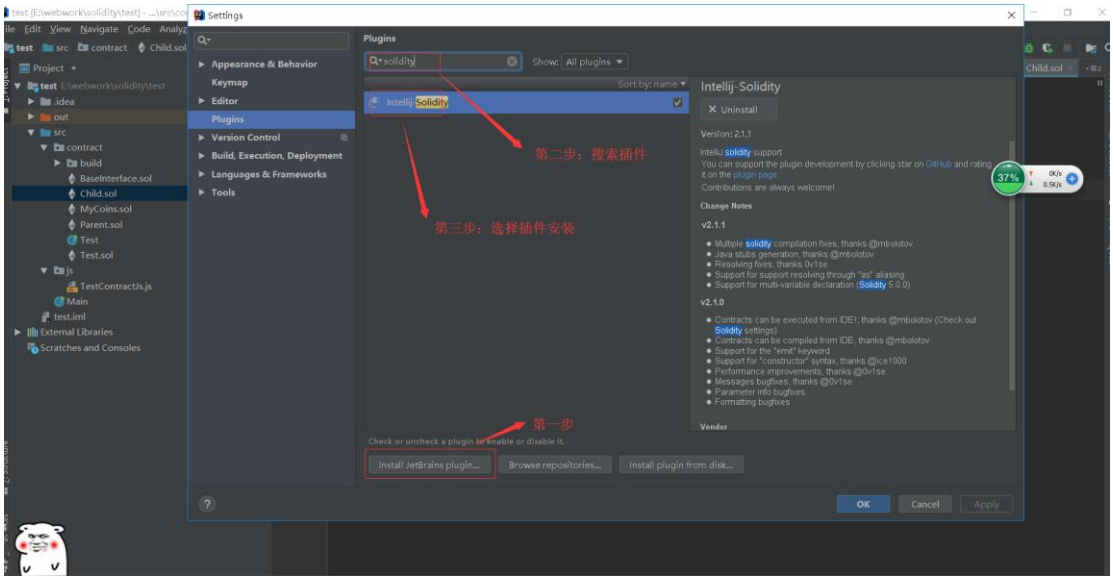
智能合约开发编译部署手册

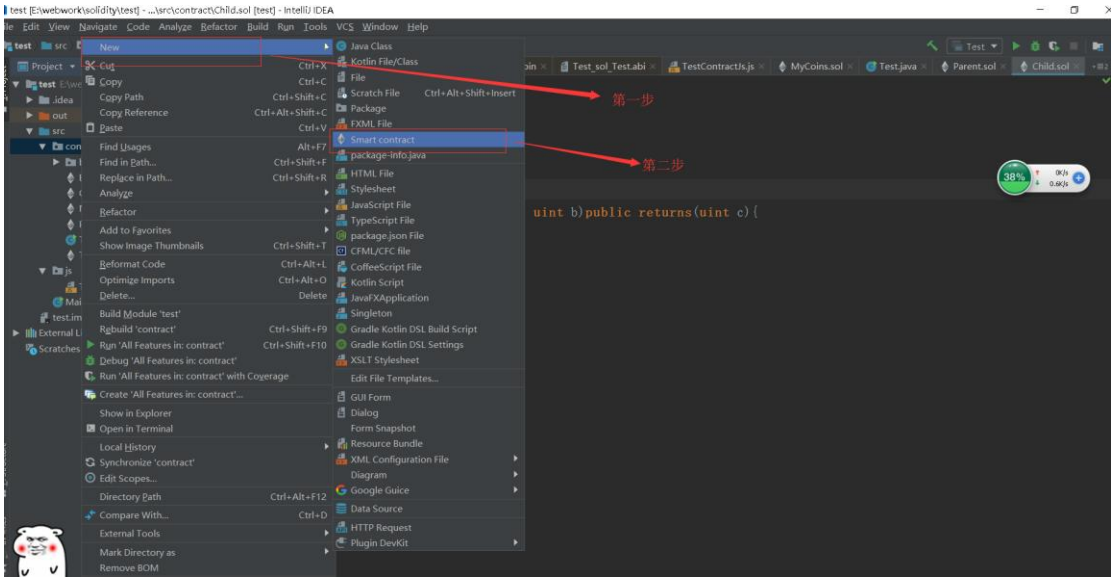
一：开发

1：开发工具　在线开发，编译，调试工具　http://remix.ethereum.org

本地 Intellij IDEA 安装插件



2:使用 idea 开发　新建智能合约



二：编译和发布环境

1：安装 node

Windows 下安装

1：下载 node.js 安装包 node-v10.3.0-x64.msi

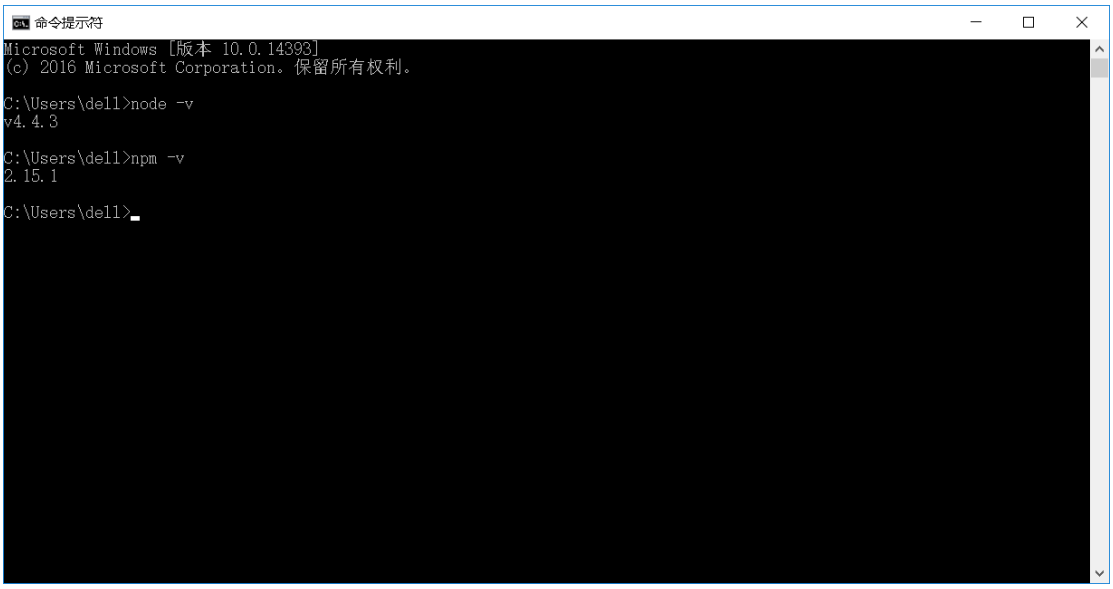
下载地址：http://nodejs.cn/download/

2：一直下一步就好，直到安装完成

3：打开 cmd 运行下面命令，出现如图所示，安装成功

node -v

npm -v



Linux 下安装

1:下载 node.js 安装包 node-v10.3.0-linux-x64.tar.xz

下载地址：http://nodejs.cn/download/

2：解压安装包 node-v10.3.0-linux-x64.tar.xz

```
root@Graphene3:/usr/local/nodejs# tar -xvf node-v10.3.0-linux-x64.tar.xz
```

3：重命名解压文件名

```
root@Graphene3:/usr/local/nodejs# mv node-v10.3.0-linux-x64 node
```

4：配置环境变量

```
root@Graphene3:/usr/local/nodejs# vim /etc/profile
```

```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if [ "$PS1" ]; then
  if [ "$BASH" ] && [ "$BASH" != "/bin/sh" ]; then
    # The file bash.bashrc already sets the default PS1.
    # PS1='\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi
export NODE_HOME=/usr/local/nodejs/node
export PATH=$NODE_HOME/bin:$PATH
~
~
~
~
```

新增加的环境变量

2：安装 solcjs （通过 solcjs 编译智能合约源码，编译成 abi 接口文件（json 格式）和 bin 二进制源文件）

npm install -g solc (参数 -g 表示全局安装，如果不全局安装，就需要切换到安装目录才能使用 solcjs 命令)

安装完毕后，编译智能合约

solcjs --abi --bin Xxxxx.sol -o outPath

--abi 参数表示要输出 abi 接口文件

--bin 参数表示要输出 bin 二进制源文件

Xxxxx.sol 表示需要编译的智能合约

-o 表示 abi，bin 文件输出的位置是 outPath

```
E:\webwork\solidity\test\src\contract>solcjs --abi --bin MyCoins.sol Test.sol -o build
```

3：发布和调用使用 web3.js

1：安装 web3.js

npm install web3

2：发布智能合约

编写 js 脚本或者直接使用 node 控制台

编写 js 脚本方式

```
//引用web3模块
var Web3 = require('web3');
//引用fs 读取文件模块
var fs = require('fs');
//创建web3 对象（http://lcoalhost:8545 表示geth客户端的rpc地址和端口号）
var web3 = new Web3(new Web3.providers.HttpProvider('http://localhost:8545'));
console.log('OK')
var eth = web3.eth;
//读取bin文件（在数据前拼接上0x）
var data = "0x" + fs.readFileSync("bin\\MyCoins_sol_MyCoins.bin").toString();
//读取abi文件
var abi = JSON.parse(fs.readFileSync("abi\\MyCoins_sol_MyCoins.abi").toString());
//发布智能合约的账户（运行该脚本时，该账户需要解锁）
var coinbase = '0xfbb85212a2e8ba40decfd325beebd5be49456569';
//创建智能合约
var client = new web3.eth.Contract(abi);
//部署智能合约（需要发送交易，send的内容就是构建的交易）
//deploy的参数  data：二进制源文件（上面的data）  arguments：智能合约构造函数的参数，有几个传几个，没有则不传
//后面的on都是监听的回调函数
client.deploy({data:data ,
               arguments: ["wwlCoin", 'WWLC' , 10000000000]
               }).send({
                        from : coinbase ,
                         gas: 5000000,
                        gasPrice: '18000000000'

                        } , function(e , transactions){
}).on('error', function(error){ console.log(error) })
.on('transactionHash', function(transactionHash){ console.log("hash:",transactionHash)})
.on('receipt', function(receipt){console.log(receipt.contractAddress) })
.on('confirmation', function(confirmationNumber, receipt){console.log("receipt,",receipt)})
.then(function(newContractInstance){console.log(newContractInstance.options.address) });
```

3：调用智能合约

编写 js 脚本或者直接使用 node 控制台

使用 node 控制台方式

打开 cmd 切换到 web3 的安装目录（如果 web3 不是全局安装的话）

输入 node 进入 node 控制台，在控制台一步一步执行 JavaScript

```
E:\webwork\web3>node
>
>
>
>
>
>
>
>
>
```

var Web3 = require('web3');

```
> var Web3 = require('web3');
undefined
>
```

var web3 = new Web3(new Web3.providers.HttpProvider('http://localhost:8545'));

```
> var web3 = new Web3(new Web3.providers.HttpProvider('http://localhost:8545'));
undefined
>
```

var fs = require('fs');

```
> var fs = require('fs');
undefined
```

var abi = JSON.parse(fs.readFileSync("abi\\MyCoins_sol_MyCoins.abi").toString());

```
> var abi = JSON.parse(fs.readFileSync("abi\\MyCoins_sol_MyCoins.abi").toString());
undefined
```

var contract = '0x341E18Da4Cc0aCa881B03C8B290df1C58d5AE6F2';//合约地址，成功部署智能合约后会返回合约地址的

var client = new web3.eth.Contract(abi , contractAddress);

```
> var client = new web3.eth.Contract(abi , '0x341E18Da4Cc0aCa881B03C8B290df1C58d5AE6F2');
undefined
```

client.methods.totalSupply().call().then(console.log);//查询方法，不修改状态变量的方法使用 call(),不消耗 gas

client.methods.transfer('0xeecadea2b5b7da58cc75599c8987a172a1347c5d',100).send({from:'0xfbb85212a2e8ba40decfd325beebd5be49456569'}).then(console.log)

//使用 send 发送交易，调用需要消耗 gas 的方法