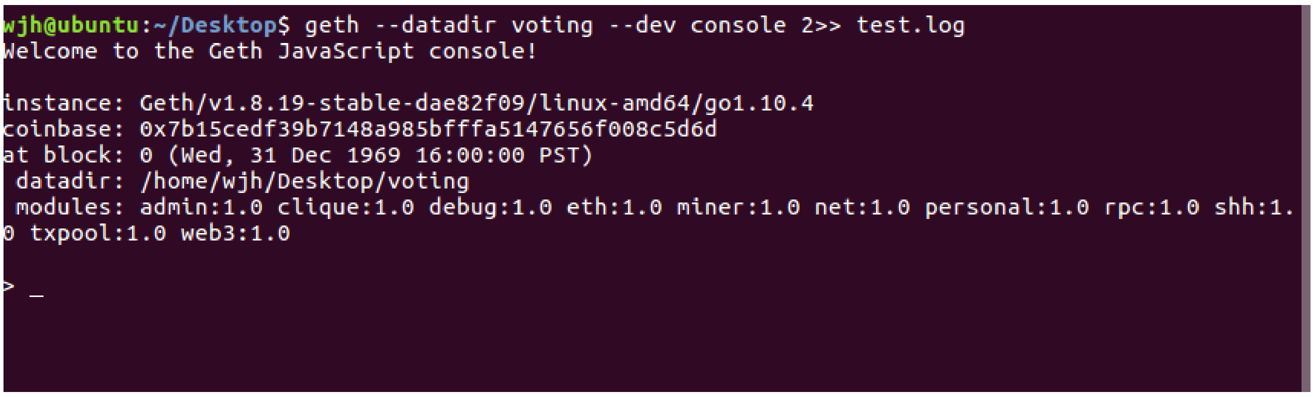
最终报告

启动环境

`geth` 是一个以太坊客户端，现在利用 `geth` 启动一个以太坊（开发者）网络节点。

`geth --datadir testNet --dev console 2>> test.log`

执行命名后，会进入geth控制台，这时光标停在一个向右的箭头处，像这样



```
wjh@ubuntu:~/Desktop$ geth --datadir voting --dev console 2>> test.log
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.19-stable-dae82f09/linux-amd64/go1.10.4
coinbase: 0x7b15cedf39b7148a985bfffa5147656f008c5d6d
at block: 0 (Wed, 31 Dec 1969 16:00:00 PST)
 datadir: /home/wjh/Desktop/voting
 modules: admin:1.0 clique:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 shh:1.
0 txpool:1.0 web3:1.0

> _
```

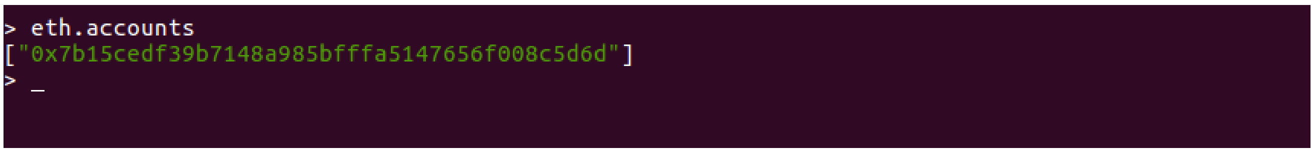
命令参数说明: `--dev` 启用开发者网络（模式），开发者网络会使用POA共识，默认预分配一个开发者账户并且会自动开启挖矿。 `--datadir` 后面的参数是区块数据及秘钥存放目录。第一次输入命令后，它会放在当前目录下新建一个 `testNet` 目录来存放数据。 `console` 进入控制台 `2>> test.log` 表示把控制台日志输出到 `test.log` 文件

准备账户

部署智能合约需要一个外部账户，我们先来看看分配的开发者账户，在控制台使用以下命令查看账户：

`eth.accounts`

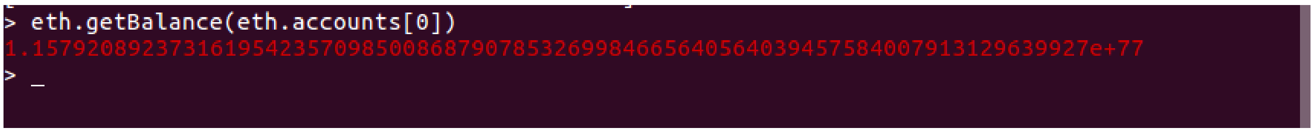
回车后，返回一个账户数组，里面有一个默认账户，如：



```
> eth.accounts
["0x7b15cedf39b7148a985bfffa5147656f008c5d6d"]
> _
```

再来看一下账户里的余额，使用一下命令：

`eth.getBalance(eth.accounts[0])`



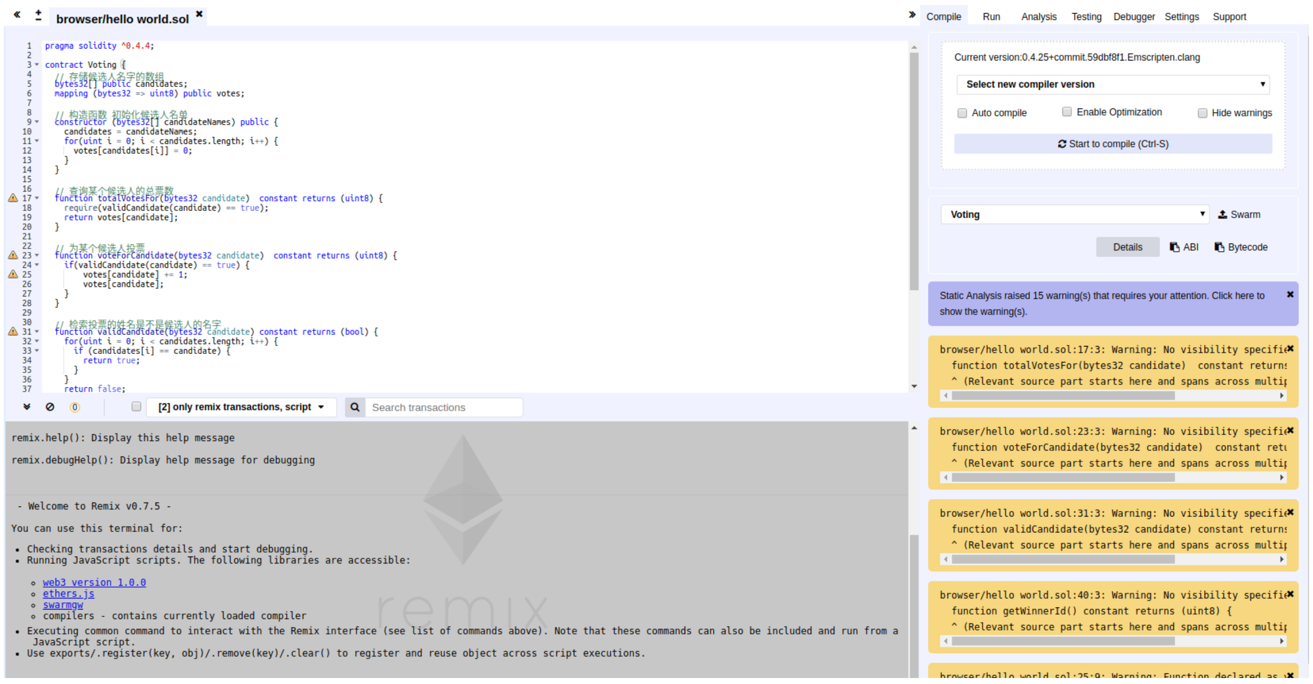
```
> eth.getBalance(eth.accounts[0])
1.15792089237316195423570985008687907853269984665640564039457584007913129639927e+77
> _
```

编写合约代码

现在我们来开始编写第一个智能合约代码，`solidity` 代码如下：

```
pragma solidity ^0.4.4;

contract Voting {
  // 存储候选人名字的数组
  bytes32[] public candidateList;
  mapping (bytes32 => uint8) public votesReceived;

  // 构造函数 初始化候选人名单
  constructor (bytes32[] candidateNames) public {
    candidateList = candidateNames;
    for(uint i = 0; i < candidateList.length; i++) {
      votesReceived[candidateList[i]] = 0;
    }
  }

  // 查询某个候选人的总票数
  function totalVotesFor(bytes32 candidate)  constant returns (uint8) {
    require(validCandidate(candidate) == true);
    return votesReceived[candidate];
  }

  // 为某个候选人投票
  function voteForCandidate(bytes32 candidate)  constant returns (uint8) {
    if(validCandidate(candidate) == true) {
        votesReceived[candidate] += 1;
        votesReceived[candidate];
    }
  }

  // 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;
  }
  // 获取获胜者在候选人数组的序列号
  function getWinnerId() constant returns (uint8) {
    uint8 nameId = 0;
    uint8 k = 0;
    for(uint8 i = 0; i < candidateList.length; i++) {
      if (k <= votesReceived[candidateList[i]]) {
        nameId = i;
        k = votesReceived[candidateList[i]];
      }
    }
    return nameId;
  }
}
```

把这段代码写(拷贝)到 `Browser-Solidity`，如果没有错误，点击 `Details` 获取部署代码，如：



```
pragma solidity ^0.4.4;

contract Voting {
  // 存储候选人名字的数组
  bytes32[] public candidates;
  mapping (bytes32 => uint8) public votes;

  // 构造函数 初始化候选人名单
  constructor (bytes32[] candidateNames) public {
    candidates = candidateNames;
    for(uint i = 0; i < candidates.length; i++) {
      votes[candidates[i]] = 0;
    }
  }

  // 查询某个候选人的总票数
  function totalVotesFor(bytes32 candidate)  constant returns (uint8) {
    require(validCandidate(candidate) == true);
    return votes[candidate];
  }

  // 为某个候选人投票
  function voteForCandidate(bytes32 candidate)  constant returns (uint8) {
    if(validCandidate(candidate) == true) {
        votes[candidate] += 1;
        votes[candidate];
    }
  }

  // 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 < candidates.length; i++) {
      if (candidates[i] == candidate) {
        return true;
      }
    }
    return false;
```

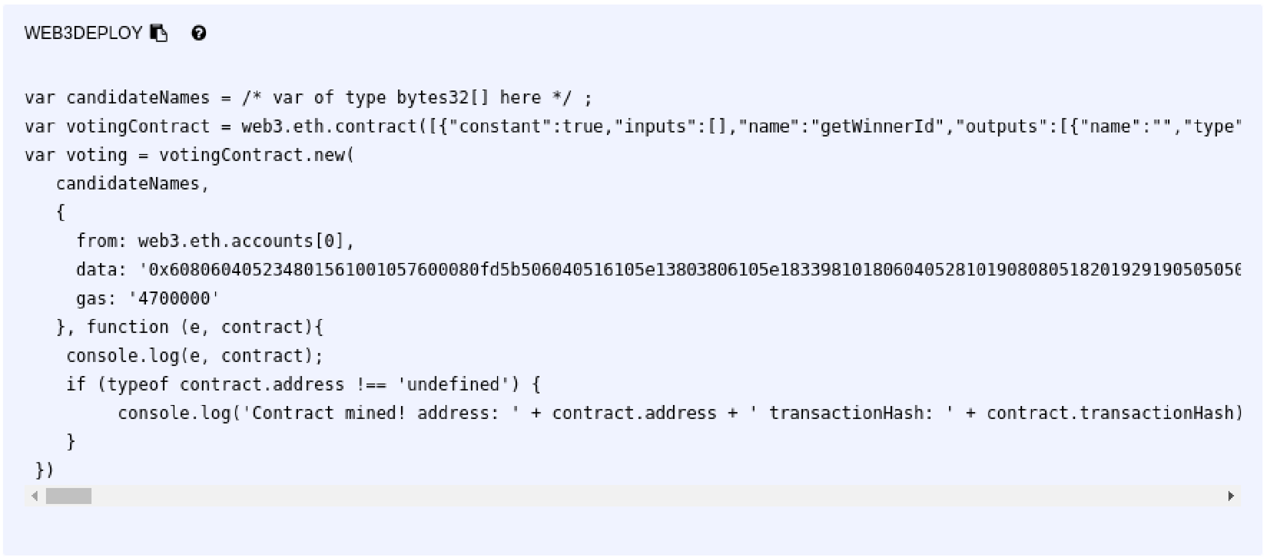
Current version:0.4.25+commit.59dbf8f1.Emscripten.clang

Static Analysis raised 15 warning(s) that requires your attention. Click here to show the warning(s).

```
browser/hello world.sol:17:3: Warning: No visibility specifie
  function totalVotesFor(bytes32 candidate)  constant returns
  ^ (Relevant source part starts here and spans across multip

browser/hello world.sol:23:3: Warning: No visibility specifie
  function voteForCandidate(bytes32 candidate)  constant retu
  ^ (Relevant source part starts here and spans across multip

browser/hello world.sol:31:3: Warning: No visibility specifie
  function validCandidate(bytes32 candidate) constant returns
  ^ (Relevant source part starts here and spans across multip

browser/hello world.sol:40:3: Warning: No visibility specifie
  function getWinnerId() constant returns (uint8) {
  ^ (Relevant source part starts here and spans across multip

browser/hello world.sol:25:9: Warning: Function declared as
```

```
remix.help(): Display this help message

remix.debugHelp(): Display help message for debugging

 - Welcome to Remix v0.7.5 -

You can use this terminal for:

• Checking transactions details and start debugging.
• Running JavaScript scripts. The following libraries are accessible:

   ◦ web3 version 1.0.0
   ◦ ethers.js
   ◦ swarmgw
   ◦ compilers - contains currently loaded compiler
• Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be included and run from a
  JavaScript script.
• Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.
```

弹出的对话框中找到 `WEB3DEPLOY` 部分，点拷贝，粘贴到编辑器


WEB3DEPLOY

```
var candidateNames = /* var of type bytes32[] here */ ;
var votingContract = web3.eth.contract([{"constant":true,"inputs":[],"name":"getWinnerId","outputs":[{"name":"","type"
var voting = votingContract.new(
   candidateNames,
   {
     from: web3.eth.accounts[0],
     data: '0x608060405234801561001057600080fd5b506040516105e13803806105e1833981018060405281019080805182019291905050505
     gas: '4700000'
   }, function (e, contract){
    console.log(e, contract);
    if (typeof contract.address !== 'undefined') {
         console.log('Contract mined! address: ' + contract.address + ' transactionHash: ' + contract.transactionHash)
    }
 })
```

添加如下代码，注意这是初始化候选人序列，这里我们有两个候选人，张三和李四

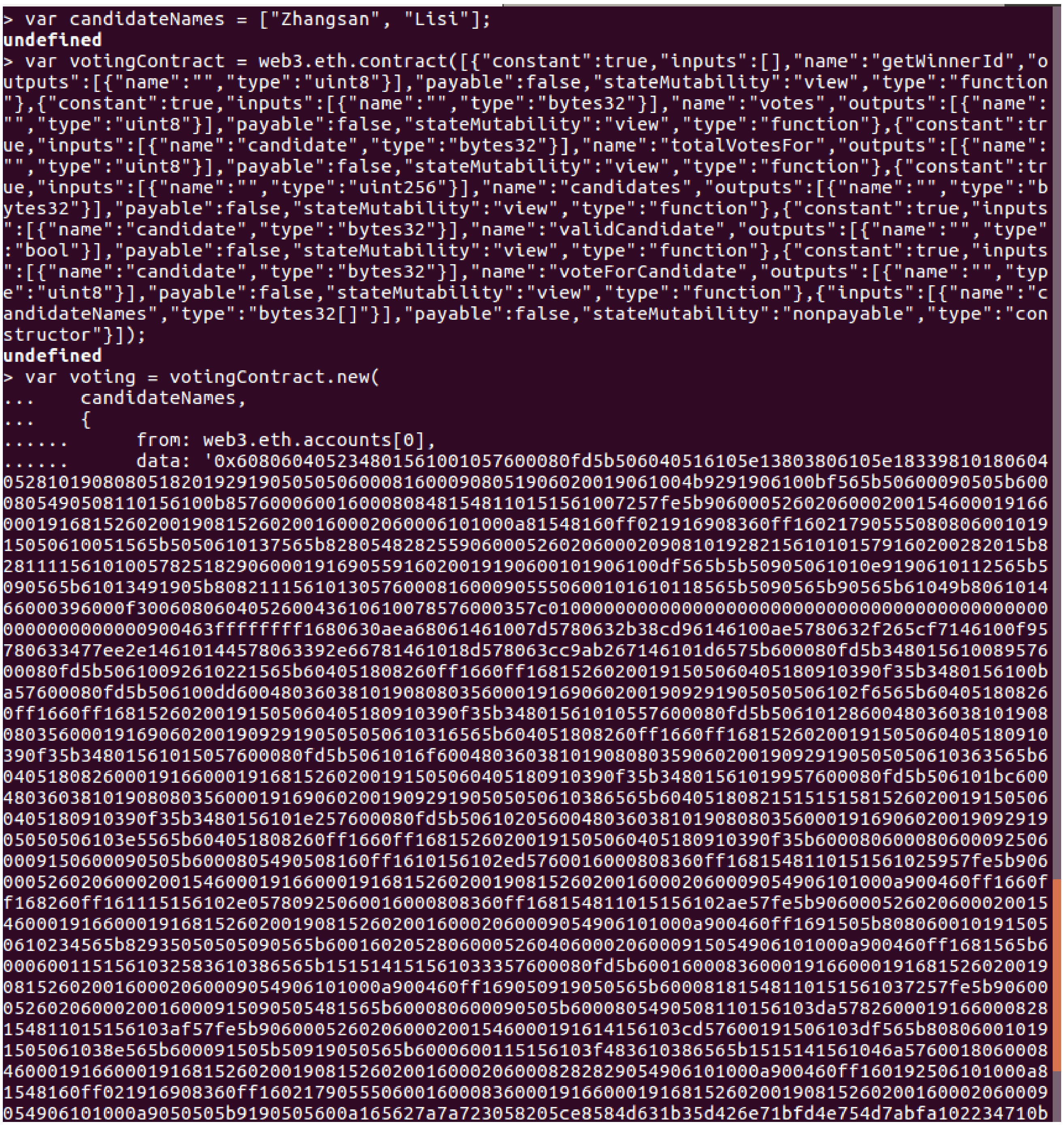
`var candidateNames = ["Zhangsan", "Lisi"];`

部署合约

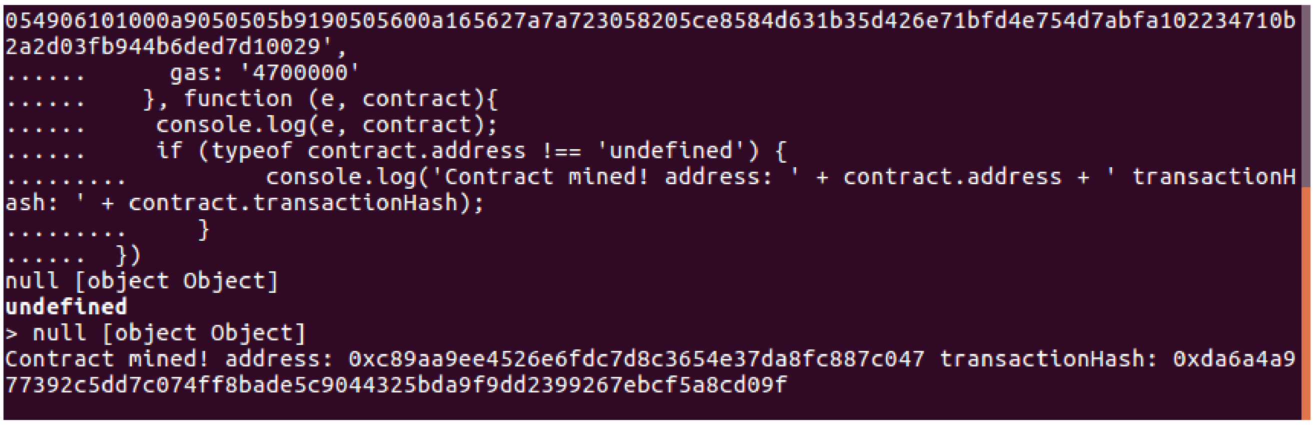
`Browser-Solidity` 生成的代码，拷贝到编辑器里修改后的代码如下：

```
var candidateNames = ["Zhangsan", "Lisi"];
var votingContract = web3.eth.contract([{"constant":true,"inputs":
[],"name":"getWinnerId","outputs":
[{"name":"","type":"uint8"}],"payable":false,"stateMutability":"view","type":"function"},
{"constant":true,"inputs":[{"name":"","type":"bytes32"}],"name":"votes","outputs":
[{"name":"","type":"uint8"}],"payable":false,"stateMutability":"view","type":"function"},
{"constant":true,"inputs":
[{"name":"candidate","type":"bytes32"}],"name":"totalVotesFor","outputs":
[{"name":"","type":"uint8"}],"payable":false,"stateMutability":"view","type":"function"},
{"constant":true,"inputs":[{"name":"","type":"uint256"}],"name":"candidates","outputs":
[{"name":"","type":"bytes32"}],"payable":false,"stateMutability":"view","type":"function"},
{"constant":true,"inputs":
[{"name":"candidate","type":"bytes32"}],"name":"validCandidate","outputs":
[{"name":"","type":"bool"}],"payable":false,"stateMutability":"view","type":"function"},
{"constant":true,"inputs":
[{"name":"candidate","type":"bytes32"}],"name":"voteForCandidate","outputs":
[{"name":"","type":"uint8"}],"payable":false,"stateMutability":"view","type":"function"},
{"inputs":
[{"name":"candidateNames","type":"bytes32[]"}],"payable":false,"stateMutability":"nonpaya
ble","type":"constructor"}]);
var voting = votingContract.new(
    candidateNames,
    {
      from: web3.eth.accounts[0],
        data:
'0x608060405234801561001057600080fd5b506040516105e13803806105e1833981018060
40528101908080518201929190505050600081600090805190602001906100 4b9291906100
bf565b50600090505b6000805490508110156100b857600060016000808481548110151561
07257fe5b906000526020600020015460001916600019168152602001908152602001600020
600006101000a81548160ff021916908360ff1602179055508080600101915050610051565b5
050610137565b82805482825590600052602060002090810192821561010157916020028 20
15b828111156101005782518290600019169055916020019190600101906100df565b5b50905
061010e9190610112565b5090565b61013491905b80821115610130576000816000905550600
101610118565b5090565b90565b61049b806101466000396000f30060806040526004361061
0078576000357c0100000000000000000000000000000000000000000000000000000000
00900463ffffffff1680630aea68061461007d5780632b38cd96146100ae5780632f265cf7146
100f95780633477ee2e1461014457806339 2e66781461018d578063cc9ab267146101d6575b60
0080fd5b34801561008957600080fd5b50610092610221565b604051808260ff1660ff168152
60200191505060405180910390f35b3480156100ba57600080fd5b506100dd6004803603810
190808035600019169060200190929190505050610 2f6565b604051808260ff1660ff168152
60200191505060405180910390f35b34801561010557600080fd5b50610128600480360381 0
190808035600019169060200190929190505050610316565b604051808260ff1660ff168152
60200191505060405180910390f35b34801561015057600080fd5b5061016f6004803603810
1908080359060200190929190505050610363565b604051808260001916600019168152602
00191505060405180910390f35b34801561019957600080fd5b506101bc600480360381 01908
0803560001916906020019092919050505061038656 5b604051808215151515815260200191
505060405180910390f35b3480156101e257600080fd5b5061020560048036038101908080 3
5600019169060200190929190505050610 3e5565b604051808260ff1660ff16815260200191
505060405180910390f35b600080600080600092506000915060009050 5b60008054905 0
8160ff1610156102ed57600160008083 60ff16815481101515610259 57fe5b906000526020600
020015460001916600019168152602001908152602001600020600090549061010 00a90046
0ff1660ff168260ff161115156102e0578092506001600080836 0ff168154811015156102ae57f
e5b906000526020600020015460001916600019168152602001908152602001600020600 0
9054906101000a900460ff1691505b808060010191505061023456 5b829350505050 90565b6
0016020528060005260406000206000915054906101000a900460ff1681565b60006001151
56103258361038656 5b15151415156103335760008 0fd5b6001600083600019166000191681 5
2602001908152602001600020600090549061010 00a900460ff16905091905056 5b600081
815481101515610372 57fe5b906000526020600020016000915090505481565b6000806000
90505b6000805490508110156103da57826000191660008281548110151561 03af57fe5b9060
0052602060002001546000191614156103cd5760019150610 3df565b80806001019150506 10
38e565b600091505b509190505 65b6000600115156103f4836103865 65b1515141561046a576
0018060008460001916600019168152602001908152602001600020600082828290549061
01000a900460ff1601925061010 00a81548160ff021916908360ff16021790555060016000 83
600019166000191681526020019081526020016000206000905490610 1000a9050505b9190
505600a165627a7a723058205ce8584d631b35d426e71bfd4e754d7abfa102234710b2a2d03fb
944b6ded7d10029',
        gas: '4700000'
      }, function (e, contract){
       console.log(e, contract);
       if (typeof contract.address !== 'undefined') {
            console.log('Contract mined! address: ' + contract.address + ' transactionHash:
  ' + contract.transactionHash);
       }
    })
```

拷贝会 `geth` 控制台里，回车后，看到输出如：



```
> var candidateNames = ["Zhangsan", "Lisi"];
undefined
> var votingContract = web3.eth.contract([{"constant":true,"inputs":[],"name":"getWinnerId","o
utputs":[{"name":"","type":"uint8"}],"payable":false,"stateMutability":"view","type":"function
"},{"constant":true,"inputs":[{"name":"","type":"bytes32"}],"name":"votes","outputs":[{"name":
"","type":"uint8"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":tr
ue,"inputs":[{"name":"candidate","type":"bytes32"}],"name":"totalVotesFor","outputs":[{"name":
"","type":"uint8"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":tr
ue,"inputs":[{"name":"","type":"uint256"}],"name":"candidates","outputs":[{"name":"","type":"b
ytes32"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":true,"inputs
":[{"name":"candidate","type":"bytes32"}],"name":"validCandidate","outputs":[{"name":"","type"
:"bool"}],"payable":false,"stateMutability":"view","type":"function"},{"constant":true,"inputs
":[{"name":"candidate","type":"bytes32"}],"name":"voteForCandidate","outputs":[{"name":"","typ
e":"uint8"}],"payable":false,"stateMutability":"view","type":"function"},{"inputs":[{"name":"c
andidateNames","type":"bytes32[]"}],"payable":false,"stateMutability":"nonpayable","type":"con
structor"}]);
undefined
> var voting = votingContract.new(
...     candidateNames,
...     {
......       from: web3.eth.accounts[0],
......       data: '0x608060405234801561001057600080fd5b506040516105e13803806105e18339810180604
05281019080805182019291905050506000816000908051906020019061004b9291906100bf565b50600090505b600
0805490508110156100b857600060016000808481548110151561007257fe5b906000526020600020015460001916
60001916815260200190815260200160002060006101000a81548160ff021916908360ff160217905550808060010191
5050610051565b5050610137565b828054828255906000526020600020908101928215610101579160200282015b8
28111156101005782518290600019169055916020019190600101906100df565b5b50905061010e9190610112565b5
090565b61013491905b80821115610130576000816000905550600101610118565b5090565b90565b61049b8061014
66000396000f300608060405260043610610078576000357c0100000000000000000000000000000000000000000000
0000000000000900463ffffffff1680630aea68061461007d5780632b38cd96146100ae5780632f265cf7146100f95
780633477ee2e14610144578063392e66781461018d578063cc9ab267146101d6575b600080fd5b348015610089576
00080fd5b50610092610221565b604051808260ff1660ff16815260200191505060405180910390f35b3480156100b
a57600080fd5b506100dd60048036038101908080356000191690602001909291905050506102f6565b60405180826
0ff1660ff16815260200191505060405180910390f35b34801561010557600080fd5b5061012860048036038101908
08035600019169060200190929190505050610316565b604051808260ff1660ff16815260200191505060405180910
390f35b34801561015057600080fd5b5061016f60048036038101908080359060200190929190505050610363565b6
0405180826000191660001916815260200191505060405180910390f35b34801561019957600080fd5b506101bc600
4803603810190808035600019169060200190929190505050610386565b60405180821515151581526020019150506
0405180910390f35b3480156101e257600080fd5b50610205600480360381019080803560001916906020019092919
05050506103e5565b604051808260ff1660ff16815260200191505060405180910390f35b600080600080600092506
0009150600090505b6000805490508160ff1610156102ed5760016000808360ff1681548110151561025957fe5b906
00052602060002001546000191660001916815260200190815260200160002060009054906101000a900460ff1660f
f168260ff161115156102e05780925060016000808360ff168154811015156102ae57fe5b906000526020600020015
46000191660001916815260200190815260200160002060009054906101000a900460ff1691505b808060010191505
0610234565b82935050505090565b60016020528060005260406000206000915054906101000a900460ff1681565b6
0006001151561032583610386565b151514151561033357600080fd5b6001600083600019166000191681526020019
0815260200160002060009054906101000a900460ff169050919050565b60008181548110151561037257fe5b90600
0526020600020016000915090505481565b600080600090505b6000805490508110156103da5782600019166000828
154811015156103af57fe5b90600052602060002001546000191614156103cd57600191506103df565b808060010191
505061038e565b600091505b50919050565b6000600115156103f483610386565b1515141561046a5760018060008
4600019166000191681526020019081526020016000208282829054906101000a900460ff160192506101000a8
1548160ff021916908360ff16021790555060016000836000191660001916815260200190815260200160002060009
054906101000a9050505b9190505600a165627a7a723058205ce8584d631b35d426e71bfd4e754d7abfa102234710b
```

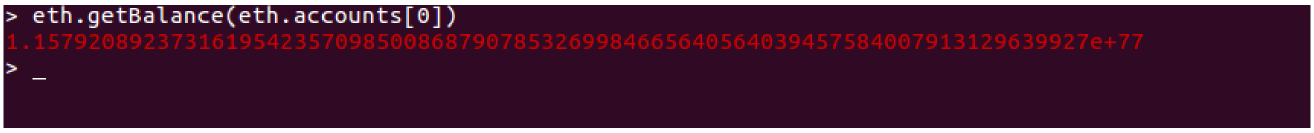


```
054906101000a9050505b9190505600a165627a7a723058205ce8584d631b35d426e71bfd4e754d7abfa102234710b
2a2d03fb944b6ded7d10029',
......       gas: '4700000'
......     }, function (e, contract){
......      console.log(e, contract);
......      if (typeof contract.address !== 'undefined') {
.........          console.log('Contract mined! address: ' + contract.address + ' transactionH
ash: ' + contract.transactionHash);
.........       }
......   })
null [object Object]
undefined
> null [object Object]
Contract mined! address: 0xc89aa9ee4526e6fdc7d8c3654e37da8fc887c047 transactionHash: 0xda6a4a9
77392c5dd7c074ff8bade5c9044325bda9f9dd2399267ebcf5a8cd09f
```

说明合约已经部署成功
]]>

现在我们查看下新账户的余额：

`eth.getBalance(eth.accounts[0])`



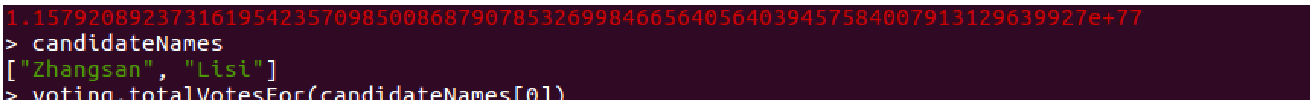
```
> eth.getBalance(eth.accounts[0])
1.15792089237316195423570985008687907853269984665640564039457584007913129639927e+77
> _
```

是不是比之前转账的余额少呀！

运行合约进行投票测试

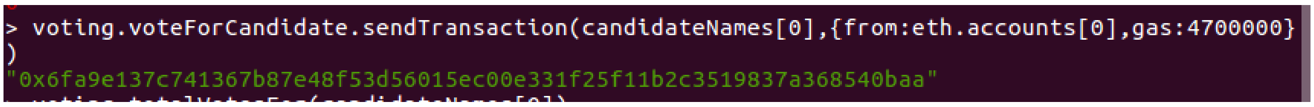
候选人列表展示

`candidateNames`



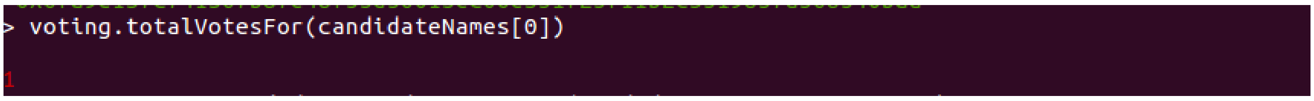
```
1.15792089237316195423570985008687907853269984665640564039457584007913129639927e+77
> candidateNames
["Zhangsan", "Lisi"]
> voting.totalVotesFor(candidateNames[0])
```

给 0 号候选人及 `Zhangsan` 投一票



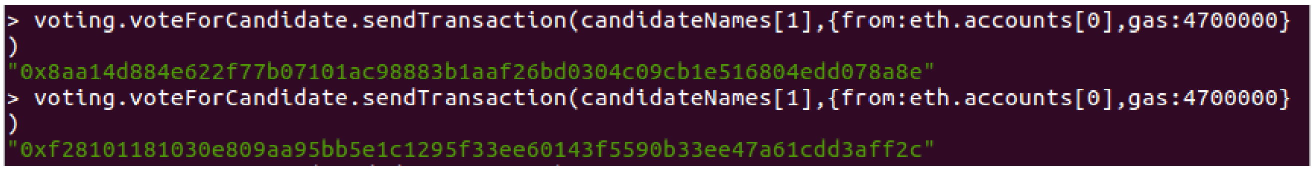
```
> voting.voteForCandidate.sendTransaction(candidateNames[0],{from:eth.accounts[0],gas:4700000}
)
"0x6fa9e137c741367b87e48f53d56015ec00e331f25f11b2c3519837a368540baa"
```

查看 0 号候选人的票数，显示 1 票



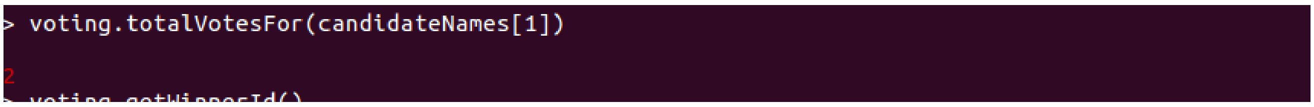
```
> voting.totalVotesFor(candidateNames[0])

1
```

给 1 号候选人及 `Lisi` 投两票



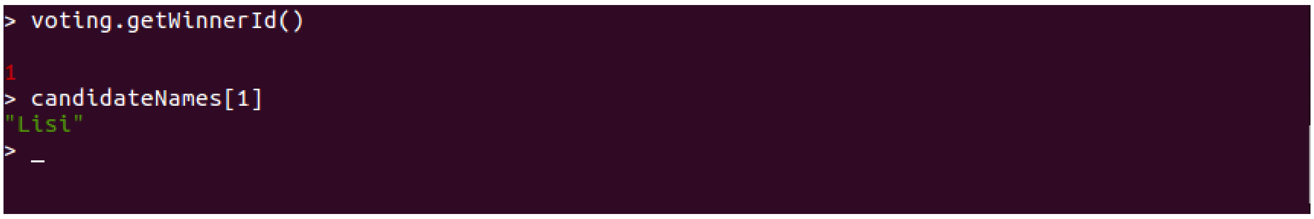
```
> voting.voteForCandidate.sendTransaction(candidateNames[1],{from:eth.accounts[0],gas:4700000}
)
"0x8aa14d884e622f77b07101ac98883b1aaf26bd0304c09cb1e516804edd078a8e"
> voting.voteForCandidate.sendTransaction(candidateNames[1],{from:eth.accounts[0],gas:4700000}
)
"0xf28101181030e809aa95bb5e1c1295f33ee60143f5590b33ee47a61cdd3aff2c"
```

查看 1 号候选人的票数，显示 2 票



```
> voting.totalVotesFor(candidateNames[1])

2
> voting.getWinnerId()
```

或得目前获胜候选人的 序列号，并得到名字 `Lisi`



```
> voting.getWinnerId()

1
> candidateNames[1]
"Lisi"
> _
```