**构建和部署去中心化投票Dapp**

课程目标

1. 了解区块链智能合约
2. 学会搭建智能合约开发环境
3. 学会如何编译智能合约
4. 学会如何将智能合约部署到区块链
5. 学会如何通过WebApp和智能合约尽心互动
6. 掌握DApp（去中心化App）的整个开发部署流程

7掌握去中心化在实战产品中应用的重大意义

编辑器选择

理论上讲任何编辑器都可以编写Solidity合约代码，比如：WebStorm，VSCode，Sublime，等等。我选择的是Atom，没有任何理由，因为Atom轻量并且界面漂亮。

移步<https://atom.io/>地址，下载安装Atom。

autocomplete-solidity代码自动补齐

linter-solium、linter-solidity代码错误检查

language-ethereum支持Solidity代码高亮以及Solidity代码片段

安装所需工具

首先开发机上必须装好Node.js，再使用以下命令安装所需的工具：

$ npm install -g ethereumjs-testrpc truffle

liyuechun:~ yuechunli$ npm install -g ethereumjs-testrpc truffle

/usr/local/bin/testrpc -> /usr/local/lib/node\_modules/ethereumjs-testrpc/build/cli.node.js

/usr/local/bin/truffle -> /usr/local/lib/node\_modules/truffle/build/cli.bundled.js

+ truffle@3.4.9

+ ethereumjs-testrpc@4.1.3

added 1 package and updated 7 packages in 76.132s

liyuechun:~ yuechunli$

创建项目

/Users/liyuechun/Desktop/1012/Voting

liyuechun:Voting yuechunli$ ls

liyuechun:Voting yuechunli$ pwd

/Users/liyuechun/Desktop/1012/Voting

liyuechun:Voting yuechunli$ truffle unbox react-box

Downloading...

Unpacking...

Setting up...

Unbox successful. Sweet!

Commands:

Compile: truffle compile

Migrate: truffle migrate

Test contracts: truffle test

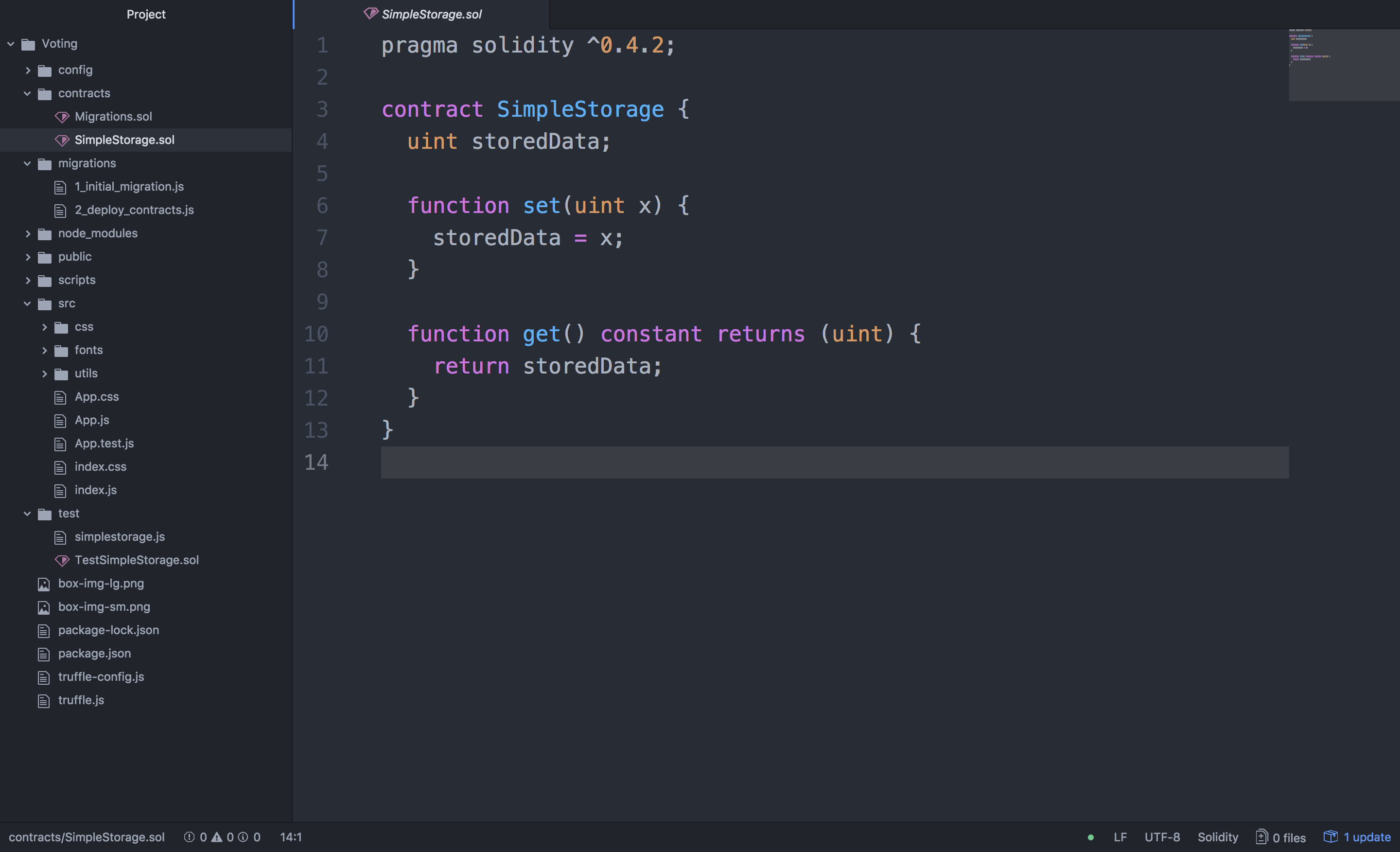
Test dapp: npm test

Run dev server: npm run start

Build for production: npm run build

liyuechun:Voting yuechunli$

项目结构



contracts：编写智能合约的文件夹，所有的智能合约文件都放置在这里

migrations：部署合约配置的文件夹

src：基于React的Web端源码

test：智能合约测试用例文件夹

编写投票Dapp智能合约

在contracts文件夹下创建Voting.sol文件，将下面的代码拷贝到文件中。

pragma solidity ^0.4.4;

contract Voting {

// liyuechun -> 10

// xietingfeng -> 5

// liudehua -> 20

mapping (bytes32 => uint8) public votesReceived;

// 存储候选人名字的数组

bytes32[] public candidateList;

// 构造函数 初始化候选人名单

function Voting(bytes32[] candidateNames) {

candidateList = candidateNames;

}

// 查询某个候选人的总票数

function totalVotesFor(bytes32 candidate) constant returns (uint8) {

require(validCandidate(candidate) == true);

// 或者

// assert(validCandidate(candidate) == true);

return votesReceived[candidate];

}

// 为某个候选人投票

function voteForCandidate(bytes32 candidate) {

assert(validCandidate(candidate) == true);

votesReceived[candidate] += 1;

}

// 检索投票的姓名是不是候选人的名字

function validCandidate(bytes32 candidate) constant returns (bool) {

for(uint i = 0; i < candidateList.length; i++) {

if (candidateList[i] == candidate) {

return true;

}

}

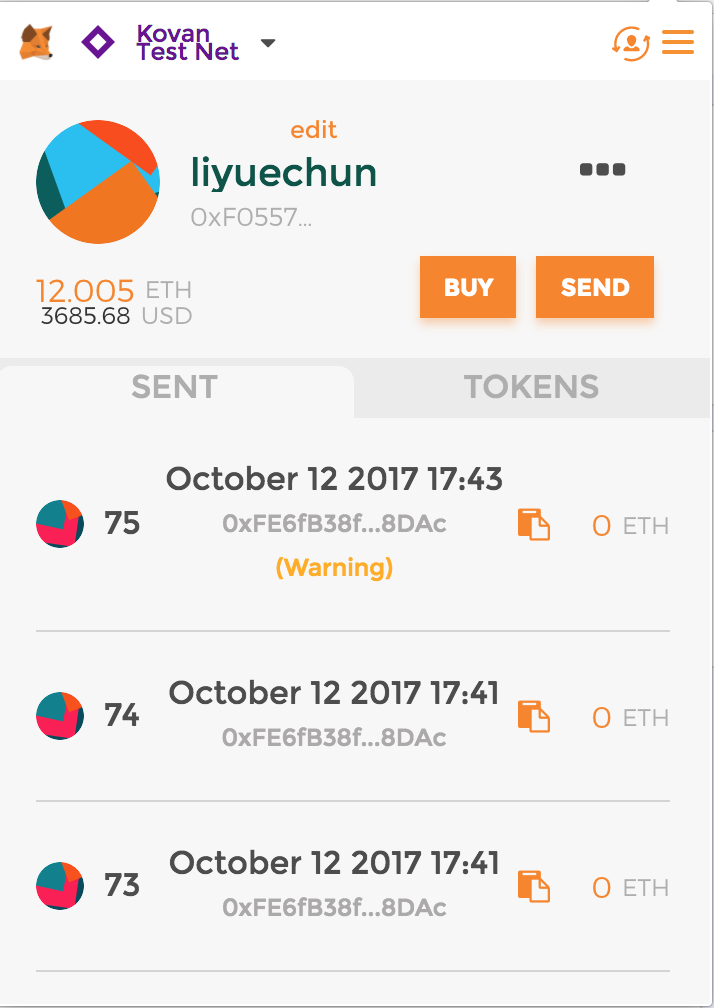
return false;

}

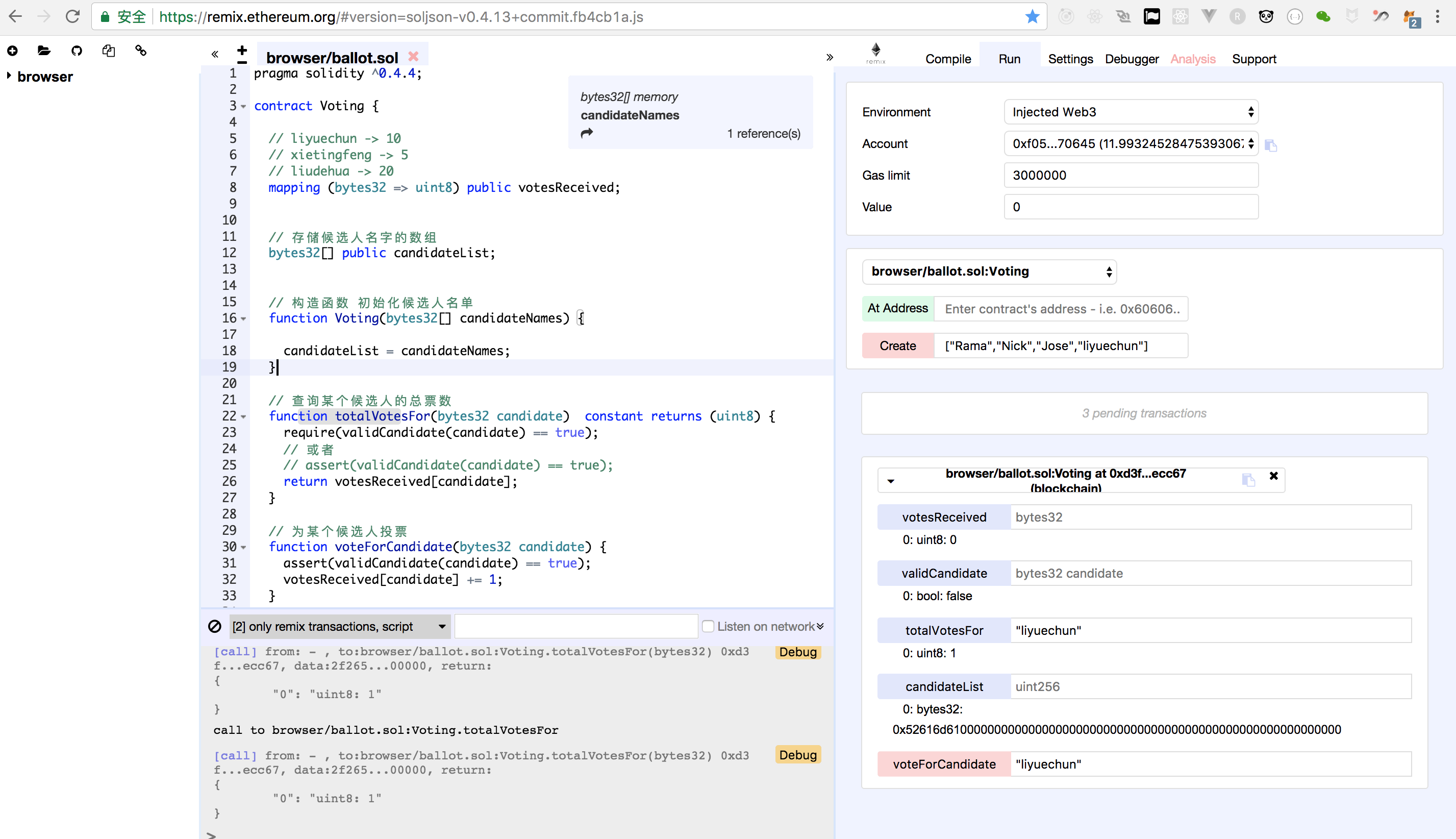
}

通过remix + metamask部署合约到Kovan Test Net

在Google浏览器里面安装MetaMask插件



打开[https://remix.ethereum.org](https://remix.ethereum.org/)将合约代码拷贝到里面



确保MetaMask账号处于等于状态，并且有一定的以太币支付给矿工。

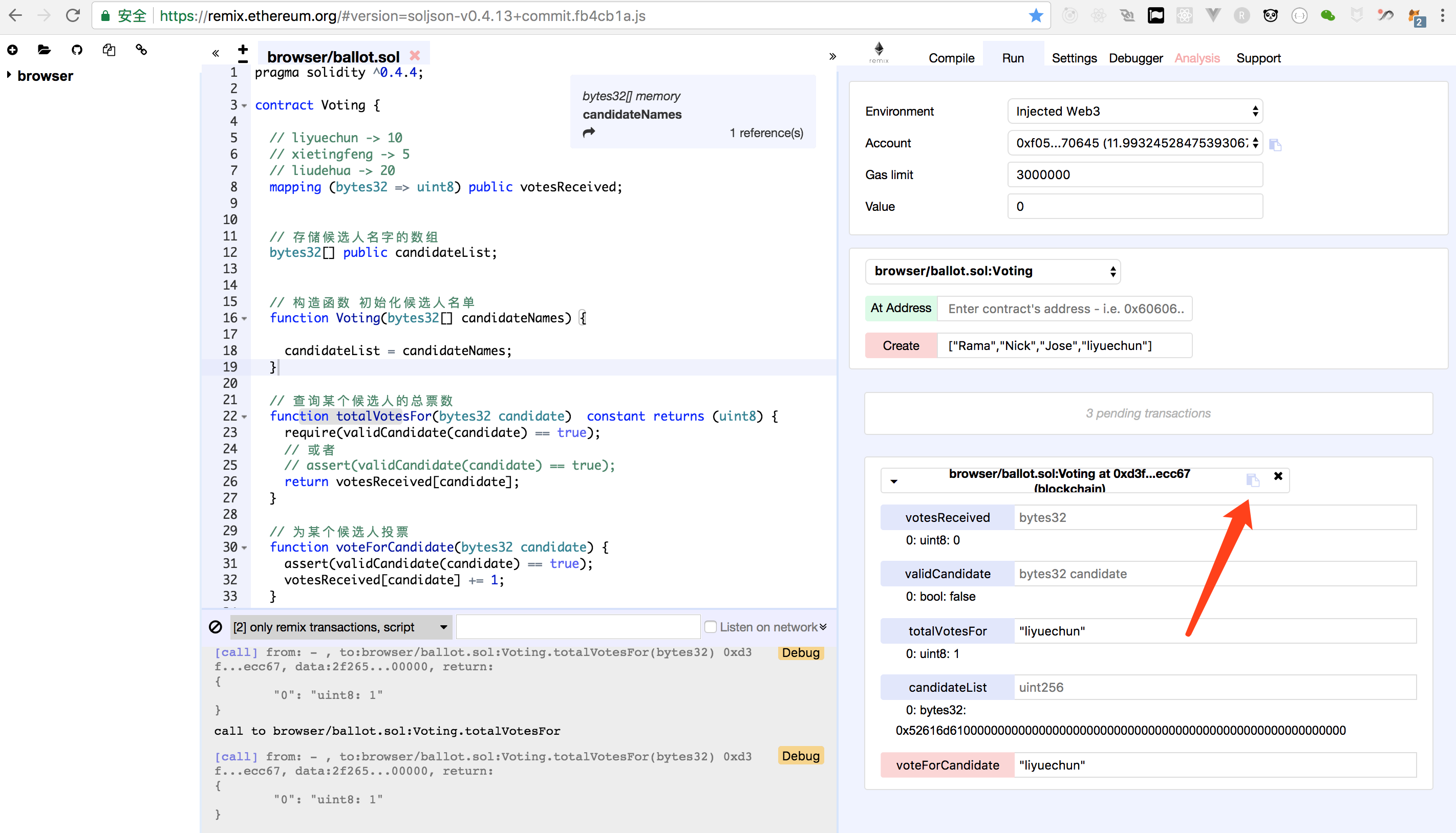
确保Environment是Injected Web3，如果切换不过来，关掉浏览器重新启动

在create函数中输入一个数组，数组里面的内容为**候选人名单**

点击create按钮，会弹出MetaMask界面让你确认，确认提交，过一会儿，合约就部署成功

可以测试给某个候选人投票，查询某个候选人的票数

拷贝合约地址



编译合约

liyuechun:Voting yuechunli$ truffle compile

Compiling ./contracts/Migrations.sol...

Compiling ./contracts/SimpleStorage.sol...

Compiling ./contracts/Voting.sol...

Writing artifacts to ./build/contracts

liyuechun:Voting yuechunli$

编译完合约以后在build/contracts文件夹下面会有一个Voting.json的abi文件。

查看Voting.json文件内容

{

"contract\_name": "Voting",

"abi": [

{

"constant": true,

"inputs": [

{

"name": "candidate",

"type": "bytes32"

}

],

"name": "totalVotesFor",

"outputs": [

{

"name": "",

"type": "uint8"

}

],

"payable": false,

"type": "function"

},

{

"constant": true,

"inputs": [

{

"name": "candidate",

"type": "bytes32"

}

],

"name": "validCandidate",

"outputs": [

{

"name": "",

"type": "bool"

}

],

"payable": false,

"type": "function"

},

{

"constant": true,

"inputs": [

{

"name": "",

"type": "bytes32"

}

],

"name": "votesReceived",

"outputs": [

{

"name": "",

"type": "uint8"

}

],

"payable": false,

"type": "function"

},

{

"constant": true,

"inputs": [

{

"name": "",

"type": "uint256"

}

],

"name": "candidateList",

"outputs": [

{

"name": "",

"type": "bytes32"

}

],

"payable": false,

"type": "function"

},

{

"constant": false,

"inputs": [

{

"name": "candidate",

"type": "bytes32"

}

],

"name": "voteForCandidate",

"outputs": [],

"payable": false,

"type": "function"

},

{

"inputs": [

{

"name": "candidateNames",

"type": "bytes32[]"

}

],

"payable": false,

"type": "constructor"

}

],

"unlinked\_binary": "",

"networks": {},

"schema\_version": "0.0.5",

"updated\_at": 1507806214330

}

这个文件是编译后的**abi**文件，待会儿需要将这个文件的**json**导入到**App.json**中。

查看src/utils/getWeb3.js文件内容

import Web3 from 'web3'

let getWeb3 = new Promise(function(resolve, reject) {

// Wait for loading completion to avoid race conditions with web3 injection timing.

window.addEventListener('load', function() {

var results

var web3 = window.web3

// Checking if Web3 has been injected by the browser (Mist/MetaMask)

if (typeof web3 !== 'undefined') {

// Use Mist/MetaMask's provider.

web3 = new Web3(web3.currentProvider)

results = {

web3: web3

}

console.log('Injected web3 detected.');

resolve(results)

} else {

// Fallback to localhost if no web3 injection.

var provider = new Web3.providers.HttpProvider('http://localhost:8545')

web3 = new Web3(provider)

results = {

web3: web3

}

console.log('No web3 instance injected, using Local web3.');

resolve(results)

}

})

})

export default getWeb3

这个文件主要是封装了一个getWeb3的promiss供我们直接使用，可以从getWeb3直接获取到web3对象供App.js文件中使用。

修改app.js前端代码和合约进行互动

import React, { Component } from 'react'

import VotingContract from '../build/contracts/Voting.json'

import getWeb3 from './utils/getWeb3'

import './css/oswald.css'

import './css/open-sans.css'

import './css/pure-min.css'

import './App.css'

const contractAddress = "0xd3f33a2e553b363b432d7f81f721a2a6202ecc67";

var votingContractInstance;

var account;

var \_modifyVotingCount = (candidates,i,votingCount) => {

console.log("---------");

console.log(candidates);

console.log(i);

console.log(votingCount);

let obj = candidates[i];

obj.votingCount = votingCount;

return candidates;

}

class App extends Component {

constructor(props) {

super(props)

this.state = {

candidates: [

{

"name": "Rama",

"id": 100,

"votingCount": 0

},

{

"name": "Nick",

"id": 101,

"votingCount": 0

},

{

"name": "Jose",

"id": 102,

"votingCount": 0

},

{

"name": "liyuechun",

"id": 103,

"votingCount": 0

}

],

candidatesVoteCount: ["0","0","0","0"],

web3: null

}

}

componentWillMount() {

// Get network provider and web3 instance.

// See utils/getWeb3 for more info.

getWeb3

.then(results => {

this.setState({

web3: results.web3

})

// Instantiate contract once web3 provided.

this.instantiateContract()

})

.catch(() => {

console.log('Error finding web3.')

})

}

instantiateContract() {

/\*

\* SMART CONTRACT EXAMPLE

\*

\* Normally these functions would be called in the context of a

\* state management library, but for convenience I've placed them here.

\*/

const contract = require('truffle-contract')

const votingContract = contract(VotingContract)

votingContract.setProvider(this.state.web3.currentProvider)

// Declaring this for later so we can chain functions on SimpleStorage.

// Get accounts.

this.state.web3.eth.getAccounts((error, accounts) => {

votingContract.at(contractAddress).then((instance) => {

account = accounts[0];

votingContractInstance = instance;

for (let i = 0; i < this.state.candidates.length; i++) {

let object = this.state.candidates[i];

console.log(accounts[0]);

console.log(votingContractInstance);

console.log(votingContractInstance.totalVotesFor(object.name));

votingContractInstance.totalVotesFor(object.name).then(result => {

console.log(i);

console.log(result.c[0]);

this.setState({

candidates: \_modifyVotingCount(this.state.candidates,i,result.c[0])

});

});

}

})

})

}

render() {

return (

<div className="App">

<ul>

{

this.state.candidates.map((object) => {

console.log(object);

return (

<li key={object.id}>候选人：{object.name} 支持票数：{object.votingCount}</li>

)

})

}

</ul>

<input

style=

placeholder="请输入候选人姓名..."

ref="candidateInput"

/>

<button style= onClick={() => {

console.log(this.refs.candidateInput);

console.log(this.refs.candidateInput.value);

let candidateName = this.refs.candidateInput.value;

console.log(this.state.web3.eth.accounts[0]);

votingContractInstance.voteForCandidate(candidateName,{from:account}).then((result => {

console.log(result);

console.log(candidateName);

let number = 0;

for(let i = 0; i < this.state.candidates.length; i++) {

let object = this.state.candidates[i];

if (object.name === candidateName) {

number = i;

break;

}

}

votingContractInstance.totalVotesFor(candidateName).then(result => {

this.setState({

candidates: \_modifyVotingCount(this.state.candidates,number,result.c[0])

});

});

}));

}}>Voting</button>

</div>

);

}

}

export default App