法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

关注 小象学院

区块链编程：Solidity以太坊智能合约

王亮

第五课 合约调用与web3.js

5.2 合约调用与web3 api

智能合约调用

- 以太坊支持通过各种方式与节点进行交互:
- JSON-RPC
- JavaScript Console
- web3

以太坊JSON RPC

- JSON RPC可以理解为一个rest服务。
- 大部分客户端均通过JSON RPC调功能、传数据。
- JSON RPC只是一个传输通道，以太坊还有IPC的接口。

RPC调用客户端命令

- 假设我们要调用客户端命令eth.getBalance(),查询地址为0x407的账号的余额，命令如下：
- curl --data '{"jsonrpc":"2.0","method":"eth_getBalance","params":["0x407", "latest"],"id":1}' localhost:8123
- 其中：jsonrpc字段指定JSON-RPC版本号，method字段指定需要调用的api方法名，params字段为传送的参数，id为消息标识字段；

RPC调用智能合约

- 假设目前有部署的智能合约，地址为0x6ff93，我们要调用的合约方法签名multiply(uint256)，传入的参数值为6，那么调用命令的格式如下：

RPC调用智能合约

```
curl --data
{
"jsonrpc": “2.0”,
"method": “eth_sendtransaction”,
"params"[{
    "from":"0xeb85a5",
    "to":"0x6ff93",
    "data":"0xcdddddd"
}]
"id":8}
localhost:8123
```

其中，from为扣除GAS的账户地址，to为智能合约部署的地址，data为调用方法的签名和传入参数，编码方式为：

“0x”+sha3(“multiply(uint256)”).substring(0,8)+to_32bit_Hex_str(6)

RPC合约调用

- 直接使用RPC对智能合约的调用需要进行复杂的编码。
- 具体规则可以参考Ethereum Contract ABI文档。
- 实际编程中，基本都使用web3等方式，对RPC进行了友好封装。

Web3 概述

- 与合约交互，最常用的方式就是使用web3.js library提供的web3。
- 底层实现上，它通过RPC 调用与本地节点通信。
- web3.js可以与任何暴露了RPC接口的以太坊节点连接。
- Web3已随truffle安装。

Web3调用合约

- web3.js封装了合约调用的方法。
- 使用可以直接使用web3.eth.contract的里的sendTransaction来修改区块链数据。
- 调用合约，可能需要from等参数，否则可能出现调用异常。

Web3 API体系

- Web3-eth: 以太坊区块链基本操作和智能合约相关操作
- Web3-ssh: 实现whisper相关操作，包括p2p和广播操作
- Web3-bzz：swarm协议相关，分布式存储
- web3-utils：Dapp开发辅助功能

Web3 版本说明

- 本课使用的web3为>1.0的版本
- npm ls web3，使用该命令查看版本
- npm update web3, 如果版本过低，请升级
- 版本文档以此为准:
  https://web3js.readthedocs.io/en/1.0/web3-eth.html

Web3初始化

```
// 引入web3
// in node.js use: var Web3 = require('web3');

// 如果浏览器按照了MetaMask，则会提供一个默认的web3.currentProvider
// 如果为空，则连接远程/本地节点
var web3 = new Web3(Web3.givenProvider || "ws://localhost:8546");
```

Web3 常用API

- web3.setProvider, 设置Provider
- 参数：无
- 返回值：无
- 示例：web3.setProvider(new web3.providers.HttpProvider('http://localhost:8545'));

Web3 常用API

- **web3.toWei**
- 按对应货币转为以wei为单位。最常用的单位为ether。
- 例子：**var value** = web3.toWei('1', 'ether');
  console.log(**value**); *// "1000000000000000000"*

Web3 常用API

- **web3.eth.account**
- 以太坊账号
- 例子：web3.eth.getAccounts([callback])
- web3.eth.getBalance(address [, defaultBlock] [, callback])

Web3 常用API

- **web3.eth.contract**，创建一个Solidity的合约对象，用来在某个地址上初始化合约。
- 参数：Array - 一到多个描述合约的函数，事件的ABI对象。
- 返回值：Object - 一个合约对象。
- 例子：**var** MyContract = new web3.eth.Contract(abiArray);

Web3 常用API

- 合约对象的方法
- 显示调用call。 myContract.methods.myMethod([param1[, param2[, ...]]]).call(options[, callback]); (不修改数据)
- 显式调用send。 myContract.methods.myMethod([param1[, param2[, ...]]]).send(options[, callback]) ; (修改数据， 消耗gas)

Web3 常用API

合约调用方法：

```
//合约实例
var contract = new web3.eth.Contract(abi,address);

//callback
contract.methods.helloWorld().call(function(error, result){
    console.log(result);
});

//promise
contract.methods.helloWorld().call().then(
    function(result){
        console.log(result);
    }
);
```

Web3 常用API

- 合约对象的事件
- 参数：
- Object - 你想返回的索引值（过滤哪些日志）。如，{'valueA': 1, 'valueB': [myFirstAddress, mySecondAddress]}。默认情况下，所以有过滤项被设置为null。意味着默认匹配的是合约所有的日志。
- Object - 附加的过滤选项。参见web3.eth.filter的第一个参数。默认情况下，这个对象会设置address为当前合约地址，同时第一个主题为事件的签名。
- Function -（可选）传入一个回调函数，将立即开始监听，这样就不用主动调

Web3 常用API

- 合约对象的事件，回调返回值：
- Object - 事件对象，如下：
- address: String，32字节 - 日志产生的合约地址。
- args: Object - 事件的参数。
- blockHash: String，32字节 - 日志所在块的哈希。如果是pending的日志，则为null。
- blockNumber: Number - 日志所在块的块号。如果是pending的日志，则为null。

Web3 常用API

- 合约对象的事件，回调返回值：
- logIndex: Number - 日志在区块中的序号。如果是pending的日志，则为null。
- event: String - 事件名称。
- removed: bool - 标识产生事件的这个交易是否被移除（因为孤块），或从未生效（被拒绝的交易）。
- transactionIndex: Number - 产生日志的交易在区块中的序号。如果是pending的日志，则为null。
- transactionHash: String，32字节 - 产生日志的交易哈希值。

流程演示

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：**小象学院**