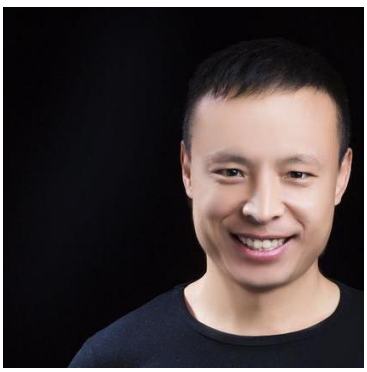
上手 DApp 开发（V0.1）

0–概述

大家好，最近几个月我一直在我的知乎账户 https://www.zhihu.com/people/peterlovemoney/activities 下发布区块链和 DApp 开发的文章。2018年10月21日，本人很荣幸的受到 Nervos 北京社区的邀请，去太库做一场闭门的 Nervos 讲师培训。于是我就把我最近的写作内容，按照《上手 DApp 开发》这条主线，梳理编排了一下，于是有了这个 PDF 。个人水平有限，本文档的内容难免疏漏比较多，欢迎大家关注我的知乎账户，以及《Nervos 学习站》https://learning.nervos.org ，获得更新的内容。

作者介绍




我是 Peter ，中文名：王广忠。Github 上写了差不多十年开源代码了，一直没有间断过。2014年受李笑来老师邀请，入职云币。有幸跟现在的 Nervos 核心团队，Big.one 团队，Imtoken 团队，比特大陆技术负责人潘志斌都有过或多或少的合作。好我个人的情况就不多扯了，喜欢八卦的同学欢迎访问我的博客：http://happypeter.github.io/ 。

重点我想说说我目前在做的事情。我给自己定的职业目标是”网红，专业区块链讲解员“。具体来说就是不断生产内容，生产”最易懂的严肃内容“。所以过去几个月来，我一直坚持每天发一篇文章。

内容组织形式

内容上是刻意安排了先后循序的一篇篇的可以独立分享的文章组成。内容组织思路我们从两个维度说。

首先拿出单篇文章来看。我的风格是这样，每篇文章瞄准一个话题。例如标题是《隔离见证》，那我就全力讲隔离见证，文章做到有主线，有中心，所有的内容都紧扣中心，有论点，

有论据。总之，就是要把概念讲的简单好懂。因为直接上手看文档，有些内容的确不好懂。Peter 要做的就是把难懂的文章转换成好懂的教程。

第二个维度，就是从当前这个 PDF 的章节编排上。我会努力梳理出从一个普通 App 开发者成长为一个 DApp 开发者的学习路径来。比如，我们要先学点比特币入门吧，然后区块链深入的知识要学一些吧，密码学基础总还是要会的吧。等等这些，哪些要先学哪些后学，我整体的理解的一个学习路线图大家都可以从本文档的章节编排上看出来。

当然，整体的内容咱们会不断改版。

内容概要

总体上内容不是以具体的 DApp 代码为主，因为 DApp 技术还在迅速迭代中，所以具体代码讲解 Peter 都会放到独立的小一些的视频课程中，类似 https://learning.nervos.org/ 上的这些课程。所以咱们这里的内容，相对偏基础一些。

首先我们要理解为何要开发 DApp ，意义何在。所以我们会聊聊比特币背后的故事，密码朋克运动，区块链和智能合约基本哲学，什么是加密经济等等。

然后重头戏还是比较纯技术的。我们会以 Nervos 整套技术栈为例，讲解分层架构相关的技术点。首先是密码学的基础，然后介绍侧链技术，SPV ，隔离见证等等这些理解分层架构所必须的区块链基础知识。

总结

好，这些就是 PDF 内容的一个概要介绍了。

私钥写死到代码里面显然不是一个 DApp 应该有的状态，本节就来把账户信息从源码中移除。这样 DApp 运行在浏览器中就不能签名交易了，也就不能发送留言了。解决方式就是让 DApp 运行在 AppChain 官方的 Neuron 钱包中。

Neuron 基本原理

所以我们先介绍一下 Neuron 的基本工作原理。

首先 Neuron 是一个加密货币钱包，所以传统上一个加密货币钱包应该有的基本功能 Neuron 也是有的。具体来讲就是为用户管理私钥，用户可以把自己的各种加密货币导入钱包中，通过钱包进行转账操作。嗯，没错，钱包的基本作用就是用来进行转账，而转账的实质就是用私钥去签署交易，这样交易就会得到有效的授权，转账也就成功了。

但是 Neuron 不是一个传统钱包，而是一个 DApp 钱包。Neuron 可以理解成一个浏览器，我们用 Web 技术开发的 DApp 是直接可以运行在 Neuron 里面的，注意，不需要浏览器，只要在 Neuron 打开 DApp 就可以。当然 Neuron

要比浏览器的功能多，其中最重要的依然是 Neuron 中可以完成签名交易的功能。跟 DApp 互动的时候，签名交易的目的都是为了操作 DApp 的智能合约，而不是普通意义上的用户 A 给用户 B 进行转账了。

更多关于 Neuron 的介绍可以参考官方文档：https://docs.nervos.org/neuron-android/#/ 。下面我们操作的的大思路是这样：把 DApp 中的用户私钥信息从代码中移除，当 DApp 需要跟智能合约交互的时候，让 Neuron 去负责签名。

代码调整

更多的细节我们跟着代码一起来聊。

```
cp dapp1 dapp2
```

把上一节中的 dapp1 文件夹拷贝一份，命名为 dapp2 ，代码调整我们都在 dapp2 中去完成，也就是说 dapp2 中最终会变成一个可以跟 Neuron 互动的 DApp 。

首先到 transaction.js 中

```
-const nervos = require('./nervos')
const transaction = {
- from: nervos.appchain.accounts.wallet[0].address,
- privateKey: nervos.appchain.accounts.wallet[0].privateKey,
```

其他内容不动，删除三行文件。首先删除对 nervos 的导入。然后删除 from 和 privateKey 两项，也就是账户信息删除掉，因为这些信息应该由 Neuron 来提供了。

nerovs.js

```
const { default: Nervos } = require('@nervos/chain')

const config = require('./config')
if (typeof window.nervos !== 'undefined') {
  window.nervos = Nervos(window.nervos.currentProvider)
  window.nervos.currentProvider.setHost('https://node.cryptape.com')
} else {
  console.log('No Nervos web3? You should consider trying Neuron!')
  // fallback - use your fallback strategy (local node / hosted node + in-dapp id mgmt / fail)
  window.nervos = Nervos(config.chain)
}
var nervos = window.nervos

module.exports = nervos
```

再到 nervos.js 文件中国，首先导入 Nervos ，然后导入配置。因为 Neuron 是 Nervos AppChain 专用的钱包，所以只要在 Neuron 中打开项目， window.nervos 这个全局变量默认就是存在的。所以 if 条件肯定是满足的。再看 if

成立的代码块中这两条语句，由于 Neuron 是支持多链的，所以这里要明确指明一下，当前 DApp 要去交互的是那条链。这里 node.cryptape.com 对应的就是测试链，或者准确的说是测试链上的一个节点。因为区块链一定是一个网络，网络上的节点数量一般都不是一个。这里要注意的是保证设置的链就是我们部署合约的链。else 代码块中的语句是为了应对 DApp 不运行在 Neuron 中的情况，我们可以忽略这部分。另外补充一点，到底要使用那条链，也可以在 manifest.json 文件中去配置，具体细节可以参考官方案例：https://github.com/cryptape/nervos-appchain-docs/blob/develop/zh-CN/quick-start/build-dapp.md#配置-manifestjson ，我们这里也忽略。

config.js

```
- chain: 'http://121.196.200.225:1337 ',
- privateKey:
-  '0x5d3c73fa94c85bbcb516cb256a4e82c414255a270b5d065179a94aa0d5dc3efe',
```

config.js 中要删除两项内容。一项是 chain ，参考官方文档，https://docs.nervos.org/nervos-appchain-docs/#/quick-start/deploy-appchain ，可以看到这里这个链接跟上面我们在 nervos.js 中设置的 node.cryptape.com ，都是同一测试网节点。所以这里就不用重复指定了。要删除的第二项是 privateKey ，因为签署交易用的是 Neuron 钱包中的账户。最终 config.js 保留的就是一个合约地址了。

App.js

```
   const tx = {
      ...simpleStore.transaction,
+     from: window.neuron.getAccount(),
      validUntilBlock: +current + 88
    }

  componentDidMount = async () => {
-  const from =
-     nervos.appchain.accounts.wallet[0] &&
-     nervos.appchain.accounts.wallet[0].address
+   const from = window.neuron.getAccount()
```

App.js 中也要稍作调整。交易的发起方，不再是从本地代码中去获取了，而是通过 window.neuron.getAccount() 去从钱包中获取。

至此，代码调整咱们就完成了，内容比较多，大家手动跟着我改比较容易改错，可以直接下载 Github 上我存放的代码： https://github.com/happypeter/NervFirst/tree/master/dapp2 。

到钱包中测试

下面来实际的测试一下。

```
yarn start
```

首先把 react 项目启动起来，比如启动到了 localhost:3000 。

```
ifconfig|grep 192
```

查看一下我们开发机的本地局域网 IP ，比如我这里是 192.168.1.108 。

下面来找一部 Andriod 手机，安装 Neuron 。然后按照界面上的步骤，去创建测试网账户，并且也去申请一些测试网的 NOS 代币。过程都是比较直白的，咱们这里就不演示了。接下来就输入 192.168.1.108:3000 ，在 Neuron 中打开 DApp 。

可以随意输入一些文字，然后点提交。接下来会弹出的几个界面，但是总体要做的其实就是一件事，就是用钱包私钥签署一下交易。因为交易过程中虽然转账金额为0，但是还是要有手续费的，是要花 Gas 的。

交易成功后，到 https://microscope.cryptape.com/ ，输入我们的合约地址搜一下，可以看到界面上显示出合约账户下又多了一个交易，可见刚才的提交操作成功了。为了保持课程的简单，我们这里没有做出一套完整的 UI 体验。

在 Github 仓库的 complete 文件夹下，我托管了官方的一个流程完整的代码镜像：https://github.com/happypeter/NervFirst/tree/master/complete/first_forever 。

总结

本节的内容就是这些了，我们把一个运行在浏览器中的 DApp ，改成了一个运行在 Neuron 钱包中的 DApp 。首先聊了 Neuron 作为一个 DApp 钱包的核心作用，那就是可以跟 DApp 配合，完成交易签名。接下来调整了代码，把代码中关于用户账户的信息全部删除掉，最后在 Neuron 中验证了，果然可以用 Neuron 中的私钥去成功签署交易。至此，我们的课程的核心内容也就介绍完毕了。