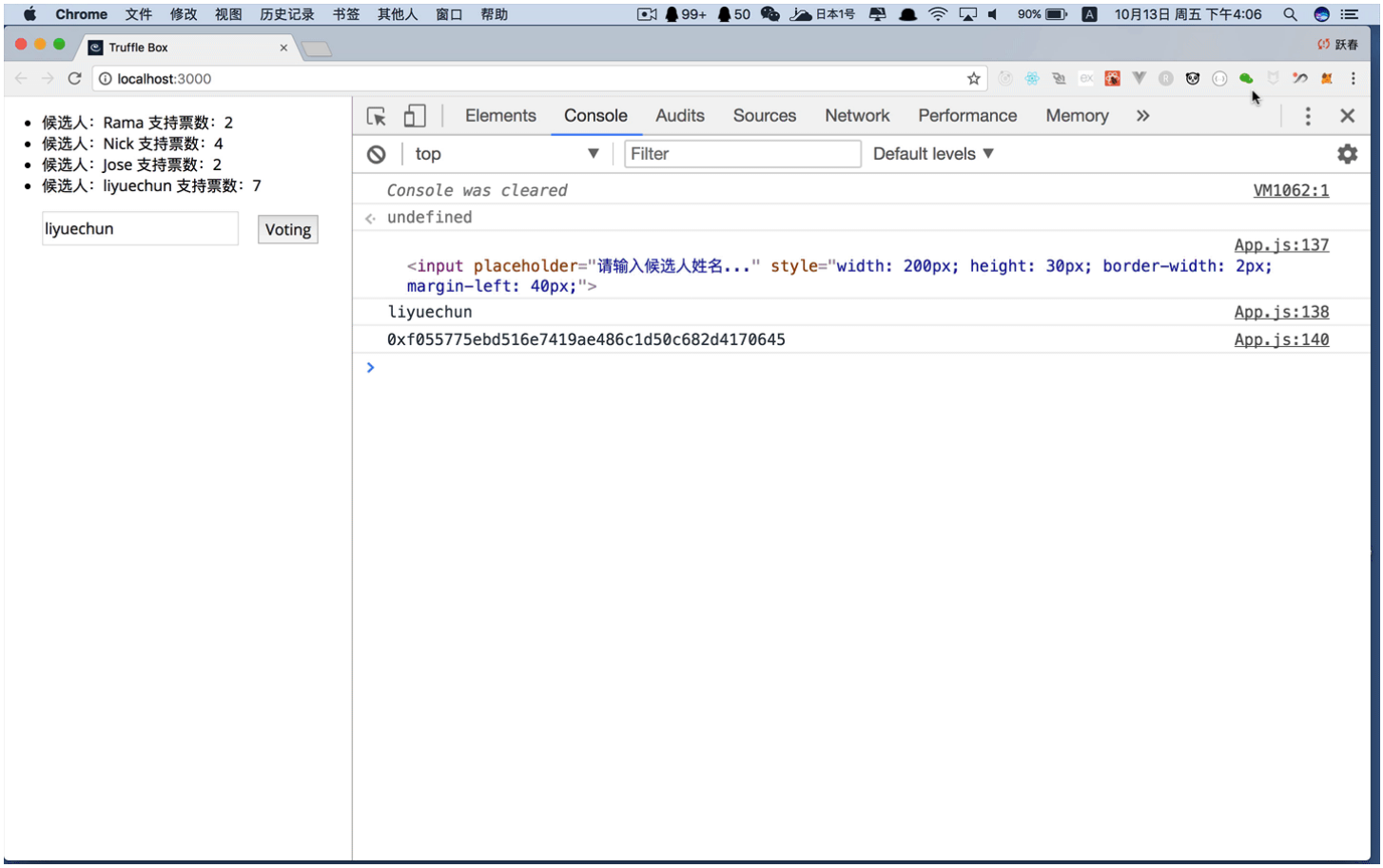
如何编写智能合约（Smart Contract）- 从零构建和部署去中心化投票App，decentralization Voting Dapp

> 孔壹学院：国内区块链职业教育领先品牌
>
> 作者：黎跃春，区块链、高可用架构工程师
> 微信：liyc1215 QQ群：348924182 博客：http://liyuechun.org

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

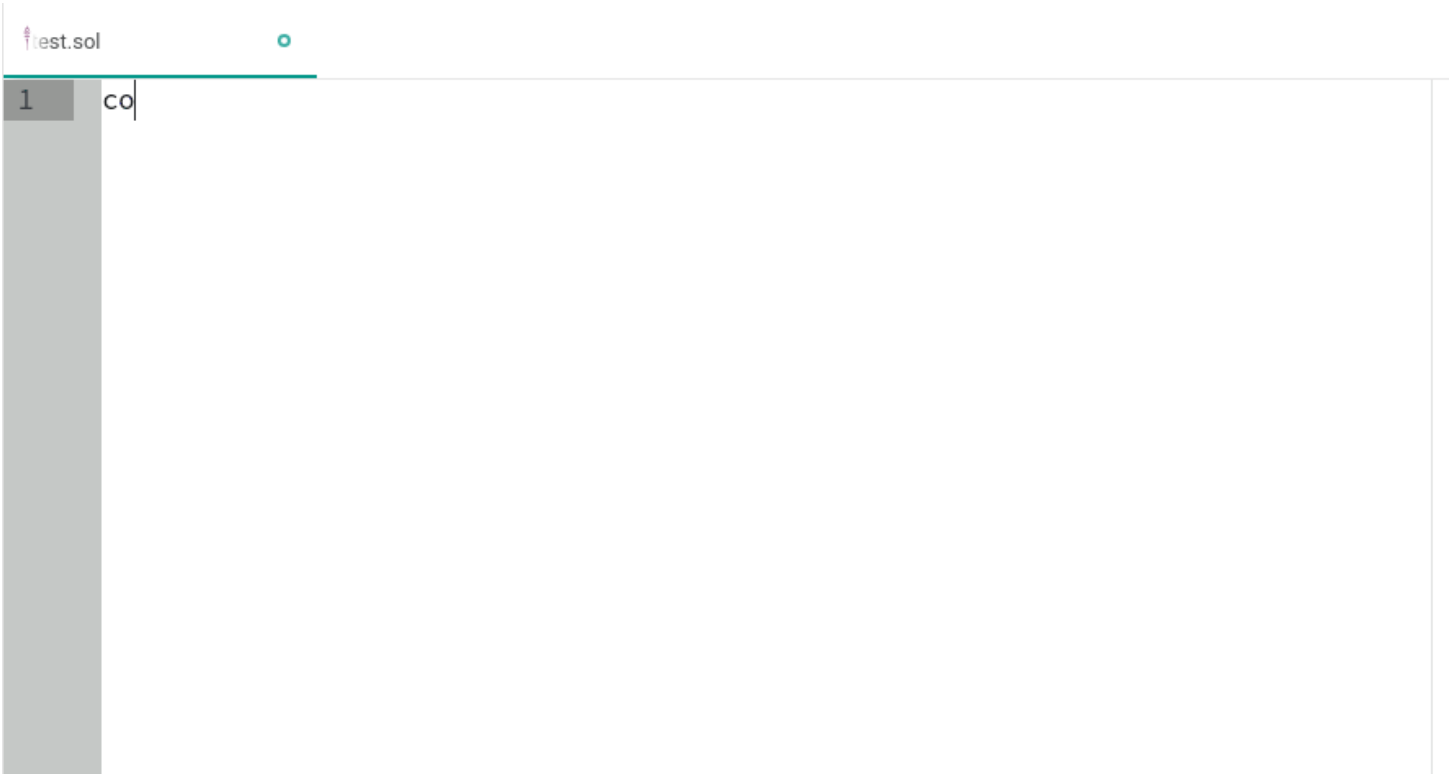
项目效果图



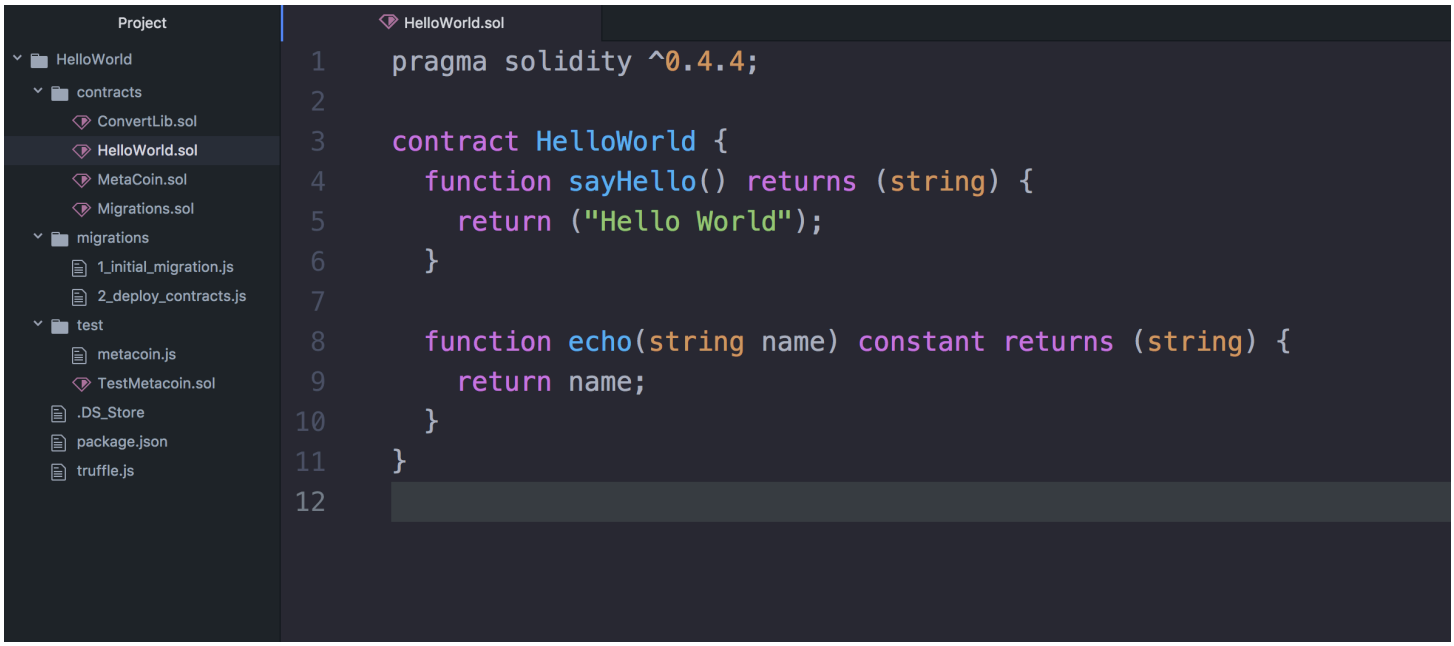
编辑器选择

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：

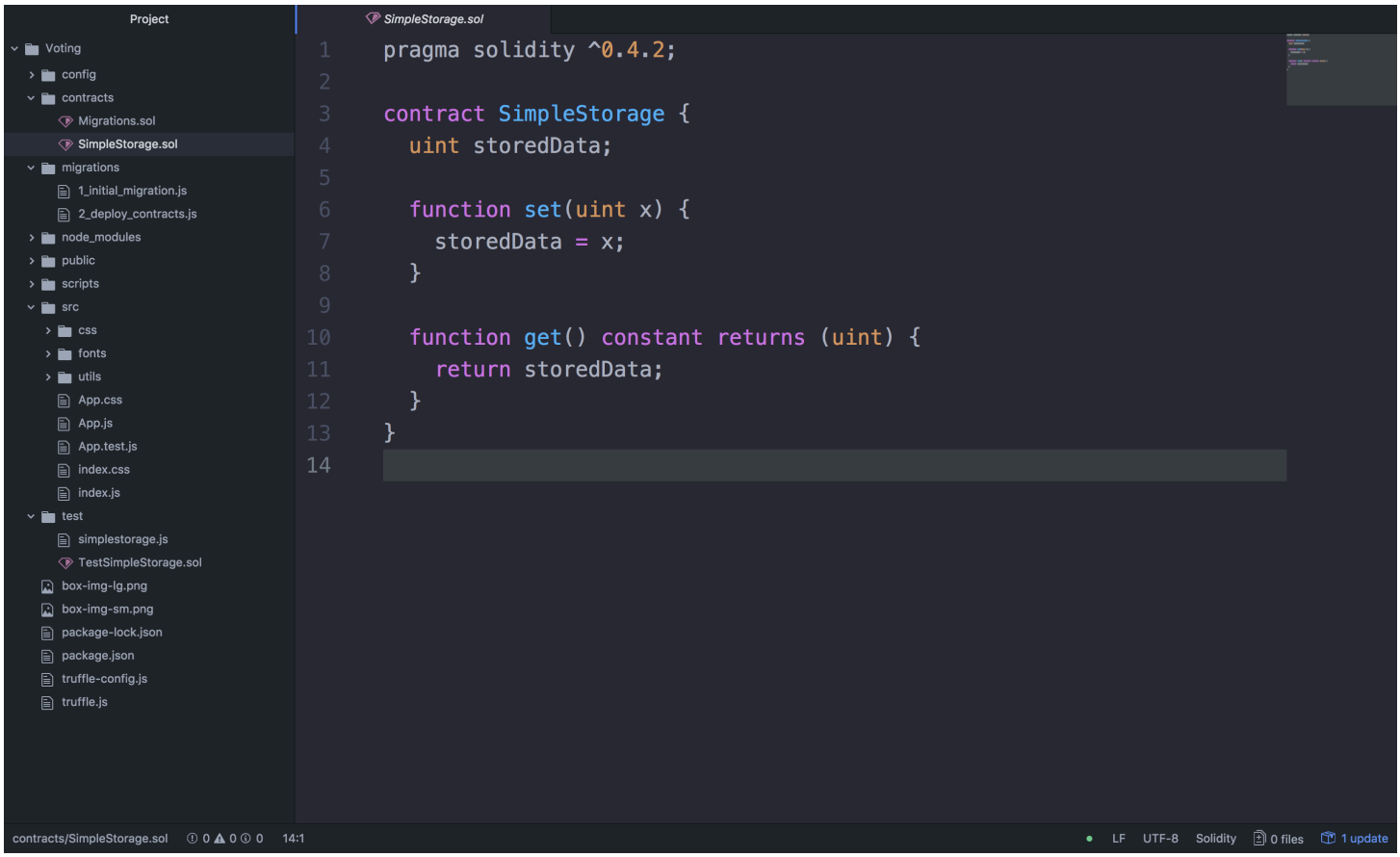
```
$ npm install -g ethereumjs-testrpc truffle
```

```
liyuechun:~ yuechunli$ npm install -g ethereumjs-testrpc truffle
/usr/local/bin/testrpc -> /usr/local/lib/node_modules/ethereumjs-testrpc/bui
ld/cli.node.js
/usr/local/bin/truffle -> /usr/local/lib/node_modules/truffle/build/cli.bund
led.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

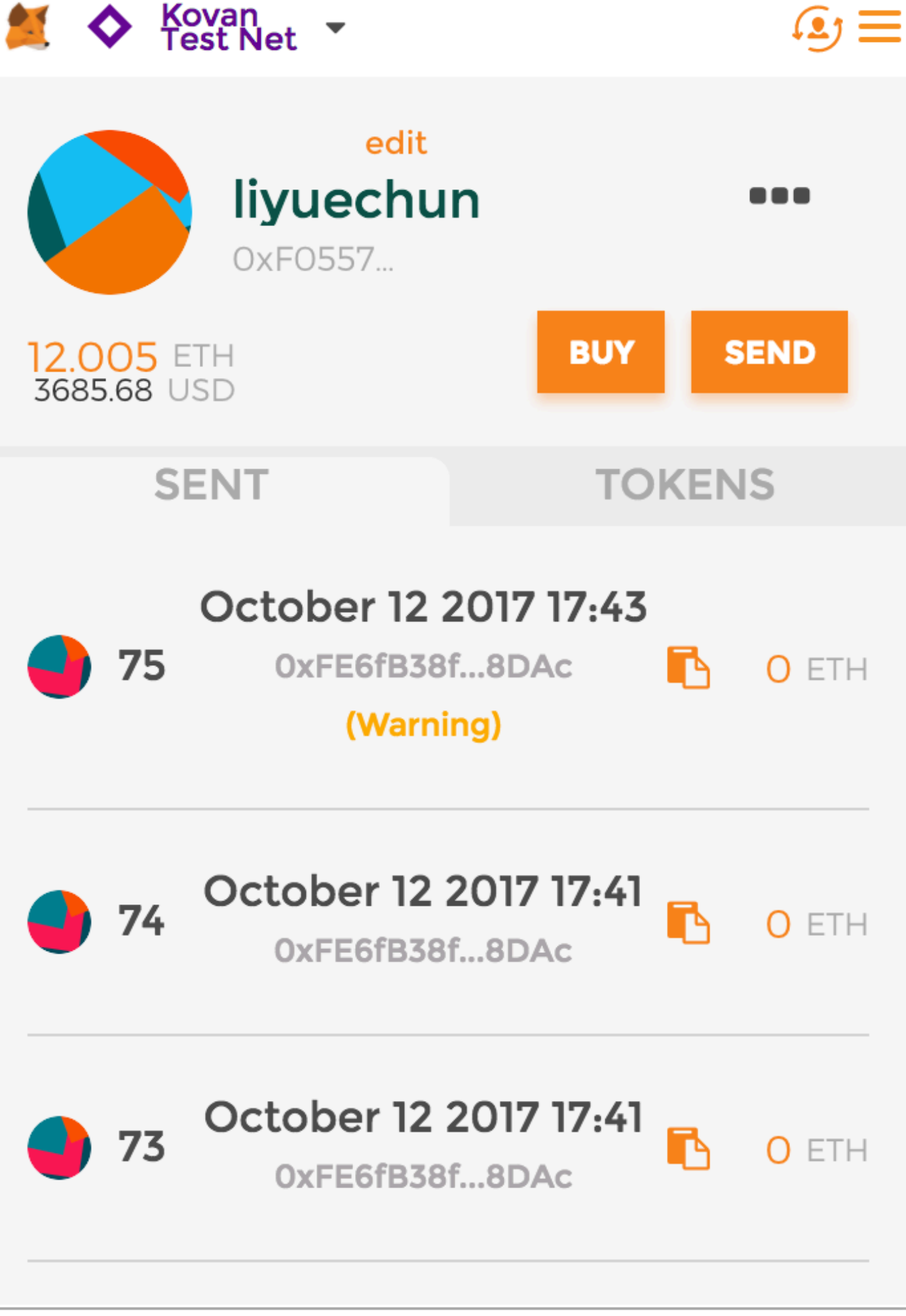
在`contracts`文件夹下创建`Voting.sol`文件，将下面的代码拷贝到文件中。

```
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
```

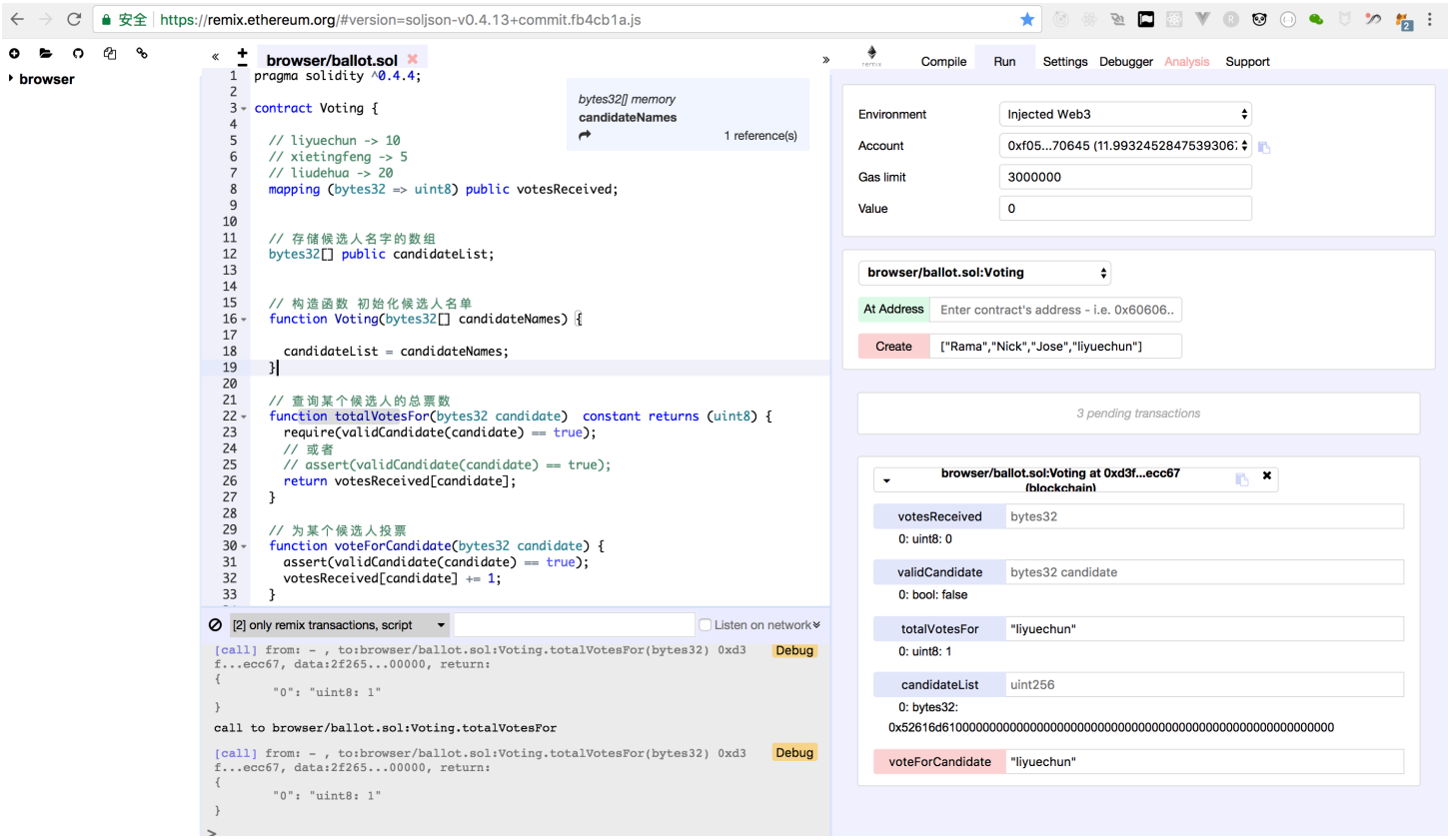
```
  function Voting(bytes32[] candidateNames) {

    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

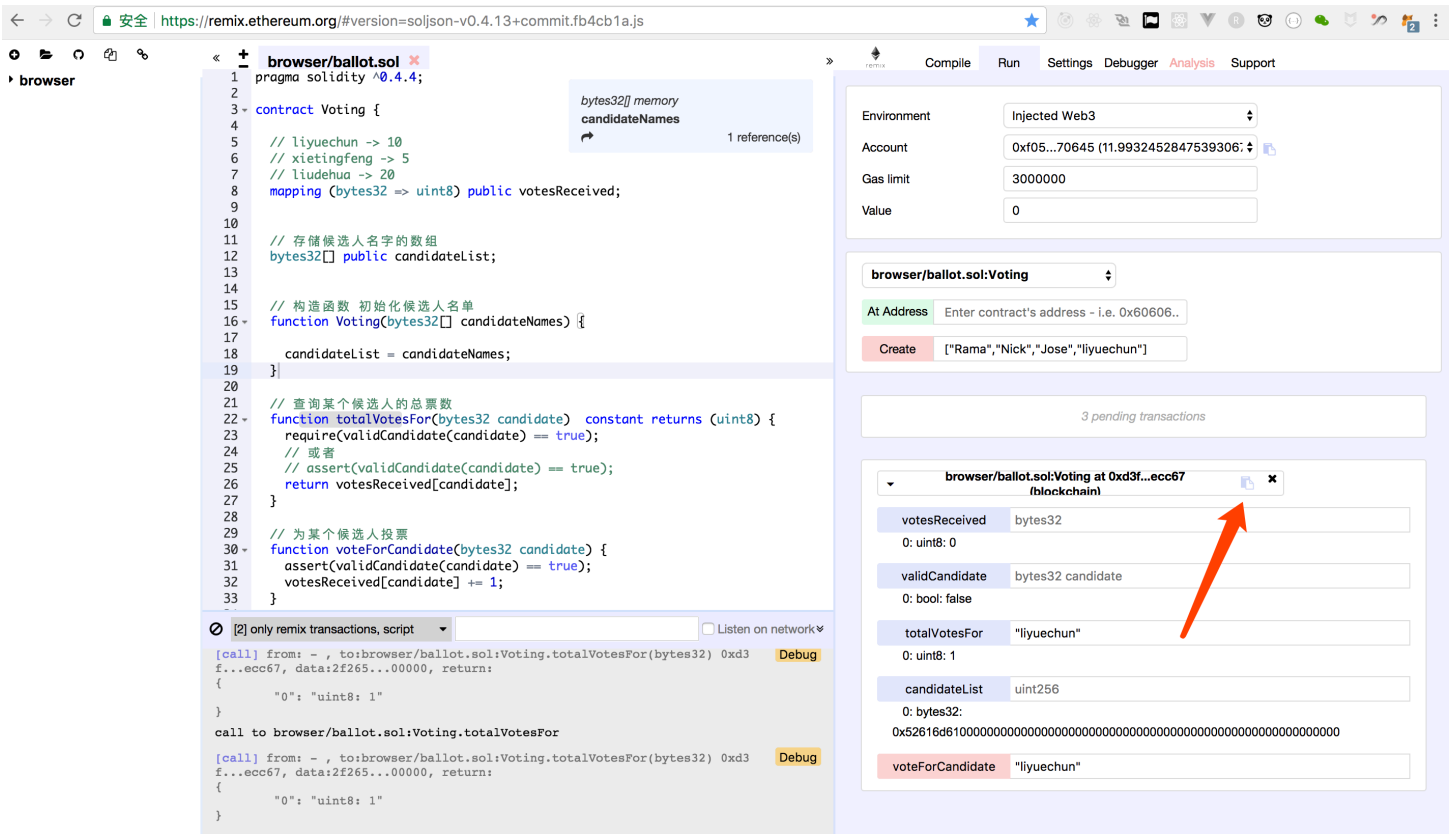

Kovan
Test Net
edit
liyuechun
0xF0557...
12.005 ETH
3685.68 USD
BUY
SEND
SENT
TOKENS
October 12 2017 17:43
75
0xFE6fB38f...8DAc
0 ETH
(Warning)
74
October 12 2017 17:41
0xFE6fB38f...8DAc
0 ETH
73
October 12 2017 17:41
0xFE6fB38f...8DAc
0 ETH

- 打开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




```
0xd3f33a2e553b363b432d7f81f721a2a6202ecc67
```

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": "0x606060405234156100f57600080fd5b6040516103113803806103118339810160405280805190910190505b600181805161003e929160200190610046565b505b506100b5565b828054828255906000526020600020908101928215610083579160200282015
```

```
5b828111156100835782518255602090920191600190910190610066565b5b50610090929150
610094565b5090565b6100b291905b8082111561009057600081556001016100 9a565b509056
5b90565b61024d806100c46000396000f300606060405263ffffffff7c01000000000000000000
0000000000000000000000000000000000000000000060003504166 32f265cf78114610069578063
392e667814610095578063 7021939f146100bf578063b13c744b146100eb578063cc9ab26714
610113575b600080fd5b34156100 7457600080fd5b61007f60043561012b565b60405160ff90
9116815260200160405180910390f35b34156100a057600080fd5b6100ab60043561015d565b
60405190151581526020016040518091039 0f35b34156100ca57600080fd5b61007f60043561
01af565b60405160ff9091168152602001604051809 10390f35b34156100f657600080fd5b61
01016004356101c4565b604051908152602001604051809 10390f35b341561011e57600080fd
5b6101296004356101e7565b005b60006101368261015d565b151560011461014457600080fd
5b506000818152602081905260409020 5460ff165b919050565b6000805b6001548110156101
a4576001805484919083908 11061017c57fe5b906000526020600020900160005b5054141561
019b57600191506101a9565b5b600101610161565b600091505b50919050565b600060208190
52908152604090205460ff1681565b6001805482908 1106101d257fe5b906000526020600020
900160005b5054905081565b6101f08161015d565b151560011461 01fb57fe5b600081815260
208190526040902080546 0ff80821660010116 60ff199091161790555b505600a165627a7a72
3058206783a7ff47eae16f18011a9db2a3cc983350b779bf3181b7623c18d4bce363180029",
  "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 injectio
n 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

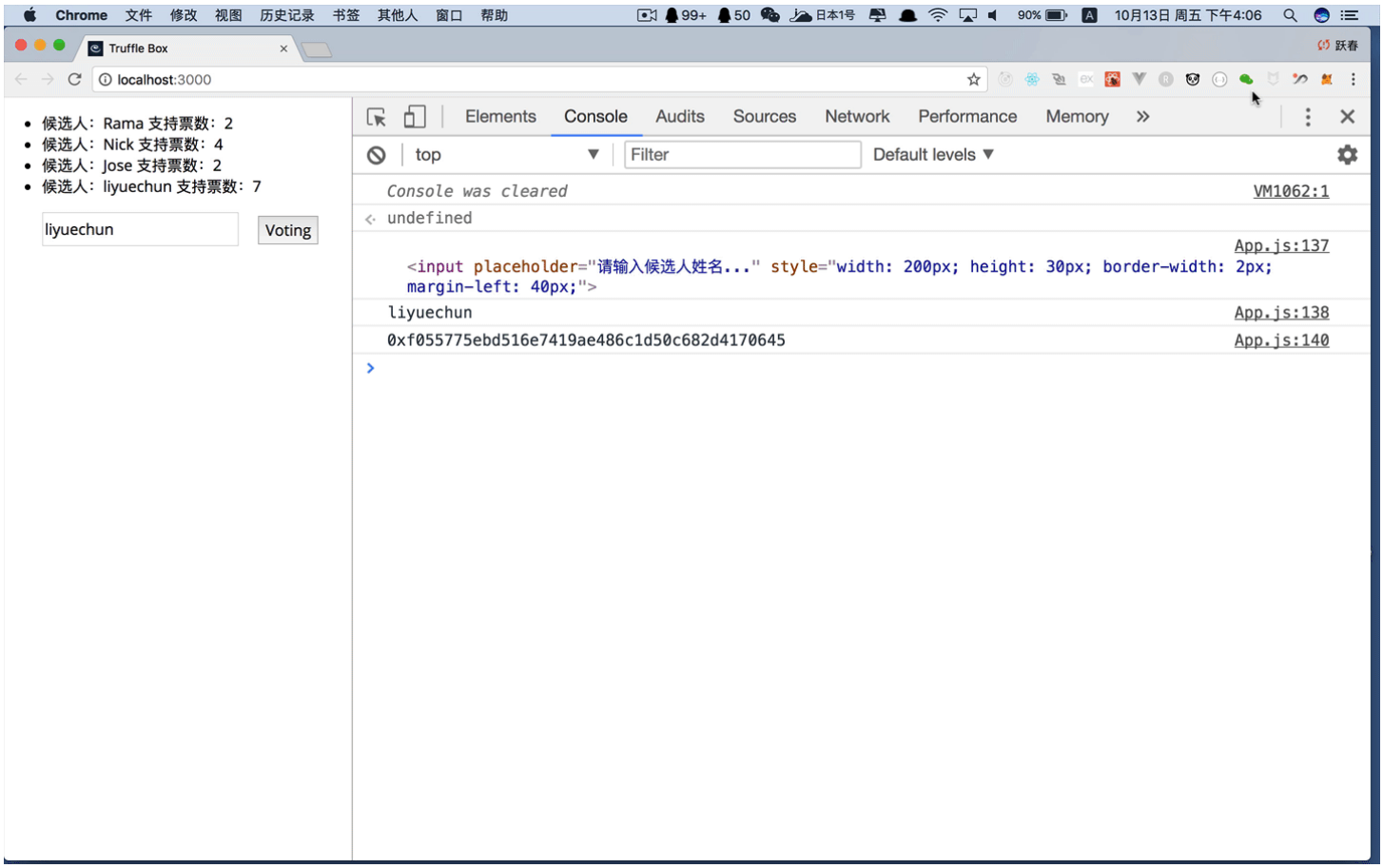
```
import React, { Component } from 'react'
import VotingContract from '../build/contracts/Voting.json'
import getWeb3 from './utils/getWeb3'

import './css/oswald.css'
import './css/open-sans.css'
import './css/pure-min.css'
import './App.css'


const contractAddress = "0xd3f33a2e553b363b432d7f81f721a2a6202ecc67";
var votingContractInstance;


var _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) => {

        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,resul
t.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：{o
bject.votingCount}</li>
            )
         })
        }
      </ul>

      <input
            style={{width: 200,height: 30,borderWidth: 2,marginLeft: 40}}
            placeholder="请输入候选人姓名..."
            ref="candidateInput"
      />

      <button style={{height: 30,borderWidth: 2,marginLeft: 20}} 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).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,re
sult.c[0])
            });
```

```
            });

          }));
        }}>Voting</button>

        </div>
      );
    }
}

export default App
```



打赏地址

比特币：1FcbBw62FHBJKTiLGNoguSwkBdVnJQ9NUn

以太坊：0xF055775eBD516e7419ae486C1d50C682d4170645

技术交流

- 区块链技术交流QQ群：348924182