链滴

以太坊 JavaScript API web3.js 打币

作者：88250

原文链接：https://ld246.com/article/1520583695227

来源网站：链滴

ABI 简介

关于什么是 ABI（Application Binary Interface）请看官网文档。简单来说 ABI 就是外界和合约交互方式：

- ABI 用 JSON 描述合约的接口定义
- 用约定好的编码方式进行实际调用

做过 RPC 的同学可以理解为这就是异构平台的 RPC 实现，需要进行接口定义、存根生成、寻址、序化、网络通讯等步骤就能调用到合约上的函数了。

ABI 的实例请看这里。

web3.js 简介

web3.js 是以太坊 JavaScript API 的官方实现，用它来和以太坊节点进行可编程通讯，基于 HTTP 或 PC 连接。简单来说 web3.js 就是可以通过 JavaScript 调用本地或远程的以太坊节点，完成网络管理账户管理、交易等操作。

下面的代码示例基于 web3.js v1.0 开发，请注意不同版本之间的差异。

同步区块

人们都说以太坊发币最大的技术难点就是同步区块，一点儿没错！

试了 ETHFANS 星火计划提供的静态节点也不行，最后还是放弃本地电脑同步，主要问题是我家这里网络质量太差，很多时候藕断丝连。

后来在阿里云**海外节点**开了台服务器几个小时就同步好了（国内节点我也试过，一晚上没同步好，没

心就放弃了），下面给出服务器配置大家可以参考下：

- 内存至少 4G，否则 geth 进程会被 OOM Killer 干掉或者各种崩溃
- 磁盘至少 100G（区块高度 530W）
- 带宽我开的是 5M

同步使用官方 geth 就行，启动不用带参。每次下载最新的 geth 后最好浏览一下帮助，避免用了过的参数而掉坑里。

交易（打币/转账）

前面铺垫完，正式进入今天的主题：如何通过 web3.js 打币。我们大致需要用到如下几个 API（看命就懂的 API 我就不啰嗦了）：

- personal.unlockAccount：调用后等待用户输入对应解锁账户的私钥密码完成解锁
- eth.sendTransaction：发送交易到以太坊网络，返回交易哈希
- eth.contract：合约接口

先启动 geth 的交互式可执行环境：geth attach（Windows 上面可能需要用 geth attach ipc:\\.\pip\geth.ipc），然后分两种打币情况：以太坊、ERC20 令牌。

以太坊转账

这种情况是最简单的，只需两步：

1. 解锁源账户
2. 执行转账

```
personal.unlockAccount(fromAddr); // 输入正确的私钥密码后将允许进行交易操作
eth.sendTransaction({from: fromAddr, to: toAddr, value: web3.toWei(0.1, "ether")}); // 转账 0.1 eth
```

ERC20 令牌转账

1. 构造 ERC20 令牌合约操作对象
2. 解锁源账户
3. 调用 ERC20 令牌转账接口执行转账

```
var tokenContractABI = [....] // JSON ABI
var contractAddr = "0xe249e7a6f5a9efee03b4c5090c77245ef6fe0f5e";
var tokenContract = eth.contract(tokenContractABI).at(contractAddr); // 构造合约对象

var fromAddr = "0xb3d201b5963db83c434e5810b54ac62e3ee05c00"; // 源账户地址
tokenContract.balanceOf(fromAddr); // 查看令牌余额
var amount = 20000;  // 转账金额
var toAddr = "0x.."; // 目标账户地址

personal.unlockAccount(fromAddr); // 输入正确的私钥密码后将允许进行交易操作
tokenContract.transfer(toAddr, web3.toWei(amount), {from: fromAddr}); // 执行令牌转账
```

事件处理

主要使用场景是发送交易后等待网络确认，然后通知外部程序做相应的处理；也用于和合约定义的事进行交互。

使用回调

```
eth.sendTransaction({
    from: '0x..',
    to: '0x..',
    value: '10000'
}, function(error, hash){
    ...
});
```

使用 Promise

```
eth.sendTransaction({
    from: '0x..',
    to: '0x..',
    value: '10000'
}).then(function(receipt){
    ...
});
```

使用监听

```
eth.sendTransaction({
    from: '0x..',
    to: '0x..',
    value: '10000'
}).on('transactionHash', function(hash) {
    ...
}).on('receipt', function(receipt) {
    ...
}).on('confirmation', function(confirmationNumber, receipt) {
    ...
}).on('error', console.error);
```

总结

web3.js 是以太坊的 JavaScript API 实现，作为交互式可执行环境被集成在了 以太坊节点 geth 中，可以单独引入 JavaScript 应用中进行使用。它提供了通过 JS 和以太坊网络、智能合约交互的方式，箱即用。对于新手来说，学习以太坊开发的最大难点是区块同步，当然也可以使用测试网络或者自建网来进行开发调试。

关于更多以太坊、区块链开发的干货，请浏览黑客派社区以太坊、区块链。