开发和部署以太坊DApp——投票系统

1、项目简介

本篇文章将介绍如何基于以太坊平台开发简单的投票系统DApp，将学习到以下内容：

- 搭建开发环境（使用到Truffle框架）
- 编写和部署智能合约到区块链
- Web3和智能合约的交互
- MetaMask的使用

该项目比较简单，初始化一组候选人，任何人可以投票给候选人，并显示每个候选人得到的总票数。麻雀虽小，五脏俱全。通过编写此项目，可以学习到编译、部署和交互的全过程。

注意：以下操作均在MacOS上面操作

2、搭建开发环境

进行该项目前，需要安装Node.js和Truffle框架。

- 安装Node.js

  官网：https://nodejs.org/en/

  安装很简单，只需要下载安装包直接安装即可，可以先通过终端检查安装情况再安装，没有对应结果显示再安装:

  ```
  wenzildeiMac:~ wenzil$ npm -v
  3.10.10
  wenzildeiMac:~ wenzil$ node -v
  v6.9.5
  ```

- 安装Truffle：

  Truffle是目前比较流行的Solidity智能合约开发框架，功能十分强大，可以帮助开发者快速地开发一个DApp。

  ```
  npm install -g truffle
  ```

- 安装Ganache CLI：

Ganache CLI是以太坊节点仿真器软件ganache的命令行版本，可以方便开发者快速进行以太坊DApp的开发与测试，Ganache CLI已经取代了TestRPC。

```
npm install -g ganache-cli
```

3、通过Truffle创建项目

建一个项目的目录，然后进入到该目录，如下：

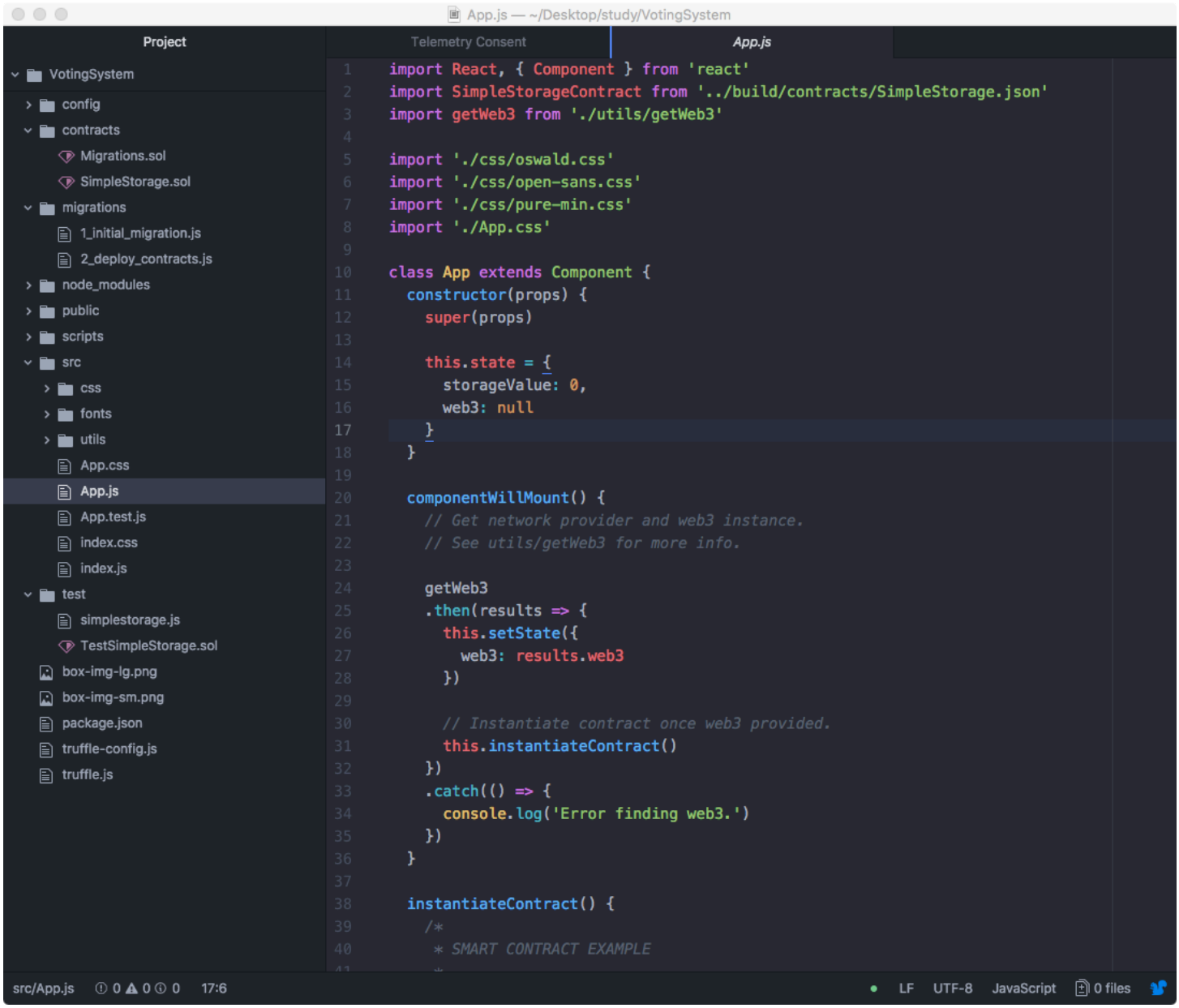
```
carisokdeiMac:study carisok$ pwd
/Users/carisok/Desktop/study
carisokdeiMac:study carisok$ mkdir VotingSystem
carisokdeiMac:study carisok$ cd VotingSystem/
```

然后通过Truffle创建项目，如下：

```
carisokdeiMac:VotingSystem carisok$ 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
```

创建项目过程可能会有点慢，因为会安装项目中"package.json"里面的第三方依赖包，打开"node_modules"目录查看占用了182.1MB。

4、查看项目结构



简单说明：

* contrcts/：存放智能合约Solidity代码的文件夹
* migrations/：存放部署智能合约脚本的文件夹
* src/：存放前端Web代码的文件夹，这里集成了React
* tests/：存放用于智能合约测试用例的文件夹
* package.json：定义了项目所需要的各个模块，项目的配置信息（比如名称、版本、许可证等数据）
* truffle.js：Truffle默认的配置文件

5、编写智能合约

在contracts目录下，创建一个名为Voting.sol的合约文件，代码如下：

```
pragma solidity ^0.4.18;
```

```
contract Voting {
  /*
  mapping：称为映射或者字典，一种键值对的映射关系存储结构
  mapping的key：存储类型为bytes32，存储的是候选人名字
  mapping的value：存储类型为uint8的无符号整型，
  bytes32类型：能存储32个字节，即32*8=256位的二进制内容
  uint8类型：能存储8个字节，即8*8=64位的二进制内容
  */
  mapping (bytes32 => uint8) public votesReceived;

  /*
  Solidity目前不允许字符串数组，这里使用bytes32类型的数组来存储候选人名字
  */
  bytes32[] public candidateList;

  /*
  构造函数，传入bytes32类型的数组，初始化所有候选人名字
  */
  constructor(bytes32[] candidateNames) public {
    candidateList = candidateNames;
  }

  /*
    查询指定候选人的总票数
  */
  function totalVotesFor(bytes32 candidate) constant public returns (uint8)
{
    /*
    require像其他语言中的断言(assert)，用于条件检查。
    条件满足时继续执行，条件不满足则抛出异常。
    */
    require(validCandidate(candidate));
    return votesReceived[candidate];
  }

  /*
    对指定候选人进行投票
  */
  function voteForCandidate(bytes32 candidate) public {
    // 投票前判断是否为候选人名字
    require(validCandidate(candidate));
    votesReceived[candidate] += 1;
  }

  /*
    检查投票名字的有效性，即判断投票名字是否在候选人名字里面
  */
  function validCandidate(bytes32 candidate) constant public returns (bool)
```

```
{
    // 循环遍历候选人列表
    for(uint i = 0; i < candidateList.length; i++) {
      // 判断投票名字是否为候选人
      if (candidateList[i] == candidate) {
        return true;
      }
    }
    return false;
  }
}
```

然后，修改Migrations.sol文件的内容

```
function Migrations() public {
    owner = msg.sender;
}
修改为
constructor() public {
    owner = msg.sender;
}
```

6、编写智能合约部署脚本

在"migratioins"目录中新建一个文件，名字为"3_deploy_voting.js"，内容如下：

```
var Voting = artifacts.require("./Voting.sol");

module.exports = function(deployer) {
  deployer.deploy(Voting, ['Jack Chen', 'Andy Lau', 'Stephen Chow', 'Wenzil'
], {
    gas: 6700000
  });
};
```

7、修改truffle.js

修改全局网络配置信息，可以设置默认gas（为了部署合约的时候忘记设置gas）

```
module.exports = {
  networks: {
    development: {
      host: 'localhost',
```

```
      port: 8545,
      network_id: '*',
      gas: 470000
    }
  }
};
```

8、编译和部署智能合约

8.1 启动Ganache CLI

输入"ganache-cli"命令启动：

```
carisokdeiMac:~ carisok$ ganache-cli
Ganache CLI v6.1.0 (ganache-core: 2.1.0)

Available Accounts
==================
(0) 0x137b0be0e36b991277193a243a5c02203df54055
(1) 0xaf96555135bf00727c9efc36d42823fd5a7a9364
(2) 0xb28720c0f3263b72bb7c742418b9b7a4e4fa707c
(3) 0xb9aa64aef84476493ed9db31549141846c66e59e
(4) 0xfe1825f4039531e88990e1a6e4a102f0ed930fa2
(5) 0x10c26ac6076fb665541dd60e50da2f759c4e5801
(6) 0x93a546b3acc2d34f0a37825ed27fc1ed6618157f
(7) 0xc0c633249d3c241b17f9ed9282c9754397b676c8
(8) 0x5652620911 59e0baf4eb5b78477187f3e9f4bd6e
(9) 0x6f8653f1174f09edc7a38b66a7567f02408938c4

Private Keys
==================
(0) b5ee4dd415b7616b0fd1d4af585579d70970b5b71e4033b02bd7bdb9b5e8ccad
(1) 0f91e7d5f7a4b366a0d95dbae9c4917d22ae95809b1acb4117feadd1a60da5de
(2) 4a61d911ccc12f02ae426c5771fcfffcd2c718dc1fc183e0ce30ed61f0bee8cc
(3) 928c4151d6c8e12f6e6aa72fe01c9baa362032eb7b5af9bd0928bf30cdc39375
(4) 98a9d8938a2375f598b0f0f19b1bfe64509f186333dbf75fefd9e44f4d5189d6
(5) b2e43d00fb5f0b1d7360bdba4f7e408b0a0d45618121d566b56c41fe28c5d4eb
(6) e4f9df75683ccce2a05865d017f756e3091cfd66c858e856b9a82273c5accbb1
(7) 627add4d489b0a8d44a6c87d684ac1119d715bbf44e87fe35fb00a2a309f3df6
(8) 6b37dd934ea80056290f7edad3c294b592b8d1548259685e50c93ca90cb94f03
(9) f4ccb4df8bf5fa6caba9a0daf3112eea09a1bde8d2056b9e129dac939bd93403

HD Wallet
==================
Mnemonic:      arch innocent borrow fog forward pepper legal bulk deposit or
chard decline actor
Base HD Path:  m/44'/60'/0'/0/{account_index}
```

```
Listening on localhost:8545
```

看到启动后自动建立了10个帐号（Accounts），每个帐号中都有100个测试用的以太币（Ether），还有每个帐号对应的私钥（Private Key）。可以用私钥来导入账号进行测试（如在MetaMask设置对应的IP和端口号，然后导入私钥）

8.2 编译智能合约

然后打开新的终端，确认在当前项目的根目录，输入"truffle compile"命令进行编译：

```
carisokdeiMac:VotingSystem carisok$ pwd
/Users/carisok/Desktop/study/VotingSystem
carisokdeiMac:VotingSystem carisok$ ls
box-img-lg.png      node_modules        test
box-img-sm.png      package.json        truffle-config.js
config          public          truffle.js
contracts       scripts
migrations      src
carisokdeiMac:VotingSystem carisok$ truffle compile
Compiling ./contracts/Migrations.sol...
Compiling ./contracts/SimpleStorage.sol...
Compiling ./contracts/Voting.sol...
Writing artifacts to ./build/contracts
```

8.3 部署智能合约

最后，执行"truffle migrate"命令部署智能合约：

```
carisokdeiMac:VotingSystem carisok$ truffle migrate
Using network 'development'.

Running migration: 1_initial_migration.js
  Deploying Migrations...
  ... 0xcea6c6aabbf29a213f4e9bd35d4b2f47fe247777e120532e3269467442a99b41
  Migrations: 0x712e510be1dea2093c9d8e5b49bf34096410f4de
Saving successful migration to network...
  ... 0x1e6cbd04ba0883ceace67973bf2aa2f773e6fd06575691db5c2888a2343f088d
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Deploying SimpleStorage...
  ... 0x55908414c1c4c8c6f0799fa9dce3841aa1015e3cf30ee5436618092c67739303
  SimpleStorage: 0x3d29036cf74ca5045c9c445638db3c2afd46d502
Saving successful migration to network...
```

```
  ... 0x19ad61a3711f497ec95818ee7a4218ab095a3d044ee90bd905f89bc14eb1ac1d
Saving artifacts...
Running migration: 3_deploy_voting.js
  Deploying Voting...
  ... 0x80327895bb6e741d620bb5a2034c8cf9d2c6738c044aba1b03eaac4d9947ab75
  Voting: 0x66c4af2ff3ca1856b57eaa89df38f7435d9ca4cb
Saving successful migration to network...
  ... 0xe7754eedfb46de0a0c740d19c4f669edf3c6912a1822bac65c651fcdfd28ac82
Saving artifacts...
```

9、与投票合约交互

如果智能合约部署成功的话，可以通过truffle控制台进行交互，比如投票和投票统计操作，如下：

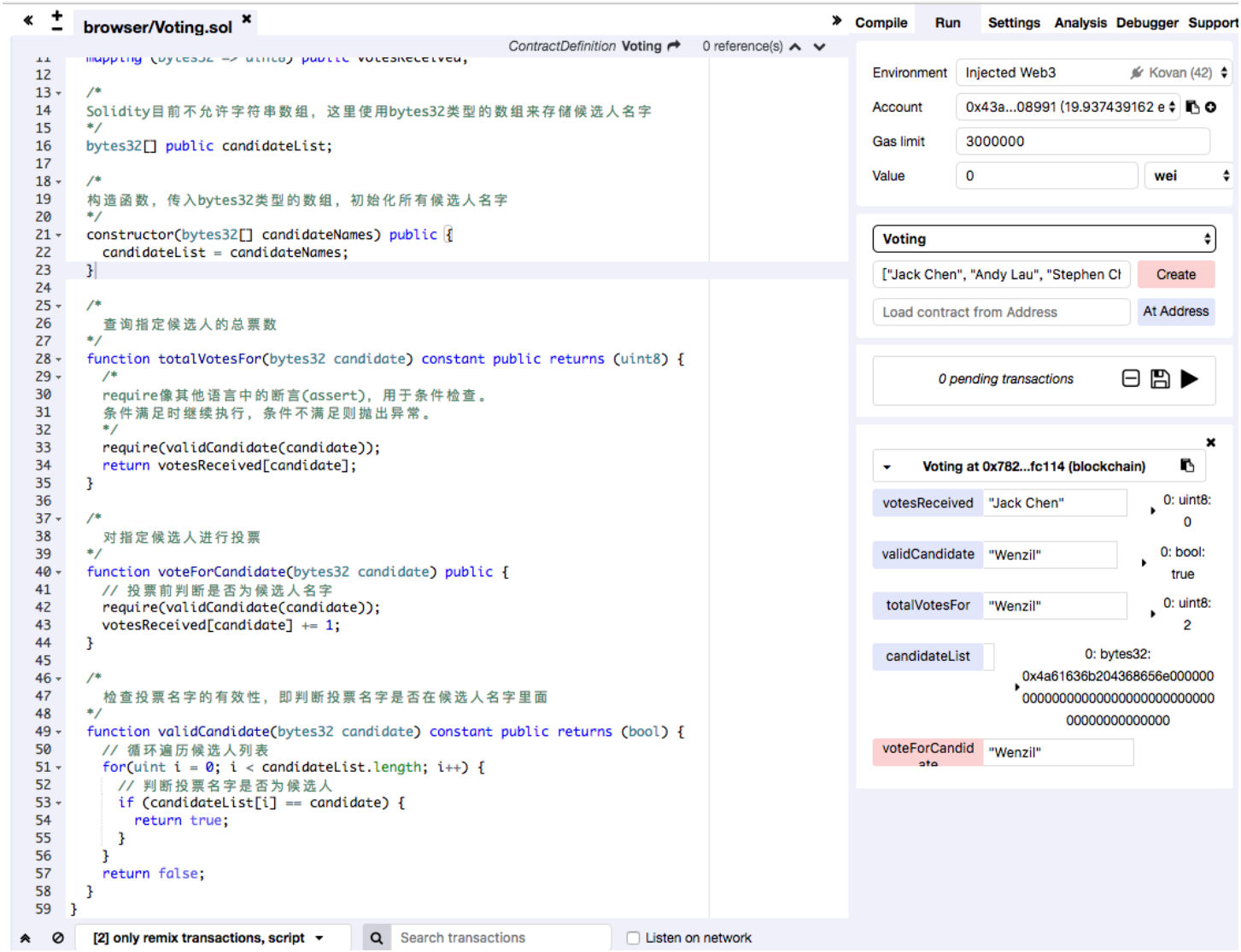
```
truffle(development)> var contract;
truffle(development)> Voting.deployed().then(instance => contract = instance
)
TruffleContract {
  constructor:
   { [Function: TruffleContract]
     _static_methods:
      { setProvider: [Function: setProvider],
        new: [Function: new],
        at: [Function: at],
        deployed: [Function: deployed],
        defaults: [Function: defaults],
        hasNetwork: [Function: hasNetwork],
        isDeployed: [Function: isDeployed],
        detectNetwork: [Function: detectNetwork],
        setNetwork: [Function: setNetwork],
        resetAddress: [Function: resetAddress],
        link: [Function: link],
        clone: [Function: clone],
        addProp: [Function: addProp],
        toJSON: [Function: toJSON] },
     _properties:
      { contract_name: [Object],
##############此处省略N行打印输出##############

truffle(development)> contract.voteForCandidate('Wenzil').then( (result) =>
{ console.log(result) })
{ tx: '0x92f72a7f4ad535a0e4bdf20cc90ee875747345f08ce065b3b538a5cfab7cfe1e',
  receipt:
   { transactionHash: '0x92f72a7f4ad535a0e4bdf20cc90ee875747345f08ce065b3b53
8a5cfab7cfe1e',
```

```
      transactionIndex: 0,
      blockHash: '0x184ef792a90a5fedf2939d8d2effbdb28bb0ca4c08455237b7d546a01
1dc3302',
      blockNumber: 7,
      gasUsed: 45921,
      cumulativeGasUsed: 45921,
      contractAddress: null,
      logs: [],
      status: '0x01',
      logsBloom: '0x00000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000'
 },
  logs: [] }

truffle(development)> contract.totalVotesFor('Wenzil').then( (result) => { c
onsole.log(result) })
{ [String: '1'] s: 1, e: 0, c: [ 1 ] }
```

这说明已成功投了"Wenzil"一票，并且可以总计其票数。

10、修改前端，运行体验

复制Voting.sol到Remix Solidity IDE部署合约，部署成功后复制合约地址



部署成功后，可以进行测试。

复制合约地址，为：
"0x7829abb7d424f118f1243ec13207a02f5bffc114"

然后，修改"src"->"utils"目录下的"getWeb3.js"，修改IP和端口，修改后的内容如下：

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

在工程中"src"目录中新建"mani.css"文件，用于App.js的样式修改，代码如下：

```
.pageTitle {
 margin: 10px;
}

.dataContent {
 border: 1px solid #ccc;
 display:flex;
 width: 50%;
```

```
 margin: 0 0 -1px 0;
}

.dataItem {
 flex: 1;
 align-items: center;
 justify-content: center;
 display: flex;
 padding: 5px;
}

.inputName {
  width: 180px;
  height: 30px;
  margin: 10px 10px 10px 0;
}

.voteButton {
  height: 38px;
  margin-left: 10px;
}
```

修改App.js文件，文件内容：

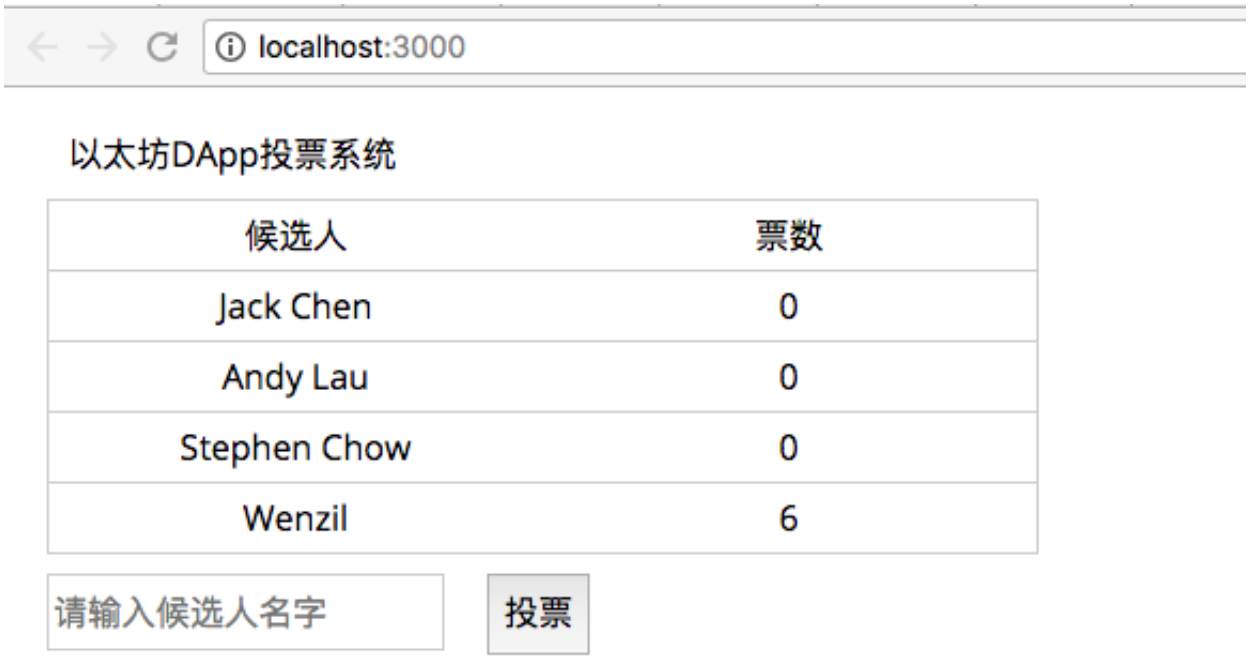
```
import React, {
  Component
}
from 'react'
import VotingContract from '../build/contracts/Voting.json'
import getWeb3 from './utils/getWeb3'

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

// 合约的实例
var votingContractInstance;
// 投票合约地址
const contractAddr = "0x7829abb7d424f118f1243ec13207a02f5bffc114"

class App extends Component {
  constructor(props) {
    super(props)

        this.state = {
```

```
        storageValue: 0,
        web3: null,
        candidateList: [{
          "candidateId": 1,
          "candidateName": "Jack Chen",
          "voteNumber": 0
        }, {
          "candidateId": 2,
          "candidateName": "Andy Lau",
          "voteNumber": 0
        }, {
          "candidateId": 3,
          "candidateName": "Stephen Chow",
          "voteNumber": 0
        }, {
          "candidateId": 4,
          "candidateName": "Wenzil",
          "voteNumber": 0
        }]
      }
}

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(contractAddr).then((instance) => {
        votingContractInstance = instance
        // 读取合约成功后，遍历所有候选人的票数，并更新到前端
        var candidateListTemp = this.state.candidateList;
        for (let i = 0; i < this.state.candidateList.length; i++) {
          let object = candidateListTemp[i];
          votingContractInstance.totalVotesFor(object.candidateName).then(re
sult => {
              object.voteNumber = result.c[0]
              this.setState({
                candidateList: candidateListTemp
              })
          });
        }
      })
    })
  }

  render() {
    return (
      <div className="App" style={{margin:20}}>
        <div className="pageTitle">以太坊DApp投票系统</div>
        <div className="dataContent">
          <div className="dataItem">候选人</div>
          <div className="dataItem">票数</div>
        </div>
        {/* 读取当前页面的候选人列表，并显示 */}
        {
         this.state.candidateList.map((item) => {
           return (
              <div className="dataContent" key={item.candidateId}>
                <div className="dataItem">{item.candidateName}</div>
                <div className="dataItem">{item.voteNumber}</div>
              </div>
           )
         })
        }
        <input className="inputName" placeholder="请输入候选人名字" ref="inputN
ame" />
        {/* 获取输入框的内容 */}
```

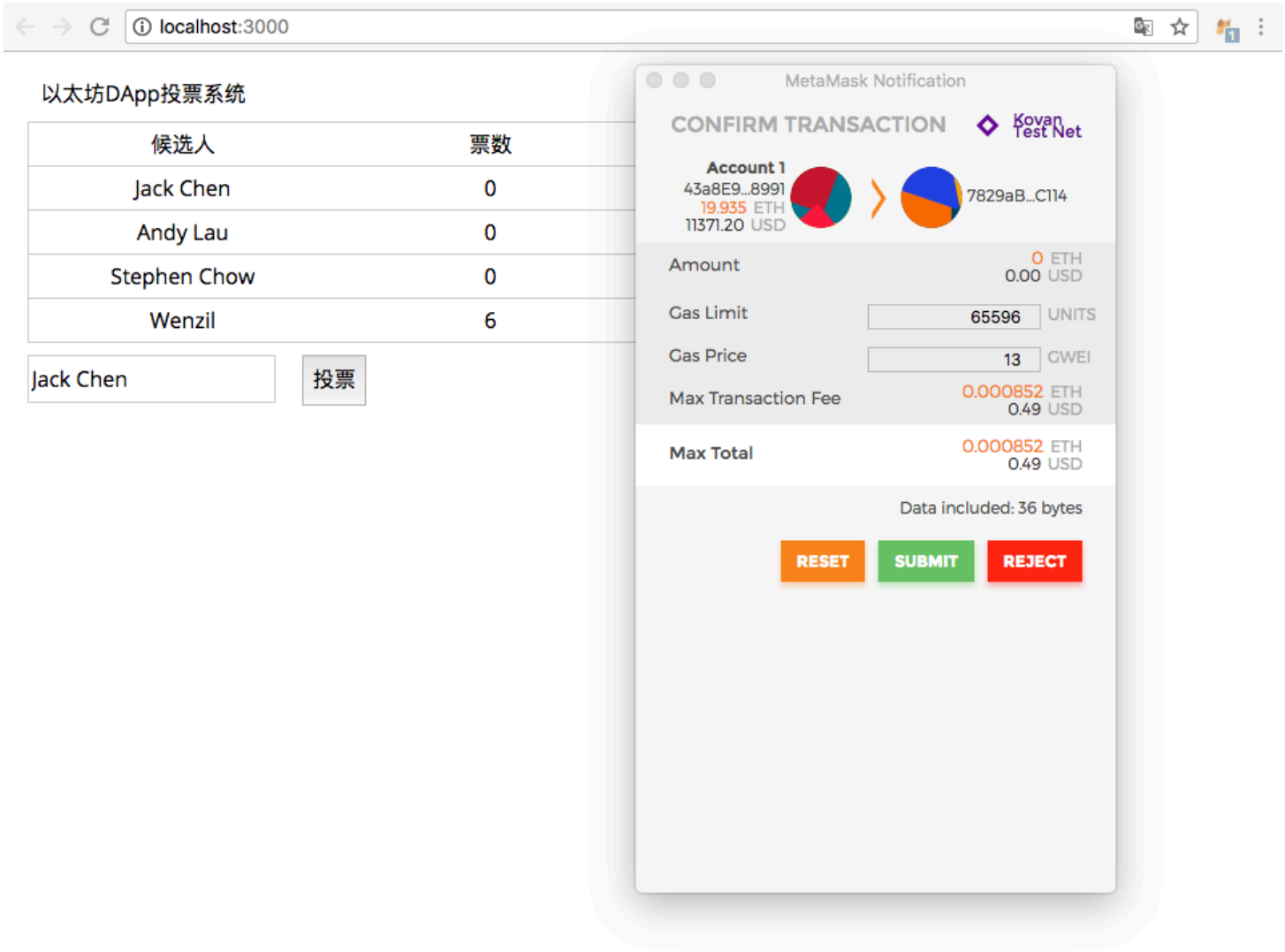
```
        <button className="voteButton"
          onClick={() => {
            let candidateName = this.refs.inputName.value;
            let account = this.state.web3.eth.accounts[0];
            {/* 为输入的候选人投票，写入到区块链中 */}
            votingContractInstance.voteForCandidate(candidateName,{from: acc
ount}).then((result) => {
              var currentCandidateIndex = 0;
              {/* 获取输入候选人的数组下标 */}

              for(let i = 0; i < this.state.candidateList.length; i++) {
                let currentCandidate = this.state.candidateList[i];
                if (currentCandidate.candidateName === candidateName) {
                   currentCandidateIndex = i;
                  break;
                }
              }
              {/* 根据合约读取区块中的候选人列表数据，并刷新到前端页面 */}
              votingContractInstance.totalVotesFor(candidateName).then(resul
t => {
                var candidateListTemp = this.state.candidateList;
                {/* 根据输入候选人的数组下标来更新列表数据 */}
                for (let i = 0; i < candidateListTemp.length; i++) {
                  if (candidateName === candidateListTemp[currentCandidateIn
dex].candidateName) {
                    candidateListTemp[currentCandidateIndex].voteNumber = re
sult.c[0]
                  }
                }
                this.setState({
                  candidateList: candidateListTemp
                })
              })
            })
          }}>投票</button>
      </div>
    );
  }
}

export default App
```

输入"npm run start"命令启动服务器，会自动打开网页。



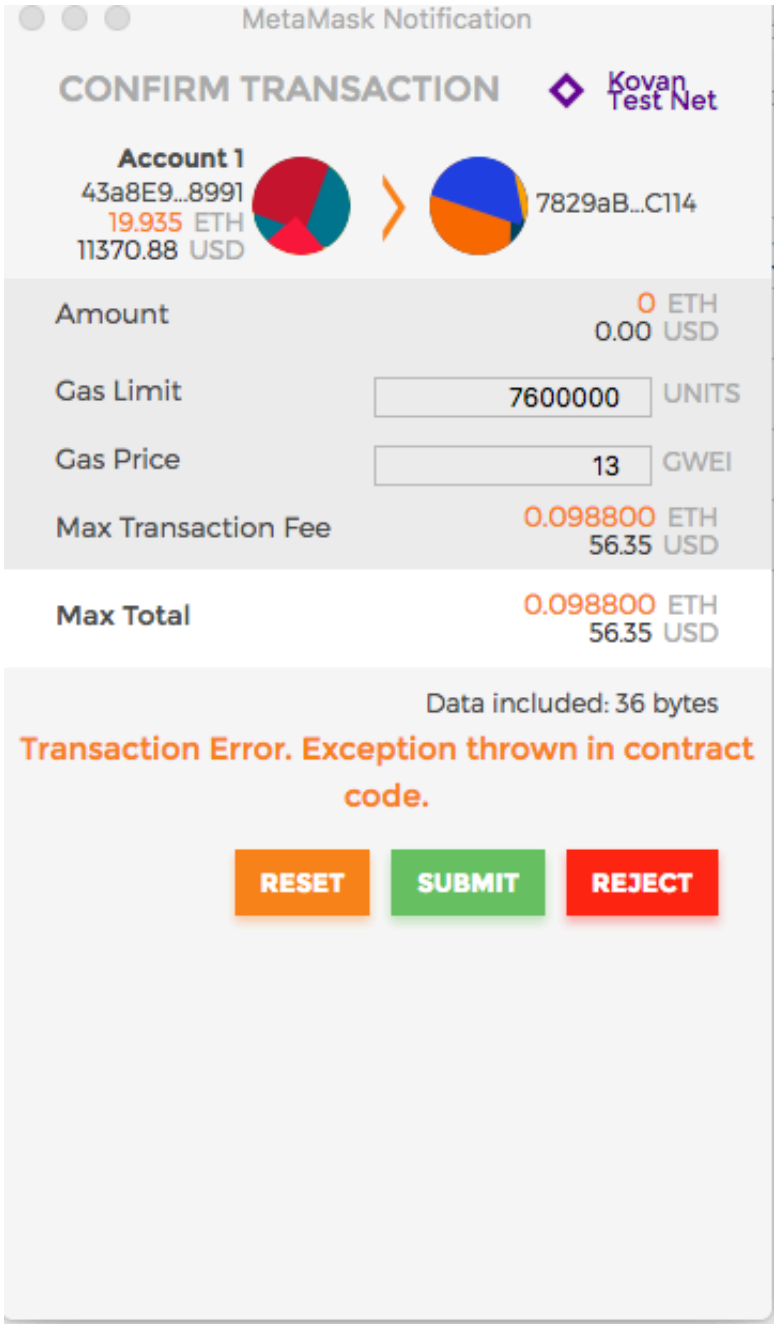
刚开始"Wenzil"的投票是1，因为在Remix Solidity IDE部署合约的时候投了1票，然后手动投票了好几次，所以为6，没有重新实验了。

演示下为"Jack Chen"投票，会弹出MetaMask确认交易

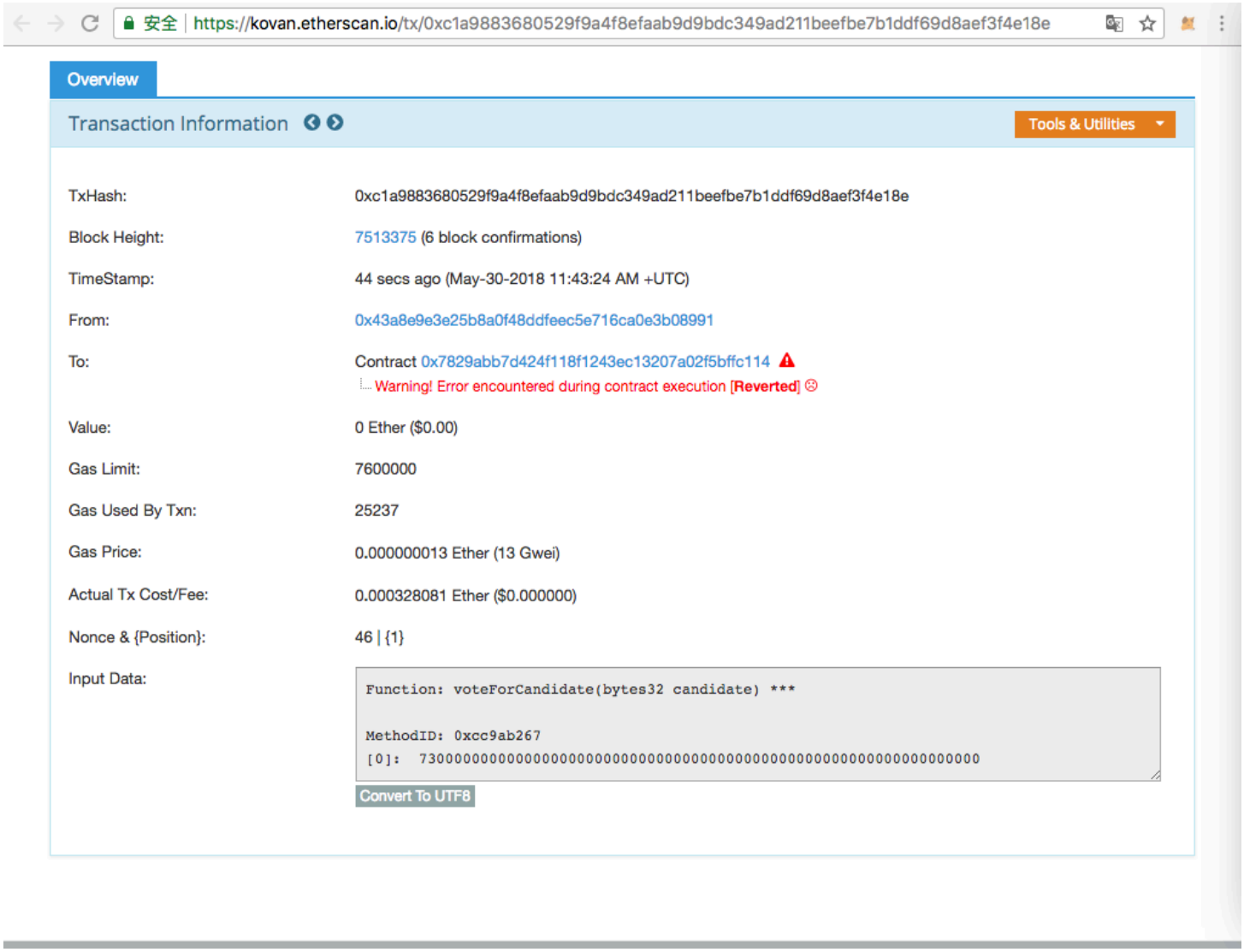


等几秒之后，页面自动更新票数。

如果输入非候选人名字的话，MetaMask会显示错误信息。



继续提交，打开Etherscan查看区块链信息，显示如图:


安全 | https://kovan.etherscan.io/tx/0xc1a9883680529f9a4f8efaab9d9bdc349ad211beefbe7b1ddf69d8aef3f4e18e
Overview
Transaction Information
Tools & Utilities
TxHash:
0xc1a9883680529f9a4f8efaab9d9bdc349ad211beefbe7b1ddf69d8aef3f4e18e
Block Height:
7513375 (6 block confirmations)
TimeStamp:
44 secs ago (May-30-2018 11:43:24 AM +UTC)
From:
0x43a8e9e3e25b8a0f48ddfeec5e716ca0e3b08991
To:
Contract 0x7829abb7d424f118f1243ec13207a02f5bffc114
Warning! Error encountered during contract execution [Reverted]
Value:
0 Ether ($0.00)
Gas Limit:
7600000
Gas Used By Txn:
25237
Gas Price:
0.000000013 Ether (13 Gwei)
Actual Tx Cost/Fee:
0.000328081 Ether ($0.000000)
Nonce & {Position}:
46 | {1}
Input Data:
Function: voteForCandidate(bytes32 candidate) ***
MethodID: 0xcc9ab267
[0]: 7300000000000000000000000000000000000000000000000000000000000000
Convert To UTF8

搞定，收工。