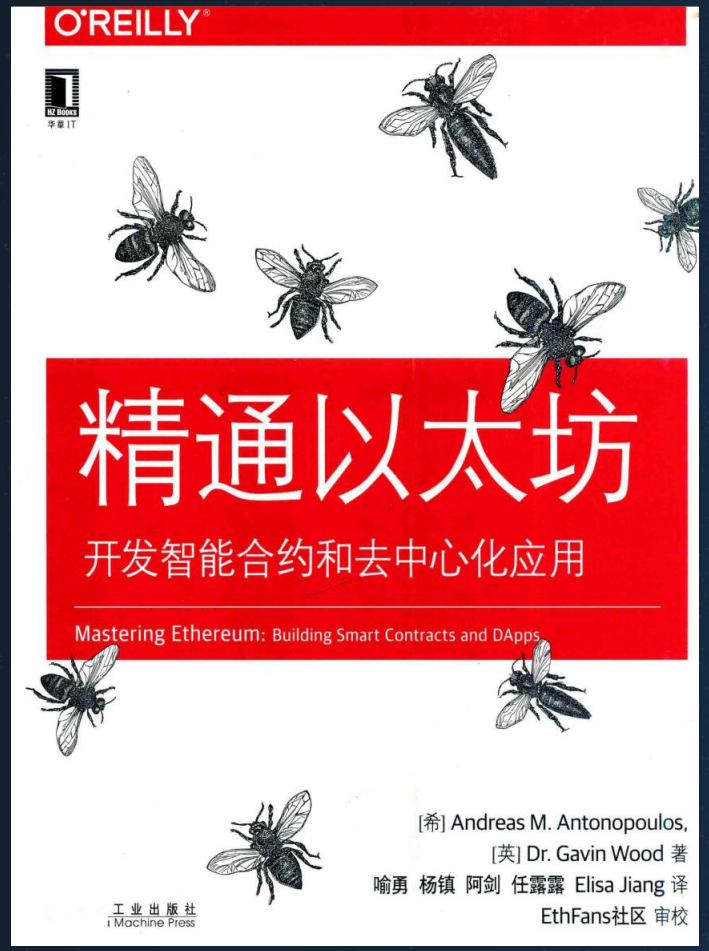


DApp示例：拍卖DApp

- 前面我们逐步构建了一个去中心化应用。我们从一对运行ERC721的拍卖智能合约开始。这两个合约被设计成没有治理或者特权账户，所以它们的运作是真正去中心化的。
- 我们添加了一个用JavaScript实现的前端，它为DApp提供了一个方便且用户友好的界面。
- 拍卖DApp使用去中心化存储系统Swarm来存储应用资源，如图片。我们将整个前端上传到Swarm，这使得DApp不依赖任何Web服务器提供文件。
- DApp还使用去中心化的消息协议Whisper来为每次拍卖提供加密聊天室，而无须任何中心化的服务器。
- 我们使用了ENS系统为DApp分配了一个名称，并将其与前端的Swarm哈希值连接起来， 以便用户以简单易记、人类可读的名称去访问它。

通过这些步骤，我们增加了应用的去中心化程度。最终的结果是这个DApp没有集中式的控制，也没有单点失败的模块，这样的设计充分表达了web3的愿景。

参考书



在随书代码库里可以找到拍卖DApp的源代码，网址为：*https://github.com/ethereumbook/ethereumbook/tree/develop/code/auction_dapp*。

谢谢！