

golang|Java|Python|Node.js

区块链学习框架图

区块链技术概况

区块链进阶

密码学基础

比特币区块链开发

以太坊智能合约开发

超级账本 - *Fabric*

IPFS - 分布式存储

算法



如何编写智能合约(Smart Contract)?(I)



讲师：黎跃春

简介：曾就职于中国石油东方地球物理公司、北京中友瑞飞公司移动业务部 担任 *hybridApp* 研发工程师；*iOS* 资深讲师，*React* 系全栈工程师；15 年接触数字货币，16 年初开始关注区块链技术的发展，16 年中旬开始专注于区块链技术的研究。16 年 12 月创办孔壹学院，旨在专注于『区块链+内容』产品的研发以及对区块链技术的推广和普及。

博客：<http://liyuechun.org>

Github：<http://github.com/liyuechun>

微博：黎跃春-追时间的人



学习目标

1. 了解智能合约
2. 简单环境搭建
3. 能够利用 `solidity` 编写 `Hello World` 合约
4. 合约部署
5. 和合约互动

使用solidity语言撰写智能合约

`Ethereum` 上的智能合约需要使用 `solidity` 语言来撰写。虽然还有其他能用来撰写智能合约的语言如 `Serpent`（类Python）、`l1l`（类Fortran），但目前看到所有公开的智能合约都是使用 `solidity` 撰写。

宣传上说，`solidity` 是一种类似 `Javascript` 的语言，而且围绕着 `solidity` 的各种开发工具链，都是使用属于 `Javascript` 生态系的 `npm` 来提供的。但我觉得 `solidity` 还是比较像Java或C#。因为和 `Javascript` 不同，`solidity` 与Java或C#同属于强类型（Strong Type，在定义变数时需要指定类型）语言、在定义函式（function）时同样需指定回传的类型（type）、同样也需要先编译才能执行。这些特性都是 `Javascript` 所不具备的。

开发前的准备

本文将使用当前最活跃的智能合约开发框架 `truffle` 为基础来开发。

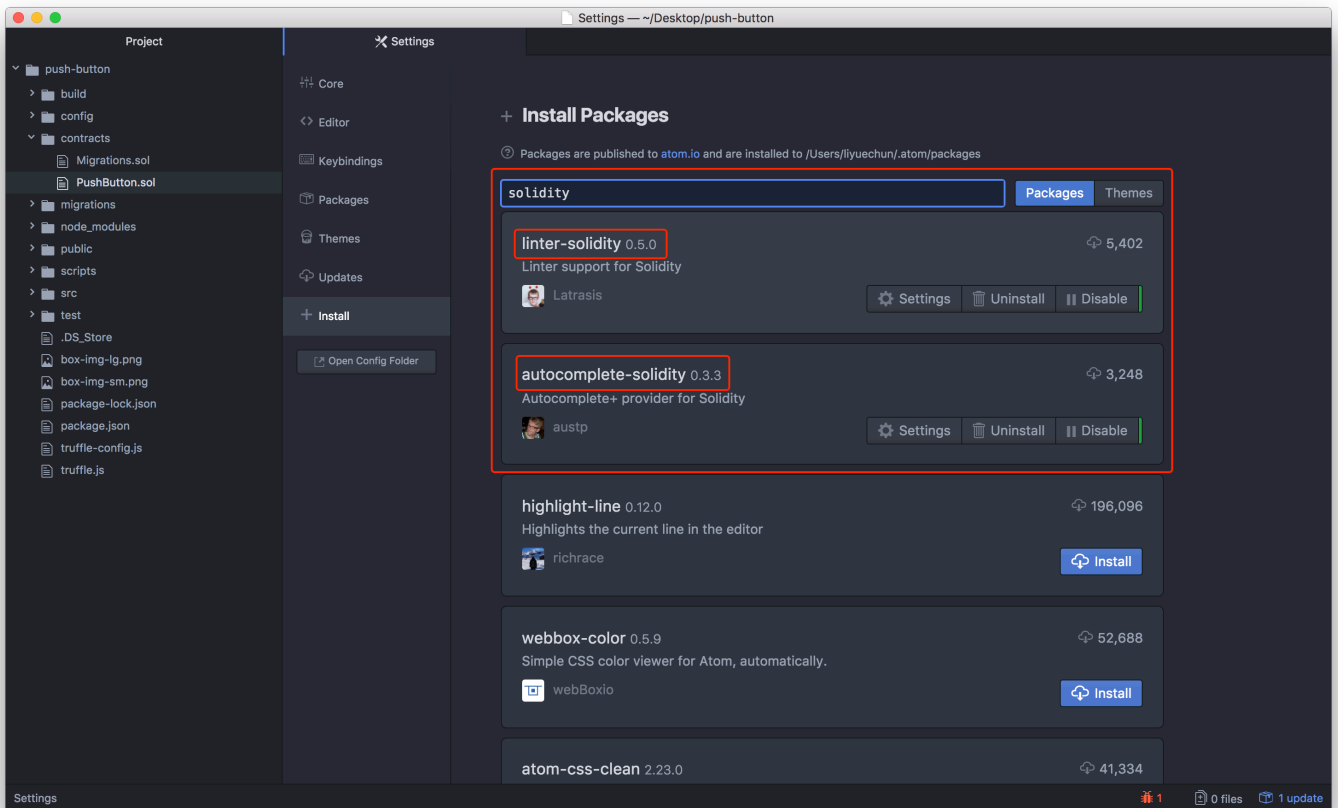
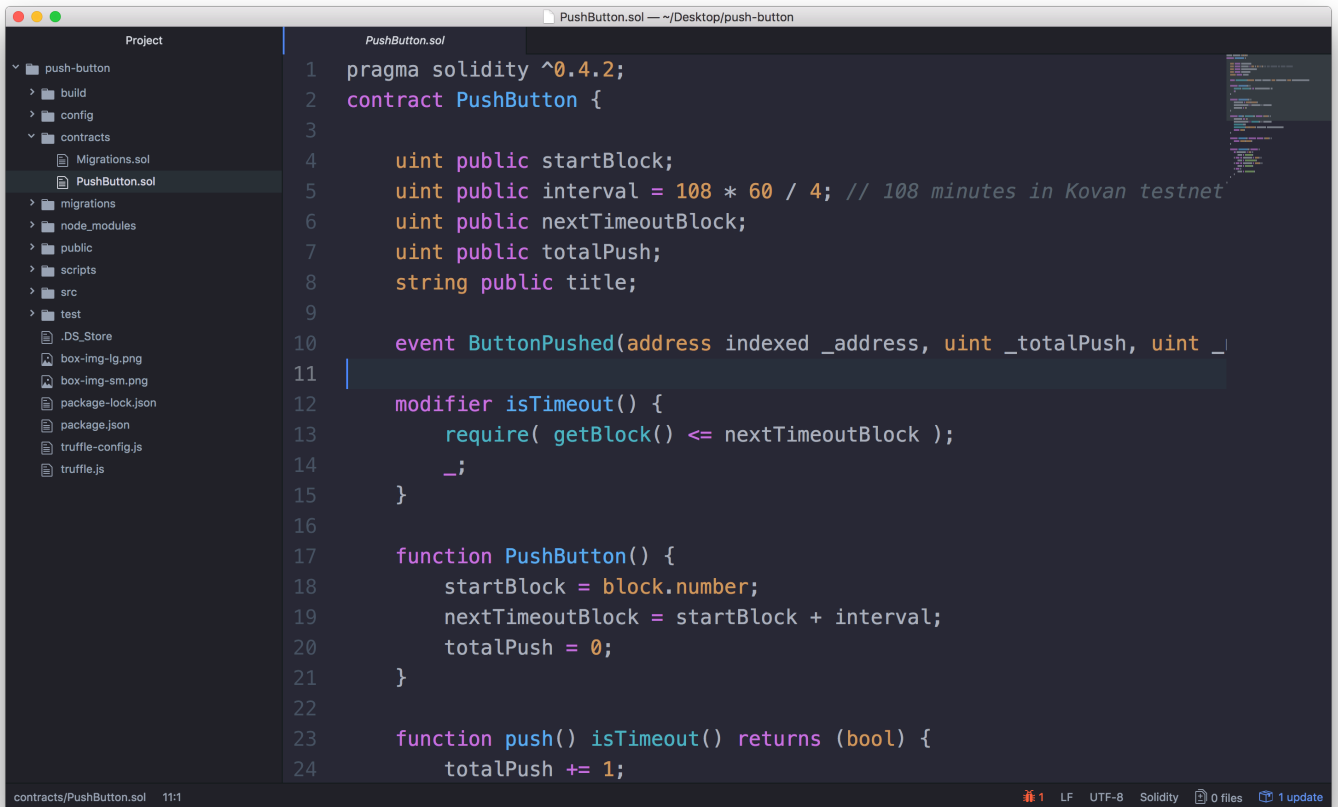
`ENS` (Ethereum Name Service) 也是采用 `truffle` 框架。其他选择还有 `embark` 等。

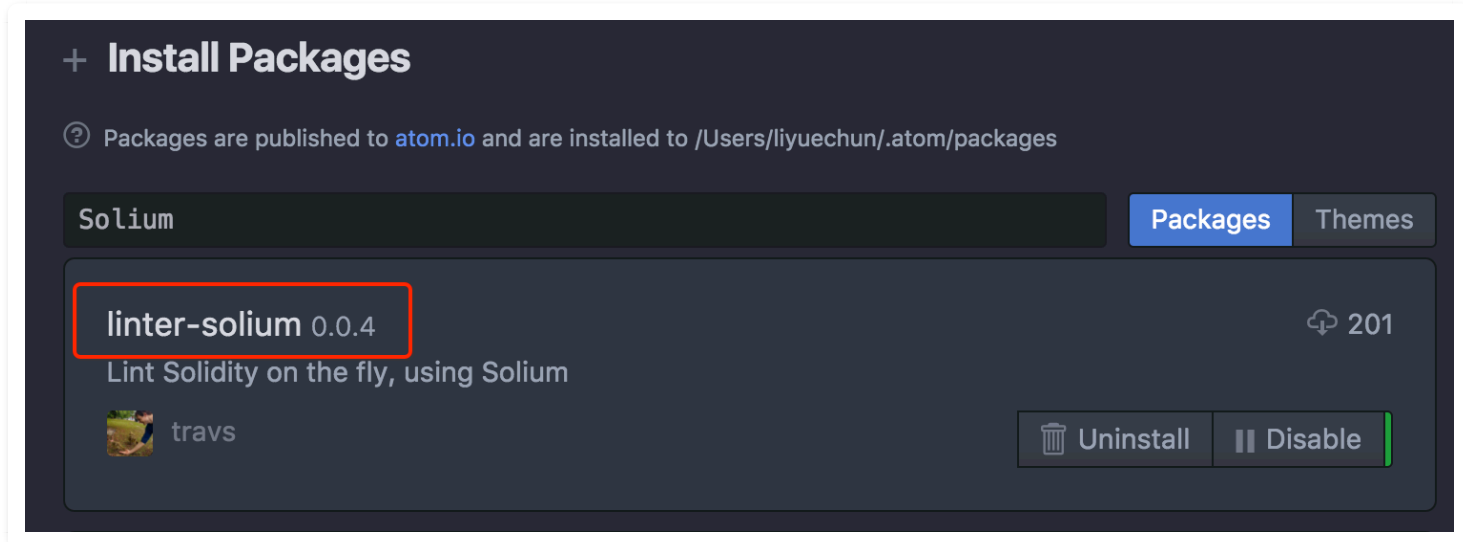
就像一般网站或App开发一样，在提供公开服务之前，开发者会在自己用于写程序的电脑（又称作本机）或透过测试网络来测试程序执行的效果，测试完成后，才会部署到公开的网络上提供服务。开发区块链智能合约（程序）的过程也是如此。特别是公开链上所有写入或读取计算结果的操作都需要真金白银（虚拟代币），而且根据网络状况，每个公开链上的操作都需要要一小段反应时间（15秒~数分钟），这些等待颇浪费宝贵的开发时间🕒。因此在开发的过程中，我们将使用 `testrpc` 工具在电脑上模拟智能合约所需的以太坊内存块链测试环境。

`testrpc` 中也包含了 `Javascript` 版本的 `Ethereum` 虚拟机（Ethereum Virtual Machine），因此可以完整地执行智能合约。

此外，开发前还需准备一个合手的编辑器。我目前是使用 `Atom` 搭配 `solidity` 插件来开发。

`solidity` 插件除了支持语法高亮之外，也会透过 `Solium` 检查并提示基本的语法错误，相当方便。其他编辑器应该也有类似的插件可选择。





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$
```

启动Testrpc

安装好后随时可以使用 `testrpc` 命令来启动以太坊测试环境。

```
liyuechun:~ yuechunli$ testrpc
EthereumJS TestRPC v4.1.3 (ganache-core: 1.1.3)
```

Available Accounts

```
=====
(0) 0xbbd414b340f2255dab9d923428c97f0b65d9df81
(1) 0xe9869e3cf29b6fca81762c314df229c7c4fea25e
(2) 0xc79e72362a4511b9e499d186654332c4d6f569be
(3) 0x9a6f0651907c149d4173c03927144dbbba1473d4
```

```
(4) 0x5b13a5d6788752b26dd4e338aae2e01058ee145e
(5) 0xfc7f56d942ad5260be23ecee92a344aba1b7e7d8
(6) 0xc48dc22c6bacd6ade4421ab54f25bc45c1c51142
(7) 0x3fe2b7d4141dd0a456661f77086d055cbaf3b78f
(8) 0x567979fed26ca85e9d1b4ac919c840e3fc9857e2
(9) 0xb2eafe245f098eef1c2c1f466d9a8dcd58764c62
```

Private Keys

```
=====
(0) 947ab78e91133103612ca099d60e6c38cac5bb769f7f097c82d003cf058500bd
(1) 8ffe0ba8dc53e16944a17ddd3378b5fba0379cd84df4e5237b8b46d05b8762f
(2) ffe2e04e43e4106b247407656f5233bcc3e0c49730972d0df9c1d1093375e2ef
(3) a20e453dc44c76aaca6a22efdbb605c2ed9eea64c11317e683461e11bd105ea7
(4) 4748268ff1b828868dc56d07a1b121b427e1bdede5dbb3c14ef1254d9d26b1a5
(5) f9957e68c6d20d38b81604a0509e6c4591478bc754f87d5682564073705fbb46
(6) 34e648b23c0ace6b2b0893651d87f70be8496f97ecf6b7b4607b2acc4e05c9bd
(7) d2477cedec217e3fb19a5981dafbc125ef66ccc9dc7df29301d08a24da843cf5
(8) d319f85ccd80e55b2e707e05f09662632564c297248f8b96f82ea5eeaeef0851
(9) 88c33ac9f1062b82f9e82f86a0ce307e3bd8fcf683b9751232c2f193f5bdc668
```

HD Wallet

```
=====
Mnemonic:      hire custom clinic expect fury fantasy try dress source spy viable
flag
Base HD Path:  m/44'/60'/0'/0/{account_index}
```

Listening on localhost:8545

可以看到 `testrpc` 启动后自动建立了 `10` 个帐号 (Accounts)，与每个帐号对应的私钥 (Private Key)。每个帐号中都有 `100` 个测试用的以太币 (Ether)。要注意 `testrpc` 仅运行在内存中，因此每次重开时都会回到全新的状态。

一切准备就绪，我们可以开始建立第一份智能合约项目了。

建立项目

开启另一个终端窗口，输入以下命令以建立项目：

```
liyuechun:Desktop yuechunli$ mkdir SmartContractDemo
liyuechun:Desktop yuechunli$ cd SmartContractDemo/
liyuechun:SmartContractDemo yuechunli$ mkdir HelloWorld
liyuechun:SmartContractDemo yuechunli$ cd HelloWorld/
liyuechun:HelloWorld yuechunli$ truffle init
```

Downloading project...
Project initialized.

Documentation: <http://truffleframework.com/docs>

Commands:

```
Compile: truffle compile
Migrate: truffle migrate
Test:    truffle test
```

```
liyuechun:HelloWorld yuechunli$ ls
contracts  migrations  test          truffle.js
```

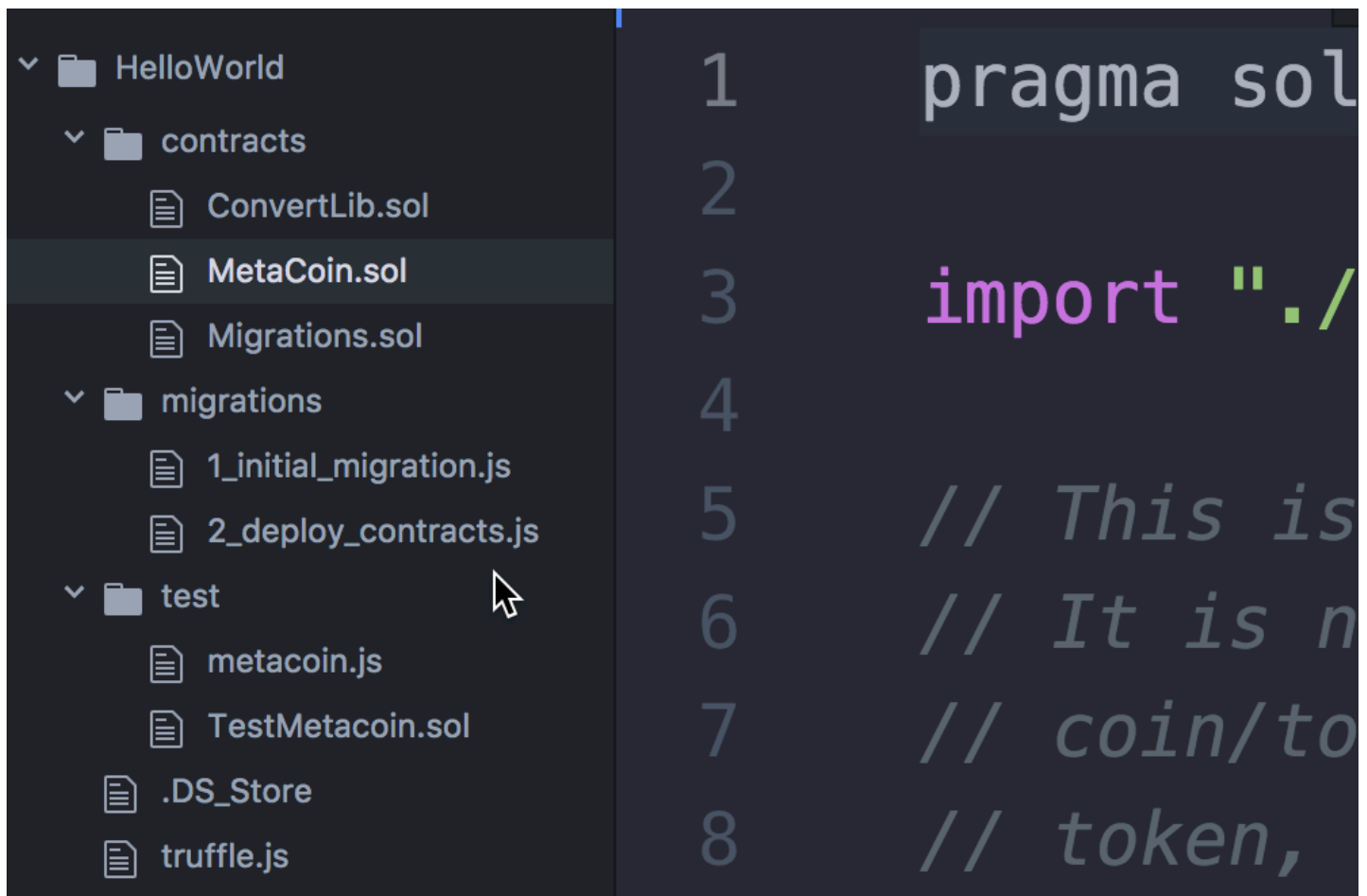
目录结构:

/contracts: 存放智能合约原始代码的地方，可以看到里面已经有三个 `.sol` 文件，我们开发的 `HelloWorld.sol` 文件就存放在这里。

/migrations: 这是 `Truffle` 用来部署智能合约的功能，待会儿我们会修改 `2_deploy_contracts.js` 来部署 `HelloWorld.sol`。

/test: 测试智能合约的代码放在这里，支持 `js` 与 `.sol` 测试。

`truffle.js`: `Truffle` 的设置文档。



新建HelloWorld合约

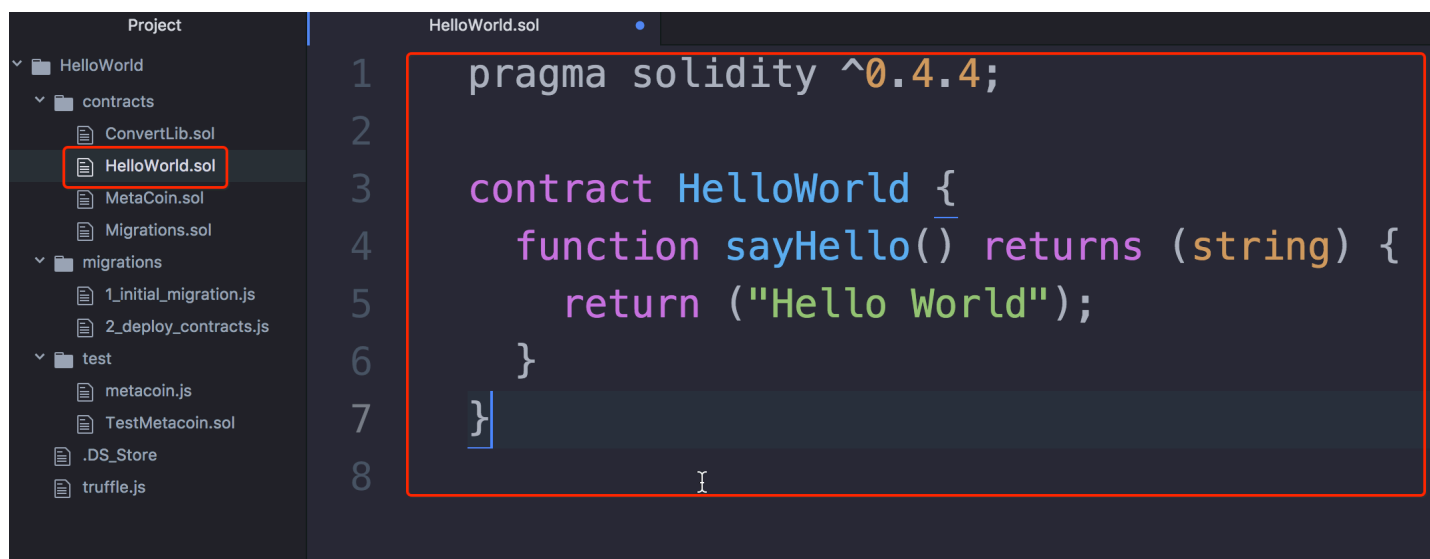
在 `contracts` 文件夹下新建 `HelloWorld.sol` 文件，当然也可以直接在 `HelloWorld` 路径下面直接执行 `truffle create contract HelloWorld` 命令来创建 `HelloWorld.sol`。

```
liyuechun:HelloWorld yuechunli$ ls
contracts      migrations    test          truffle.js
liyuechun:HelloWorld yuechunli$ truffle create contract HelloWorld
liyuechun:HelloWorld yuechunli$ cd contracts/
liyuechun:contracts yuechunli$ ls
ConvertLib.sol HelloWorld.sol MetaCoin.sol Migrations.sol
liyuechun:contracts yuechunli$
```

`HelloWorld.sol` 文件内容如下：

```
pragma solidity ^0.4.4;

contract HelloWorld {
    function sayHello() returns (string) {
        return ("Hello World");
    }
}
```



讲解

```
pragma solidity ^0.4.4;
```

第一行指名目前使用的 `solidity` 版本，不同版本的 `solidity` 可能会编译出不同的 `bytecode`。[^] 代表兼容 `solidity 0.4.4 ~ 0.4.9` 的版本。

```
contract HelloWorld {  
    ...  
}
```

`contract` 关键字类似于其他语言中较常见的 `class`。因为 `solidity` 是专为智能合约（Contract）设计的语言，声明 `contract` 后即内置了开发智能合约所需的功能。也可以把这句理解为 `class HelloWorld extends Contract`。

```
function sayHello() returns (string) {  
    return ("Hello World");  
}
```

函数的结构与其他程序类似，但如果有传入的参数或回传值，需要指定参数或回传值的类型（type）。

编译

现在执行 `truffle compile` 命令，我们可以将 `HelloWorld.sol` 原始码编译成 `Ethereum bytecode`。

```
liyuechun:HelloWorld yuechunli$ ls  
contracts  migrations  test          truffle.js  
liyuechun:HelloWorld yuechunli$ truffle compile  
Compiling ./contracts/ConvertLib.sol...  
Compiling ./contracts/HelloWorld.sol...  
Compiling ./contracts/MetaCoin.sol...  
Compiling ./contracts/Migrations.sol...  
Writing artifacts to ./build/contracts  
  
liyuechun:HelloWorld yuechunli$ ls  
build      contracts  migrations  test          truffle.js  
liyuechun:HelloWorld yuechunli$ cd build/  
liyuechun:build yuechunli$ ls  
contracts  
liyuechun:build yuechunli$ cd contracts/  
liyuechun:contracts yuechunli$ ls  
ConvertLib.json HelloWorld.json MetaCoin.json Migrations.json  
liyuechun:contracts yuechunli$ cat HelloWorld.json  
{  
  "contract_name": "HelloWorld",  
  "abi": [  
    {  
      "inputs": [],  
      "payable": false,
```



```

        "type": "constructor"
    }
],
"unlinked_binary": "0x60606040523415600e57600080fd5b5b5b5b603680601e6000396000f3
0060606040525b600080fd00a165627a7a723058203ee98a767948e9bc08094df4a46ab0361f068b2a
559032cf968df5bbf63e91430029",
"networks": {},
"schema_version": "0.0.5",
"updated_at": 1505805826302
}
liyuechun:contracts yuechunli$

```



编译成功后，会在 `HelloWorld` 文件夹下面的 `build/contracts` 文件夹下面看见 `HelloWorld.json` 文件。

部署

`truffle` 框架中提供了方便部署合约的脚本。打开 `migrations/2_deploy_contracts.js` 文件（脚本使用 `Javascript` 编写），将内容修改如下：

```

var HelloWorld = artifacts.require("HelloWorld");
module.exports = function(deployer) {
    deployer.deploy(HelloWorld);
};

```



使用 `artifacts.require` 语句来取得准备部署的合约。使用 `deployer.deploy` 语句将合约部署到区块链上。这边 `HelloWorld` 是 `contract` 的名称而不是文件名。因此可以用此语法读入任一 `.sol` 文件中的任一合约。

现在执行 `truffle migrate` 命令：

```
liyuechun:HelloWorld yuechunli$ ls
build      contracts  migrations  test      truffle.js
liyuechun:HelloWorld yuechunli$ truffle migrate
Compiling ./contracts/HelloWorld.sol...
Writing artifacts to ./build/contracts

Using network 'development'.

Running migration: 1_initial_migration.js
  Deploying Migrations...
  ... 0x218431f16a5cad6347449808d981887c90b3872898af7cc9dc9b3280c07c184
  Migrations: 0x64e9673cf962d21642a08635e6654fb7f2ea9bcd
Saving successful migration to network...
  ... 0xd9ec788c106df36b8491c95a0ab02ff1e5ef22c1965c910a2576e8259a00535c
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Deploying HelloWorld...
  ... 0x17774b4914d7bc7ab2505a53c59bda6a1fce30c9839d19d735290ca9140450ea
  HelloWorld: 0x471a22ffe2bddd02e82853059871067e4c07a7f4
Saving successful migration to network...
  ... 0xe5e2e11cf5a63ca4517221c68dadb3cae2ca42cbfed93c09c575b6d5f275fc8b
Saving artifacts...
liyuechun:HelloWorld yuechunli$
```

如此一来合约已经部署到 `testrpc` 中。切换到 `testrpc` 窗口，可以看到 `testrpc` 有反应了。

```
liyuechun — node /usr/local/bin/testrpc — 91x41
Block Number: 1
Block Time: Tue Sep 19 2017 15:41:58 GMT+0800 (CST)

eth_newBlockFilter
eth_getFilterChanges
eth_getTransactionReceipt
eth_getCode
eth_uninstallFilter
eth_sendTransaction

Transaction: 0xd9ec788c106df36b8491c95a0ab02ff1e5ef22c1965c910a2576e8259a00535c
Gas usage: 41965
Block Number: 2
Block Time: Tue Sep 19 2017 15:41:58 GMT+0800 (CST)

eth_getTransactionReceipt
eth_accounts
net_version
net_version
eth_sendTransaction

Transaction: 0x17774b4914d7bc7ab2505a53c59bda6a1fce30c9839d19d735290ca9140450ea
Contract created: 0x471a22ffe2bddd02e82853059871067e4c07a7f4
Gas usage: 138581
Block Number: 3
Block Time: Tue Sep 19 2017 15:41:58 GMT+0800 (CST)

eth_newBlockFilter
eth_getFilterChanges
eth_getTransactionReceipt
eth_getCode
eth_uninstallFilter
eth_sendTransaction

Transaction: 0xe5e2e11cf5a63ca4517221c68dadb3cae2ca42cbfed93c09c575b6d5f275fc8b
Gas usage: 26965
Block Number: 4
Block Time: Tue Sep 19 2017 15:41:58 GMT+0800 (CST)

eth_getTransactionReceipt
```

与合约互动

`truffle` 提供命令行工具，执行 `truffle console` 命令后，可用 `Javascript` 来和刚刚部署的合约互动。

```
liyuechun:HelloWorld yuechunli$ ls
build      contracts  migrations  test      truffle.js
liyuechun:HelloWorld yuechunli$ truffle console
truffle(development)> HelloWorld.deployed().then(instance => contract = instance)
TruffleContract {
  constructor:
    { [Function: TruffleContract]
      _static_methods:
        { setProvider: [Function: setProvider],
```

[illegible]


```
clone: [Function: bound clone],
addProp: [Function: bound addProp],
toJSON: [Function: bound toJSON],
web3:
  Web3 {
    _requestManager: [Object],
    currentProvider: [Object],
    eth: [Object],
    db: [Object],
    shh: [Object],
    net: [Object],
    personal: [Object],
    bzz: [Object],
    settings: [Object],
    version: [Object],
    providers: [Object],
    _extend: [Object] },
class_defaults:
  { from: '0xbbd414b340f2255dab9d923428c97f0b65d9df81',
    gas: 4712388,
    gasPrice: 1000000000000 },
currentProvider:
  HttpProvider {
    host: 'http://localhost:8545',
    timeout: 0,
    send: [Function],
    sendAsync: [Function],
    _alreadyWrapped: true },
network_id: '1505794143155' },
abi:
[ { constant: false,
  inputs: [],
  name: 'sayHello',
  outputs: [Array],
  payable: false,
  type: 'function' } ],
contract:
  Contract {
    _eth:
      Eth {
        _requestManager: [Object],
        getBalance: [Object],
        getStorageAt: [Object],
        getCode: [Object],
        getBlock: [Object],
        getUncle: [Object],
        getCompilers: [Object],
        getBlockTransactionCount: [Object],
        getBlockUncleCount: [Object],
        getTransaction: [Object],
        getTransactionFromBlock: [Object],
        getTransactionReceipt: [Object],
        getTransactionCount: [Object],
        call: [Object],
```



```

    estimateGas: [Object],
    sendRawTransaction: [Object],
    signTransaction: [Object],
    sendTransaction: [Object],
    sign: [Object],
    compile: [Object],
    submitWork: [Object],
    getWork: [Object],
    coinbase: [Getter],
    getCoinbase: [Object],
    mining: [Getter],
    getMining: [Object],
    hashrate: [Getter],
    getHashrate: [Object],
    syncing: [Getter],
    getSyncing: [Object],
    gasPrice: [Getter],
    getGasPrice: [Object],
    accounts: [Getter],
    getAccounts: [Object],
    blockNumber: [Getter],
    getBlockNumber: [Object],
    protocolVersion: [Getter],
    getProtocolVersion: [Object],
    iban: [Object],
    sendIBANTransaction: [Function: bound transfer] },
transactionHash: null,
address: '0x471a22ffe2bddd02e82853059871067e4c07a7f4',
abi: [ [Object] ],
sayHello:
  { [Function: bound ]
    request: [Function: bound ],
    call: [Function: bound ],
    sendTransaction: [Function: bound ],
    estimateGas: [Function: bound ],
    getData: [Function: bound ],
    '': [Circular] },
  allEvents: [Function: bound ] },
sayHello:
  { [Function]
    call: [Function],
    sendTransaction: [Function],
    request: [Function: bound ],
    estimateGas: [Function] },
  sendTransaction: [Function],
  send: [Function],
  allEvents: [Function: bound ],
  address: '0x471a22ffe2bddd02e82853059871067e4c07a7f4',
  transactionHash: null }
truffle(development)> contract.sayHello.call()
'Hello World'
truffle(development)>

```

讲解

```
HelloWorld.deployed().then(instance => contract = instance)
```

`truffle console` 中预载了 `truffle-contract` 函数库，以方便操作部署到区块链上的合约。

这边使用 `HelloWorld.deployed().then` 语句来取得 `HelloWorld` 合约的 `Instance`（实例），并存到 `contract` 变量中，以方便后续的调用。

上面用的是 `Javascript ES6+` 的语法，这句也可以写成：

```
HelloWorld.deployed().then(instance => {  
    contract = instance  
});
```

还可以用ES5的写法：

```
HelloWorld.deployed().then(function(instance) {  
    hello = instance;  
});
```

```
truffle(development)> contract.sayHello.call()  
'Hello World'
```

这里直接呼叫 `contract.sayHello()` 也会得到一样的结果。`truffle-contract` 提供使用 `call()` 来读取只读（read only）的数据，这样就不需提供 `gas`。因此如果遇到的操作需要向区块链写入数据，我们就不能用 `call` 语句了。

如此一来，我们已写好并部署完成了第一个智能合约，也验证了合约确实可以运作。

加入新方法

我们在 `HelloWorld.sol` 中再加入一个 `echo` 方法，`echo` 方法接受输入一个参数，并回传传送的参数。

```
function echo(string name) constant returns (string) {  
    return name;  
}
```

新的 `echo` 方法中传入了一个 `name` 参数。我们也为 `echo` 方法加入一个 `constant` 声明，表示调用这个方法并不会改变区块链的状态。如此一来，透过 `truffle-contract` 来调用此方法时，会自动选用 `call` 来呼叫，也不需要额外提供gas。

由于更新了合约内容，我们需要先重新编译一次，将编译结果部署到 `testrpc` 上，再透过 `truffle console` 执行看看结果。

```
liyuechun:HelloWorld yuechunli$ ls
build      contracts  migrations  test      truffle.js
liyuechun:HelloWorld yuechunli$ truffle compile
Compiling ./contracts/HelloWorld.sol...
Writing artifacts to ./build/contracts

liyuechun:HelloWorld yuechunli$ truffle migrate --reset
Using network 'development'.

Running migration: 1_initial_migration.js
  Replacing Migrations...
  ... 0x64cdc42e08a7e3f8070c46d4877ba246d95cbbccbfef1b9abd2450cfc02b48eda
  Migrations: 0x42843f6a470b84e2669f19686a223c1bdefb6f4d
Saving successful migration to network...
  ... 0x57042b767c0f40a4f88ce855e39549010d6d5ae5f880771a45c1f7f36ea0e5b3
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Replacing HelloWorld...
  ... 0x2330e3264aae9d6be3744d8fc71d235fc6dc2934d33ff5159ae209df4cf8f12b
  HelloWorld: 0xbf68789cdd6be1577339e8c739abfa1190c31b6c
Saving successful migration to network...
  ... 0xb4b706d7004654215067ea5954a32f0562b32724d1c646dc43b688b466b10159
Saving artifacts...
liyuechun:HelloWorld yuechunli$ truffle console
truffle(development)> let contract
undefined
truffle(development)> HelloWorld.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],
      abi: [Object],
      network: [Function: network],
      networks: [Function: networks],
      address: [Object],
      links: [Function: links],
      events: [Function: events],
      binary: [Function: binary],
      unlinked_binary: [Object],
      schema_version: [Function: schema_version],
      updated_at: [Function: updated_at] },
  _property_values: {},
  _json:
    { contract_name: 'HelloWorld',
      default_network: undefined,
      abi: [Array],
      unlinked_binary: '0x6060604052341561000f57600080fd5b5b61022c8061001f600039
6000f300606060405263fffffffff7c0100000000000000000000000000000000000000000000
000000600035041663ef5fb05b8114610048578063f15da729146100d3575b600080fd5b3415610053
57600080fd5b61005b61019c565b604051602080825281908101838181518152602001915080519060
20019080838360005b838110156100985780820151818401525b60200161007f565b50505050905090
810190601f1680156100c55780820380516001836020036101000a031916815260200191505b509250
505060405180910390f35b34156100de57600080fd5b61005b60046024813581810190830135806020
601f820181900481020160405190810160405281815292919060208401838380828437509496506101
de95505050505050565b60405160208082528190810183818151815260200191508051906020019080
838360005b838110156100985780820151818401525b60200161007f565b5050505090509081019060
1f1680156100c55780820380516001836020036101000a031916815260200191505b50925050506040
5180910390f35b6101a46101ee565b60408051908101604052600b81527f48656c6c6cf20576f726c64
00000000000000000000000000000000000000000000000000000000602082015290505b90565b6101e66101ee565b50
805b919050565b602060405190810160405260008152905600a165627a7a723058208fec8695bd430e
ed53d1591d841c6e80b1a32a91caab996bb270d54425ebd7140029',
      networks: [Object],
      schema_version: '0.0.5',
      updated_at: 1505809278810 },
  setProvider: [Function: bound setProvider],
  new: [Function: bound new],
  at: [Function: bound at],
  deployed: [Function: bound deployed],
  defaults: [Function: bound defaults],
  hasNetwork: [Function: bound hasNetwork],
  isDeployed: [Function: bound isDeployed],
  detectNetwork: [Function: bound detectNetwork],
  setNetwork: [Function: bound setNetwork],
  resetAddress: [Function: bound resetAddress],
  link: [Function: bound link],
  clone: [Function: bound clone],
  addProp: [Function: bound addProp],
  toJSON: [Function: bound toJSON],
  web3:
    Web3 {
```

```

    _requestManager: [Object],
    currentProvider: [Object],
    eth: [Object],
    db: [Object],
    shh: [Object],
    net: [Object],
    personal: [Object],
    bzz: [Object],
    settings: [Object],
    version: [Object],
    providers: [Object],
    _extend: [Object] },
class_defaults:
  { from: '0xbbd414b340f2255dab9d923428c97f0b65d9df81',
    gas: 4712388,
    gasPrice: 1000000000000 },
currentProvider:
  HttpProvider {
    host: 'http://localhost:8545',
    timeout: 0,
    send: [Function],
    sendAsync: [Function],
    _alreadyWrapped: true },
network_id: '1505794143155' },
abi:
[ { constant: false,
  inputs: [],
  name: 'sayHello',
  outputs: [Array],
  payable: false,
  type: 'function' },
  { constant: true,
  inputs: [Array],
  name: 'echo',
  outputs: [Array],
  payable: false,
  type: 'function' } ],
contract:
  Contract {
    _eth:
      Eth {
        _requestManager: [Object],
        getBalance: [Object],
        getStorageAt: [Object],
        getCode: [Object],
        getBlock: [Object],
        getUncle: [Object],
        getCompilers: [Object],
        getBlockTransactionCount: [Object],
        getBlockUncleCount: [Object],
        getTransaction: [Object],
        getTransactionFromBlock: [Object],
        getTransactionReceipt: [Object],
        getTransactionCount: [Object],

```



```
    call: [Object],
    estimateGas: [Object],
    sendRawTransaction: [Object],
    signTransaction: [Object],
    sendTransaction: [Object],
    sign: [Object],
    compile: [Object],
    submitWork: [Object],
    getWork: [Object],
    coinbase: [Getter],
    getCoinbase: [Object],
    mining: [Getter],
    getMining: [Object],
    hashrate: [Getter],
    getHashrate: [Object],
    syncing: [Getter],
    getSyncing: [Object],
    gasPrice: [Getter],
    getGasPrice: [Object],
    accounts: [Getter],
    getAccounts: [Object],
    blockNumber: [Getter],
    getBlockNumber: [Object],
    protocolVersion: [Getter],
    getProtocolVersion: [Object],
    iban: [Object],
    sendIBANTransaction: [Function: bound transfer] },
transactionHash: null,
address: '0xbf68789cdd6be1577339e8c739abfa1190c31b6c',
abi: [ [Object], [Object] ],
sayHello:
  { [Function: bound ]
    request: [Function: bound ],
    call: [Function: bound ],
    sendTransaction: [Function: bound ],
    estimateGas: [Function: bound ],
    getData: [Function: bound ],
    '': [Circular] },
echo:
  { [Function: bound ]
    request: [Function: bound ],
    call: [Function: bound ],
    sendTransaction: [Function: bound ],
    estimateGas: [Function: bound ],
    getData: [Function: bound ],
    string: [Circular] },
  allEvents: [Function: bound ] },
sayHello:
  { [Function]
    call: [Function],
    sendTransaction: [Function],
    request: [Function: bound ],
    estimateGas: [Function] },
echo:
```

```
{ [Function]
  call: [Function],
  sendTransaction: [Function],
  request: [Function: bound ],
  estimateGas: [Function] },
sendTransaction: [Function],
send: [Function],
allEvents: [Function: bound ],
address: '0xbf68789cdd6be1577339e8c739abfa1190c31b6c',
transactionHash: null }
truffle(development)> contract.echo("春哥微信:liyc1215")
'春哥微信:liyc1215'
truffle(development)>
```

`echo` 方法确实将我们输入的内容回传了。同时因为声明了 `constant`，我们不需要直接调用 `call()` 方法，`truffle` 会自动选用 `call` 来呼叫。

另一点需要注意的，是这次如果还是用 `truffle migrate` 命令，我们会得到如下信息：

```
$ truffle migrate
Using network 'development'.
Network up to date.
```

`Truffle` 会告诉你现在网络上的合约都已是最新的，但事实上刚刚程序中新增的方法并没有更新到内存块链上。要更新内存块链上已部署的程序，需要改写 `migrations` 中的脚本，但现在还不到介绍 `migration` 的时候。还好我们开发用的内存块链是怎么修改都没关系的 `testrpc`，可以使用 `truffle migrate --reset` 命令直接重新在 `testrpc` 上部署一次。

总结

这篇文章非常简单，通过这篇文章，你将掌握如何配置开发环境、如何创建新项目、如何编译、如何部署合约以及了解整个智能合约开发的流程。

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

- 「区块链部落」官方公众号



长按，识别二维码，加关注

参考资料

- [1] Solidity <http://solidity.readthedocs.io/en/latest/index.html>
- [2] Solidity線上編輯器 <https://ethereum.github.io/browser-solidity/>
- [3] Truffle Framework <http://truffleframework.com/>
- [4] Embark Framework <https://github.com/iurimatias/embark-framework>
- [5] ENS也使用Truffle框架 <https://github.com/ethereum/ens>
- [6] <https://github.com/ethereumjs/testrpc>
- [7] <https://github.com/ethereumjs/ethereumjs-vm>
- [8] HelloWorld範例修改自 <https://app.pluralsight.com/library/courses/blockchain-fundamentals/>
- [9] Truffle issue on windows <http://truffleframework.com/docs/advanced/configuration#resolving-naming-conflicts-on-windows>
- [10] <https://medium.com/taipei-ethereum-meetup/ethereum-dapp-tutorial-push-button-cae3810086a4>
- [11] Solidity支援的型別(Type) <https://solidity.readthedocs.io/en/develop/types.html>
- [12] Solium syntax check <https://github.com/duaraghav8/Solium>
- [13] <https://medium.com/taipei-ethereum-meetup/如何撰寫智能合約-smart-contract-i-363d06b1965b>
- [14] http://truffleframework.com/docs/getting_started/contracts