

# Salary Easy Pay 项目文档

## 项目地址

本项目的 github 地址：

<https://github.com/ousuixin/Salary-Easy-Pay>

## 选题背景

农民工欠薪的新闻屡见不鲜，每到年关，几乎都可以看到类似的新闻。其实不只是农民工被欠薪，各行各业都存在这种现象，甚至于大热的 IT 行业也是如此：之前 17 年初时，“百度太子”李明远投资的斯凯智能科技公司（一家无人机公司）倒闭，拖欠员工 200 多万工资。

这样的例子还有很多，尽管员工可以通过法律途径来讨回工资，但是究竟还是要付出相当的成本，包括时间、经济代价：向法院提起申诉到法院下达判决需要较长的一段时间，而对于一些员工而言可能急需工资（特别是临近过年的时候），所以这种代价可能是难以承受的。并且请律师需要付出额外的经济成本，设想如果能够有什么方法能够使公司按时结清工资，那么自然就不需要付出这些额外的时间经济成本了。然而很可惜现在好像还没有什么有效的措施能够做到这一点。

如果结合区块链技术，构建月结系统，通过智能合约自动从公司账户划分工资到员工账户，就能够有效避免欠薪情况。该系统可以看作是一个部署在某个公有链或者私有链上的智能合约，公司需要用现实货币购买代币，然后以发放代币的形式发放工资，月结后员工可以将代币兑换为现实货币，以此获得工资。由于区块链智能合约自动执行，这种工资月结是不可阻止的，也就是说不存在公司欠薪情况，除非公司资金无法流转，那么该公司可以自动移交法律处理（破产清算之类）。除此之外，员工可以看到这个公司的历史工资发放情况，由于区块链的数据公开透明性、不可篡改性，所以这些记录一定是真实的，员工可以以这种方式判断该公司的信誉好坏。

关于实现的更多细节如下：

公司招收员工时将员工加入自己的员工列表，工资将每个月自动划扣，如果公司需要裁员，则将员工从该公司员工列表删除，删除同时按照本月上班天数交付一定工资。并且可以结合公司的考核系统：将考核系统写入智能合约，然后由计算机程序自动检测每个员工是否到达考核要求，然后记录到月结工资中，待到月结时同意交付结算后的工资。

总之，个人认为，将工资结算系统结合区块链技术，能够有效避免欠薪现象，缓解企业员工生存压力（让他们不再为欠薪发愁），并促使企业不断向更好的方向发展运营（因为它们想要不因欠薪而破产必须要使自己能够支付员工工资），因为还没看到有类似做法的，所以个人认为“salary easy pay”系统还是有一定前景的。

## 环境说明

本项目开发于 Linux 环境下，使用 webpack 打包，前端界面部分采用 vue 框架

项目所需的局部安装的组件以及版本如下：

Package.json 文件

```
{
  "name": "salaryeasypay-employer",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1",
    "start": "webpack",
    "server": "webpack-dev-server --open"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "babel-core": "^6.26.0",
    "babel-loader": "^7.1.4",
    "css-loader": "^0.28.11",
    "file-loader": "^1.1.11",
    "html-loader": "^0.5.5",
    "html-webpack-plugin": "^3.2.0",
    "style-loader": "^0.20.3",
    "truffle-contract": "^3.0.7",
    "webpack": "^3.0.0",
    "webpack-dev-server": "^2.11.0"
  },
  "dependencies": {}
}
```

其它说明：

truffle 环境配置：见 truffle.config.js

webpack 环境配置：见 webpack.config.js

运行该项目方法：

方法一：下载所有项目文件（包括 node\_module 中的包，总计 80M 左右），然后 npm run start 打包、npm run server 运行项目（localhost:8080）

方法二：该方法只能查看界面。仅需下载 build 下打包好的文件，然后点击 index.html 可以查看界面（无法连接到私链，因为没有后端）

页面说明：

四个页面 index.html、introduction.html、employer.html、employee.html，分别是主页、介绍页面、雇主页面、雇员页面

数据存储说明：

采用浏览器的 localStorage 存储雇主创建的合约，一经创建永久保存在浏览器缓存（除非缓存被清空，或更换设备，这时可以通过 add contract exist 功能重新添加那些之前创建过的合约，这需要雇主自己记清楚合约地址，以防万一出现故障，也能恢复合约。

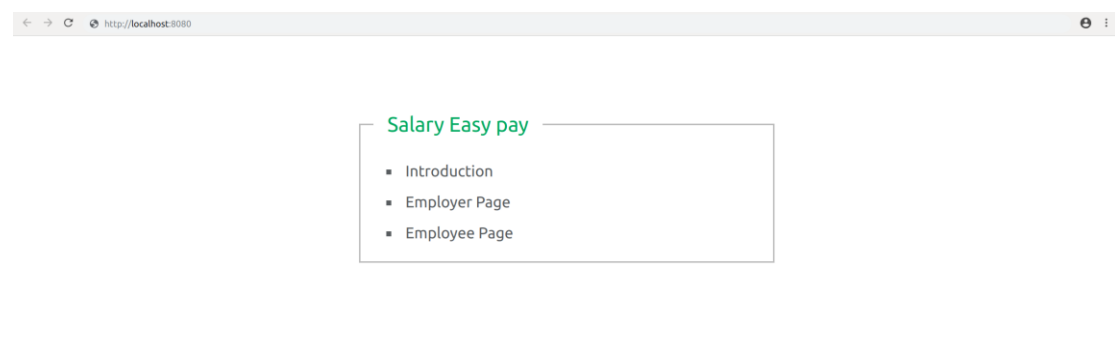
# 使用说明

## 项目运行：

构建好项目（下载好全部源代码文件）、安装好所需组件（需要使用相同版本的组件，否则将可能出错）之后：

- (1) 在项目文件 SalaryEasyPay 下使用 **npm run start** 命令可以进行打包，打包好的文件将在 build 文件夹下，我们可以看到有四个 html 页面的 html 文件、js 文件、png 文件等等
- (2) 然后在项目文件 SalaryEasyPay 下使用 **npm run server** 命令运行 dapp，最终 dapp 将自动在 <http://localhost:8080> 下打开

运行 dapp 知乎我们可以看到如下界面：



说明运行成功，进入主页面。

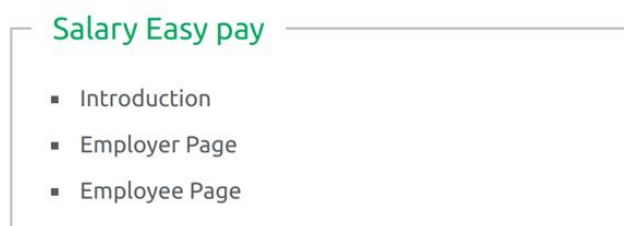
## 页面介绍：

- (1) 上面看到的截图就是我们的主页面，简洁的主页面下是三个页面选项：

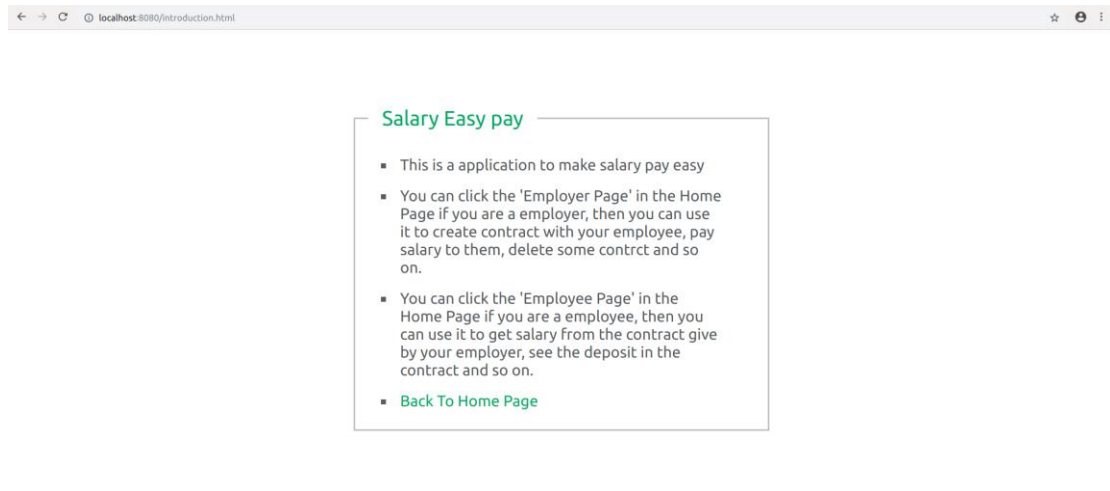
(i) 第一个页面选项“Introduction”是介绍该 dapp 的页面；

(ii) 第二个页面选项“Employer Page”是提供给雇主用户使用的页面；第三个页面选项“Employee Page”是提供给雇员用户使用的页面；

(iii) 点击选项进入相应页面，页面详情见介绍（2）、（3）、（4）。



## (2) 点击主页面下的第一个页面选项“Introduction”我们进入介绍页面：



介绍页面是对该 dapp 功能的介绍，从该页面我们得知这个 app 的使用方法：

(i) 如果用户是一个雇主，那么可以点击主页面下的第二个页面选项“Employer Page”作为雇主，用户可以：

[查询自己账户余额](#)

[创建与指定用户的工资合约](#)

[根据指定合约地址添加已经存在的工资合约](#)（误删合约可以通过地址找回、**浏览器缓存 localStorage 被清空也能通过各个合约地址找回原来的合约**、用户更换设备后新设备 localStorage 为空，则合约列表为空，可以重新通过合约信息找回原来的合约）

[更新自己当前时间应向合约交付的工资](#)（提醒雇主应该交付的工资数额，防止雇主在不知情的情况下违约）

[向指定合约付工资](#)（即给签订该合约的用户付工资）

[删除指定的工资合约](#)（解除雇主雇员关系）

[点击左上角小图标返回主页面](#)

(ii) 如果用户是一个雇员，那么可以点击主页面下的第二个页面选项“Employer Page”作为雇主，用户可以：

[查询自己账户余额](#)

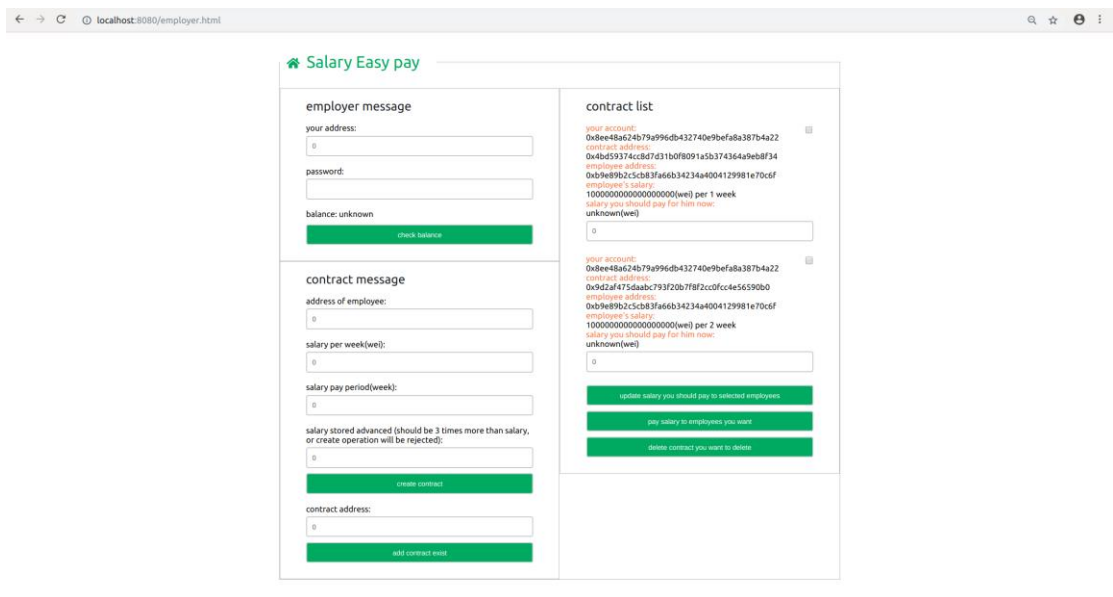
[从指定合约获取工资](#)

[获取指定合约信息](#)（合约余额、工资情况、发薪周期等）

[点击左上角小图标返回主页面](#)

(iii) 点击“Back To Home Page”返回主页面

(3) 点击主页面下的第二个页面选项“Employer Page”我们进入雇主用户页面：



详细截图：（看的更清楚）：

雇主信息板块，包括输入雇主地址、密码、执行（i）操作的 check balance 按钮：

employer message

your address:

password:

balance: unknown

check balance

合约信息板块，包括输入合约的雇员、薪水、发薪周期、预存工资、合约地址、执行（ii）操作的 create contract 按钮、执行（iii）操作的 add contract exist 按钮：

contract message

address of employee:

salary per week(wei):

salary pay period(week):

salary stored advanced (should be 3 times more than salary, or create operation will be rejected):

create contract

contract address:

add contract exist

**合约列表板块：**包括该雇主已经有的合约列表，每一个项对应一个合约，包括（a）合约的雇主地址（一个雇主可以有多个账户，对应多个地址）、雇员地址、合同地址、雇员工资、发薪周期、当前应付薪水等信息；（b）给该合约存入的资金量；（c）分别执行（iv）、（v）、（vi）操作的 update salary you should pay to selected employees 按钮、pay salary to employees you want 按钮、delete contract you want to delete 按钮：

contract list

your account:

0x8ee48a624b79a996db432740e9befa8a387b4a22

contract address:

0x4bd59374cc8d7d31b0f8091a5b374364a9eb8f34

employee address:

0xb9e89b2c5cb83fa66b34234a4004129981e70c6f

employee's salary:

1000000000000000000(wei) per 1 week

salary you should pay for him now:

unknown(wei)

0

your account:

0x8ee48a624b79a996db432740e9befa8a387b4a22

contract address:

0x9d2af475daabc793f20b7f8f2cc0fcc4e56590b0

employee address:

0xb9e89b2c5cb83fa66b34234a4004129981e70c6f

employee's salary:

1000000000000000000(wei) per 2 week

salary you should pay for him now:

unknown(wei)

0

update salary you should pay to selected employees

pay salary to employees you want

delete contract you want to delete

**头部板块：**点击小图标返回主页，即执行（vii）操作的按钮

🏠

Salary Easy pay

employer message

contract list

七个按钮使用具体方法：（具体操作截图见测试部分）

(i) 查询自己账户余额：

输入自己的账户地址， 点击 check balance 按钮即可；

(ii) 创建与指定用户的工资合约：

输入自己的账户地址和相应密码、雇员信息（地址、工资、发薪周期、预存工资），  
点击 create contract 按钮即可；

(iii) 根据指定合约地址添加已经存在的工资合约：

输入自己的账户地址和相应密码、已存在的合约地址， 点击 add contract exist 按钮即可

(iv) 查询自己当前时间应向合约交付的工资（提醒雇主应该交付的工资数额，防止雇主在不知情的情况下违约）：

输入自己的账户地址和相应密码、选中想要执行查询操作的合约， 然后点击 update salary you should pay to selected employees 按钮即可；

(v) 向指定合约付工资（即给签订该合约的用户付工资）：

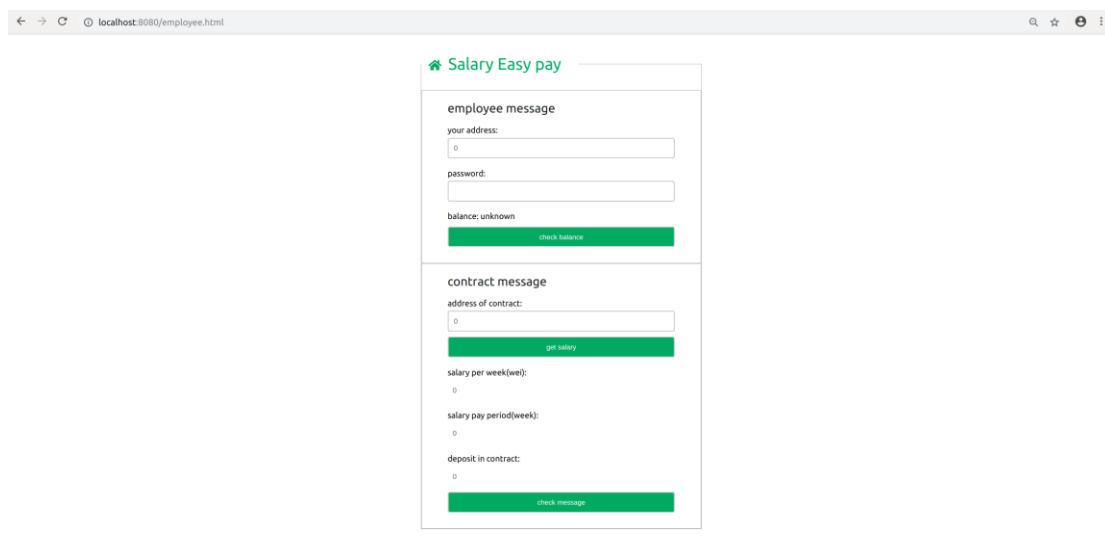
输入自己的账户地址和相应密码、选中想要发薪的合约、在对应的地方输入想要交付的资金， 点击 pay salary to employees you want 按钮即可；

(vi) 删除指定的工资合约（解除雇主雇员关系）：

输入自己的账户地址和相应密码、选中想要删除的合约， 然后点击 delete contract you want to delete 按钮即可

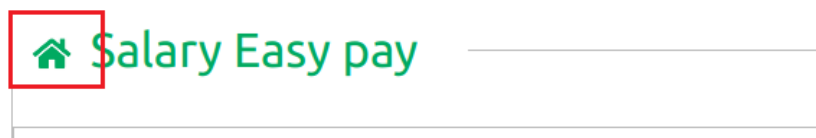
(vii) 点击左上角小图标返回主页面： 点击即可返回主页

#### (4) 点击主页面下的第三个页面选项“Employee Page”我们进入雇员用户页面：

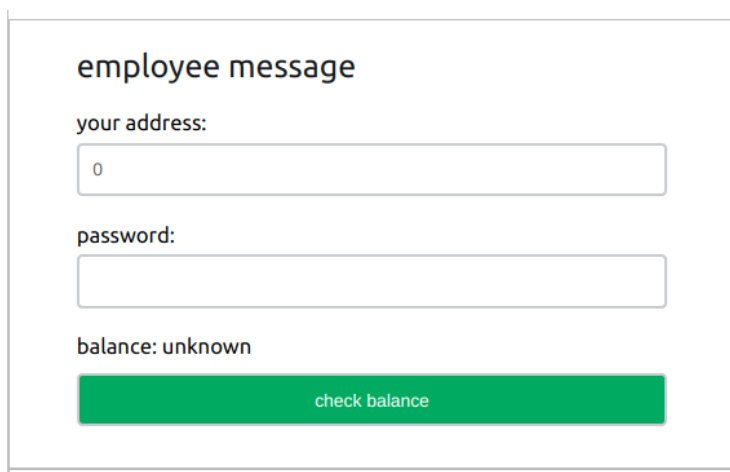


详细截图：（看的更清楚）：

头部板块： 点击小图标返回主页， 即执行（iv）操作的按钮



雇员信息板块，包括输入雇员地址、密码、执行（i）操作的 check balance 按钮：



employee message

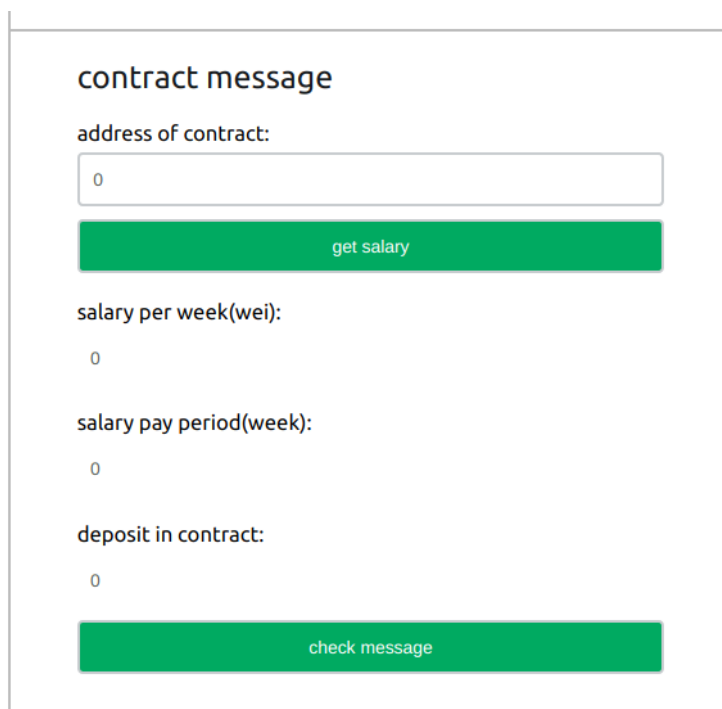
your address:

password:

balance: unknown

check balance

合约信息板块，包括输入合约的地址、输出合约薪水、发薪周期、剩余存款、执行（ii）操作的 get salary 按钮、执行（iii）操作的 check message 按钮：



contract message

address of contract:

get salary

salary per week(wei):

salary pay period(week):

deposit in contract:

check message

四个按钮使用具体方法：（具体操作截图见测试部分）

（i）查询自己账户余额：

输入自己的账户地址，点击 check balance 按钮即可；

（ii）从合约获取工资：

输入自己的账户地址和相应密码、合约地址，点击 get salary 按钮即可；

（iii）根据给定合约地址获取对应工资合约的相关信息（合约余额、工资情况、发薪周期等）：

输入自己的账户地址和相应密码、已存在的合约地址，点击 check message 按钮即可

（iv）点击左上角小图标返回主页面：点击即可返回主页



## 功能测试

测试在 geth 自搭建私链下测试。

所以首先自己使用一个矿工账户为其它 6 个新建的账户各转账 100 个以太币，用于测试。为了控制变量，这六个账户不进行挖矿，所以所有的资金流动全部源自于我们的 dapp 的活动，这样能够简单的看出账户之间的资金流动数额（在本项目中即工资结转和获取情况）

```
> personal.unlockAccount(eth.accounts[0], "123456", 1000000)
true
> eth.sendTransaction({from:eth.accounts[0], to:eth.accounts[2], value:1000000000000000000})
INFO [12-29|06:18:12.056] Submitted transaction          fullhash=0xc8815dfdc5b31a5d03ab6676
baf6e533e03134dec1039fd0b8fa48e1b1b310b7 recipient=0x5324D341E0E9F1119e13F72d1b26cF3552ea1db2
"0xc8815dfdc5b31a5d03ab6676baf6e533e03134dec1039fd0b8fa48e1b1b310b7"
> eth.sendTransaction({from:eth.accounts[0], to:eth.accounts[3], value:1000000000000000000})
INFO [12-29|06:18:21.541] Submitted transaction          fullhash=0x667d3573dbff16b6c950a683
a6d4332f82b1f3cbc99fc56fe2237e3b90f00af0 recipient=0x63511eD9CE5c7CA63431b8EF5413F7fcd37c4845
"0x667d3573dbff16b6c950a683a6d4332f82b1f3cbc99fc56fe2237e3b90f00af0"
> eth.sendTransaction({from:eth.accounts[0], to:eth.accounts[4], value:1000000000000000000})
INFO [12-29|06:18:25.368] Submitted transaction          fullhash=0x0b01ad61906db096aaea3364
640a5fe3a2d966ffa20257342e1383a9916e395d recipient=0xA08962Dcdccd844fD0dD8dbDBe8724d1b83c1606
"0x0b01ad61906db096aaea3364640a5fe3a2d966ffa20257342e1383a9916e395d"
> eth.sendTransaction({from:eth.accounts[0], to:eth.accounts[5], value:1000000000000000000})
INFO [12-29|06:18:29.007] Submitted transaction          fullhash=0x9d6acbddea9e78720e11e5849
ebe417455649bcb253975e9a3e54e176d8840346 recipient=0x011117E215F263A9b50D7dBff8df07be35405c52
"0x9d6acbddea9e78720e11e5849ebe417455649bcb253975e9a3e54e176d8840346"
> eth.sendTransaction({from:eth.accounts[0], to:eth.accounts[6], value:1000000000000000000})
INFO [12-29|06:18:36.652] Submitted transaction          fullhash=0x279ebfe83675b678a1164f0f
37d5cb85ab5b7d59a1215120a27e288c6ed60500 recipient=0x5371f730Da5b446f499659e1Db29a178a9f71a1c
"0x279ebfe83675b678a1164f0f37d5cb85ab5b7d59a1215120a27e288c6ed60500"
> eth.sendTransaction({from:eth.accounts[0], to:eth.accounts[7], value:1000000000000000000})
INFO [12-29|06:18:40.835] Submitted transaction          fullhash=0x69a53594f8f2f844c7d0d724
4cdc0509eda00e82e0e87df01f824f53a24b8df recipient=0xc8460c3c4D163b11ab10242F71Dbe30Cf6Fa4a41
"0x69a53594f8f2f844c7d0d7244cdc0509eda00e82e0e87df01f824f53a24b8df"
```

然后挖矿，使得转账成功：

```
> miner.stop()
null
> web3.fromWei(eth.getBalance(eth.accounts[2]), 'ether')
100
> web3.fromWei(eth.getBalance(eth.accounts[3]), 'ether')
100
> web3.fromWei(eth.getBalance(eth.accounts[4]), 'ether')
100
> web3.fromWei(eth.getBalance(eth.accounts[5]), 'ether')
100
> web3.fromWei(eth.getBalance(eth.accounts[6]), 'ether')
100
> web3.fromWei(eth.getBalance(eth.accounts[7]), 'ether')
100
```

将这些合约的账户、密码等信息如下：

```
eth.accounts
"0x8ee48a624b79a996db432740e9befa8a387b4a22",
"0xb9e89b2c5cb83fa66b34234a4004129981e70c6f",
"0x5324d341e0e9f1119e13f72d1b26cf3552ea1db2",
"0x63511ed9ce5c7ca63431b8ef5413f7fcd37c4845",
"0xa08962dcdccd844fd0dd8dbdbe8724d1b83c1606",
"0x011117e215f263a9b50d7dbff8df07be35405c52",
"0x5371f730da5b446f499659e1db29a178a9f71a1c",
"0xc8460c3c4d163b11ab10242f71dbe30cf6fa4a41"

their keys:
123456,
123
123,
1234
12345,
123456
1234567,
12345678
```

然后使 `account[0]` 一直挖矿，模拟公链环境：

```
> miner.start()
INFO [12-29|06:39:45.341] Updated mining threads          threads=2
INFO [12-29|06:39:45.341] Transaction pool price threshold updated price=1000000000
null

> INFO [12-29|06:40:26.080] Successfully sealed new block    number=553  sealhash=0a8431...ebb3
0 hash=dc9a1f...433754 elapsed=40.736s
INFO [12-29|06:40:26.080] ⚡block reached canonical chain  number=546  hash=d129cd...a893aa
INFO [12-29|06:40:26.080] ⚡mined potential block          number=553  hash=dc9a1f...433754
INFO [12-29|06:40:26.086] Commit new mining work           number=554  sealhash=200117...e708ed
```

然后开始测试：

### (1) 雇主页面功能:

## 查询自己账户余额

🏠

Salary Easy pay

employer message

your address:

password:

balance: 10000000000000000000  

check balance

contract message

address of employee:

salary per week(wei):

salary pay period(week):

salary stored advanced (should be 3 times more than salary,  
or create operation will be rejected):

contract list

update salary you should pay to selected employees

pay salary to employees you want

delete contract you want to delete

一开始，没有创建任何合约，合约列表为空；我们只需要输入账户 `account[2]` 的地址即可查询其余额（100 以太币）显然 `balance` 正确显示查询结果，该功能测试成功

下面是账户地址输入错误的查询结果（报出错误）：

# employer message

your address:

0x5324d341e0e9f1119e13f72d1b26cf3552ea1d

password:

balance: Error: invalid address

check balance

## 创建与指定用户的工资合约

### Salary Easy pay

#### employer message

your address:

password:

balance: 93999381394000000000

check balance

#### contract message

address of employee:

salary per week(wei):

salary pay period(week):

salary stored advanced (should be 3 times more than salary, or create operation will be rejected):

create contract

#### contract list

your account: 0x5324d341e0e9f1119e13f72d1b26cf3552ea1db2

contract address: 0x04291b847e5a1b9ff2920653bc4faa7c0ab96ea9

employee address: 0xa08962dcdccd844fd0dd8dbdbe8724d1b83c1606

employee's salary: 1700000000000000000(wei) per 2 week

salary you should pay for him now: 0(wei)

0

update salary you should pay to selected employees

pay salary to employees you want

delete contract you want to delete


```
INFO [12-29|08:50:32.268] Submitted contract creation fullhash=0x8dfe01332d2fb88452fb0e35502a798f66e4737833db276362f05ee251b31aa5 contract=0x04291b847E5A1b9ff2920653bc4FaA7c0ab96EA9
INFO [12-29|08:50:33.918] Commit new mining work number=788 seatnash=a5fa5a...f38aa7
uncles=0 txs=1 gas=618606 fees=0.000618606 elapsed=1.083ms
INFO [12-29|08:50:33.930] Successfully sealed new block number=788 sealhash=b00c95...47b541
hash=529f88...ad1363 elapsed=12.060s
INFO [12-29|08:50:33.930] block reached canonical chain number=781 hash=2c7eb0...dfdca8
INFO [12-29|08:50:33.930] mined potential block number=788 hash=529f88...ad1363
```

合约部署成功且与输入信息相同（包括雇员工资、发薪周期、雇主、雇员地址信息），也与区块链私链上挖出确认的合约相同

并且我们看到 account[2]账户上少了 6 个以太币，说明创建合约花费 6 个币（预存工资）

以上说明该功能验证成功

向指定合约付工资（即给签订该合约的用户付工资）

 Salary Easy pay

employer message

your address:

password:

balance: 89999326824000000000

contract message

address of employee:

contract list

your account:  
0x5324d341e0e9f1119e13f72d1b26cf3552ea1db2


contract address:  
0x04291b847e5a1b9ff2920653bc4faa7c0ab96ea9

employee address:  
0xa08962dcdccd844fd0dd8dbdbe8724d1b83c1606

employee's salary:  
17000000000000000000(wei) per 2 week

salary you should pay for him now:

当前时间未付工资之前，应付工资 1.6 个以太币，支付 2 个币后，应付工资为 0 个币：

 Salary Easy pay

employer message

your address:

password:

balance: 87999299539000000000

contract message

address of employee:

contract list

your account:  
0x5324d341e0e9f1119e13f72d1b26cf3552ea1db2

contract address:  
0x04291b847e5a1b9ff2920653bc4faa7c0ab96ea9

employee address:  
0xa08962dcdccd844fd0dd8dbdbe8724d1b83c1606


employee's salary:  
17000000000000000000(wei) per 2 week

salary you should pay for him now:

并且 account[2]账户上少了 2 个以太币，以上说明该功能正确

**更新自己当前时间应向合约交付的工资**（提醒雇主应该交付的工资数额，防止雇主在不知情的情况下违约）

承接上面的截图，我们本来欠薪 1.6 个以太币，付了 2 个，欠薪-0.4 个币（实际不显示负数，显示 0），于是在下一次更新（下一次发薪日）时的应付工资为  $1.7 - 0.4 = 1.3$  个以太币：

 Salary Easy pay

**employer message**

your address:

password:

balance: 87999299539000000000

**contract message**

address of employee:

salary per week(wei):

**contract list**

your account:  
0x5324d341e0e9f1119e13f72d1b26cf3552ea1db2

contract address:  
0x04291b847e5a1b9ff2920653bc4faa7c0ab96ea9

employee address:  
0xa08962dcdd844fd0dd8dbdb8724d1b83c1606

employee's salary:  
17000000000000000000(wei) per 2 week

salary you should pay for him now:  
13000000000000000000(wei)

上面的截图在 2minutes 之后截取，即一个发薪周期之后。

**以上说明该功能正确**

删除指定的工资合约（解除雇主雇员关系）

Salary Easy pay

employer message

your address:

0x5324d341e0e9f1119e13f72d1b26cf3552ea1db2

password:

\*\*\*

balance: 87999299539000000000

check balance

contract message

address of employee:

0xa08962dcdd844fd0dd8dbdbe8724d1b83c1606

contract list

your account:

0x5324d341e0e9f1119e13f72d1b26cf3552ea1db2

☒

contract address:

0x04291b847e5a1b9ff2920653bc4faa7c0ab96ea9

employee address:

0xa08962dcdd844fd0dd8dbdbe8724d1b83c1606

employee's salary:

17000000000000000000(wei) per 2 week

salary you should pay for him now:

13000000000000000000(wei)

20000000000000000000

update salary you should pay to selected employees

pay salary to employees you want

delete contract you want to delete

(删除前)

记录合约信息：（后续如果需要重新添加该已存在合约时，用该合约的地址、雇主账户等信息添加即可。操作方法见下一页的测试）

```
your account:
0x5324d341e0e9f1119e13f72d1b26cf3552ea1db2
contract address:
0x04291b847e5a1b9ff2920653bc4faa7c0ab96ea9
employee address:
0xa08962dcdd844fd0dd8dbdbe8724d1b83c1606
employee's salary:
17000000000000000000(wei) per 2 week
salary you should pay for him now:
13000000000000000000(wei)
```

Salary Easy pay

employer message

your address:

0x5324d341e0e9f1119e13f72d1b26cf3552ea1db2

password:

\*\*\*

balance: 87999299539000000000

check balance

contract list

update salary you should pay to selected employees

pay salary to employees you want

delete contract you want to delete

(删除后)

合约列表中少了对应合约，验证正确

根据指定合约地址添加已经存在的工资合约：

该功能的作用是：

- (1) 误删合约可以通过地址找回
- (2) **浏览器缓存 localStorage 被清空也能通过各个合约地址找回原来的合约、**
- (3) 用户更换设备后新设备 localStorage 为空，则合约列表为空，可以重新通过合约信息找回原来的合约)

测试：

我们删除上面的合约信息已经被记录，这时我们这些信息（合约地址、雇主账户等）添加已经存在的合约：

contract address:

0x04291b847e5a1b9ff2920653bc4faa7c0ab96ea9

add contract exist

我们看到添加的合约还是拥有和原来一样的信息：

contract list

your account:  
0x5324d341e0e9f1119e13f72d1b26cf3552ea1db2  
contract address:  
0x04291b847e5a1b9ff2920653bc4faa7c0ab96ea9  
employee address:  
0xa08962dcdccd844fd0dd8dbdbe8724d1b83c1606  
employee's salary:  
"17000000000000000000"(wei) per "2" week  
salary you should pay for him now:  
0(wei)

0

以上说明该功能正确

## (2) 雇员页面功能：

查询自己账户余额

🏠 Salary Easy pay

employee message

your address:

0xa08962dcdccd844fd0dd8dbdbe8724d1b83c1606

password:

\*\*\*\*\*

balance: 10000000000000000000

check balance

雇员 account[4]查询余额正确，是 100 个以太坊，说明该功能正确



获取指定合约信息（合约余额、工资情况、发薪周期等）

contract message

address of contract:

0x04291b847e5a1b9ff2920653bc4faa7c0ab96ea9

get salary

salary per week(wei):

1700000000000000000

salary pay period(week):

2

deposit in contract:

14000000000000000000

check message

我们使用刚才雇主 account[2]创建的合约地址来 check message，我们发现，查询到的信息与之前一致（薪水 1.7ether/周期，周期为 2）

我们用 check balance 查看 account[2]的余额，发现是 86 个 ether，即它给了合约 14 个币，对应于合约的余额：

🏠 Salary Easy pay

employee message

your address:

0x5324d341e0e9f1119e13f72d1b26cf3552ea1db2

password:

...

balance: 8599927225400000000

check balance

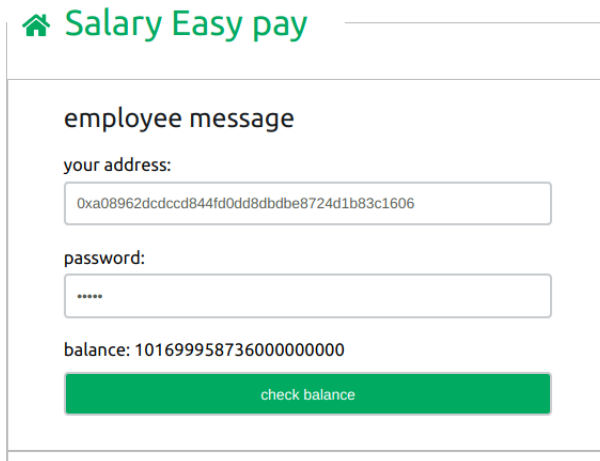
说明查询合约信息正确



### 从指定合约获取工资

我们从上面一步知道了合约的余额, 现在我们可以将这个余额全部取出, 每次 get salary 会获取一份工资:

第一次 get salary, 得到 1.7 个币, 账户余额  $100+1.7$ , 说明 get salary 成功取得一份工资



Salary Easy pay

employee message

your address:

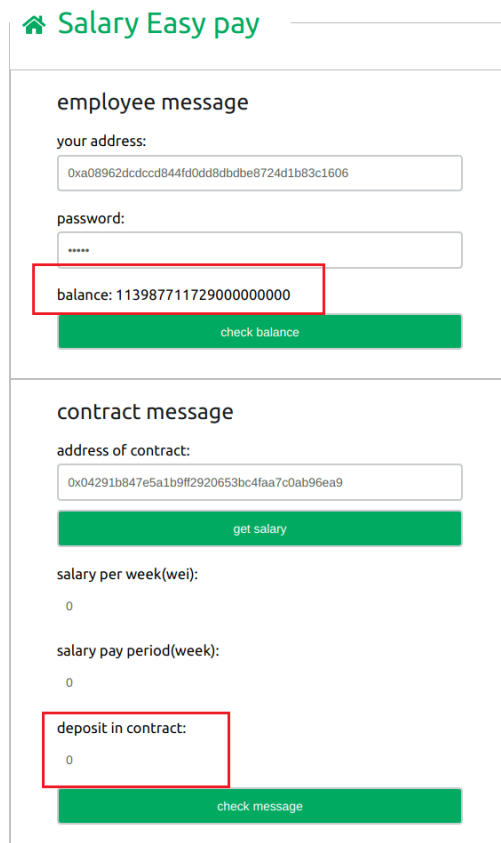
password:

balance: 101699958736000000000

以上说明该操作正确

我们可以不断执行该操作, 取出合约中所有钱, 从下面这张图片我们看到, account[4] 取得了所有的钱, 它的账户余额为 114 以太币, 而合约中的余额从 14ether 变为 0

这也说明 get salary 操作正确



Salary Easy pay

employee message

your address:

password:

balance: 113987711729000000000

contract message

address of contract:

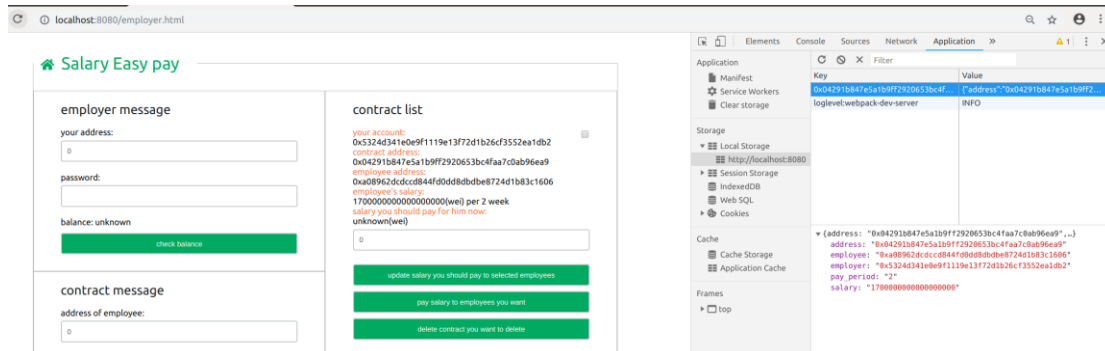
salary per week(wei):

salary pay period(week):

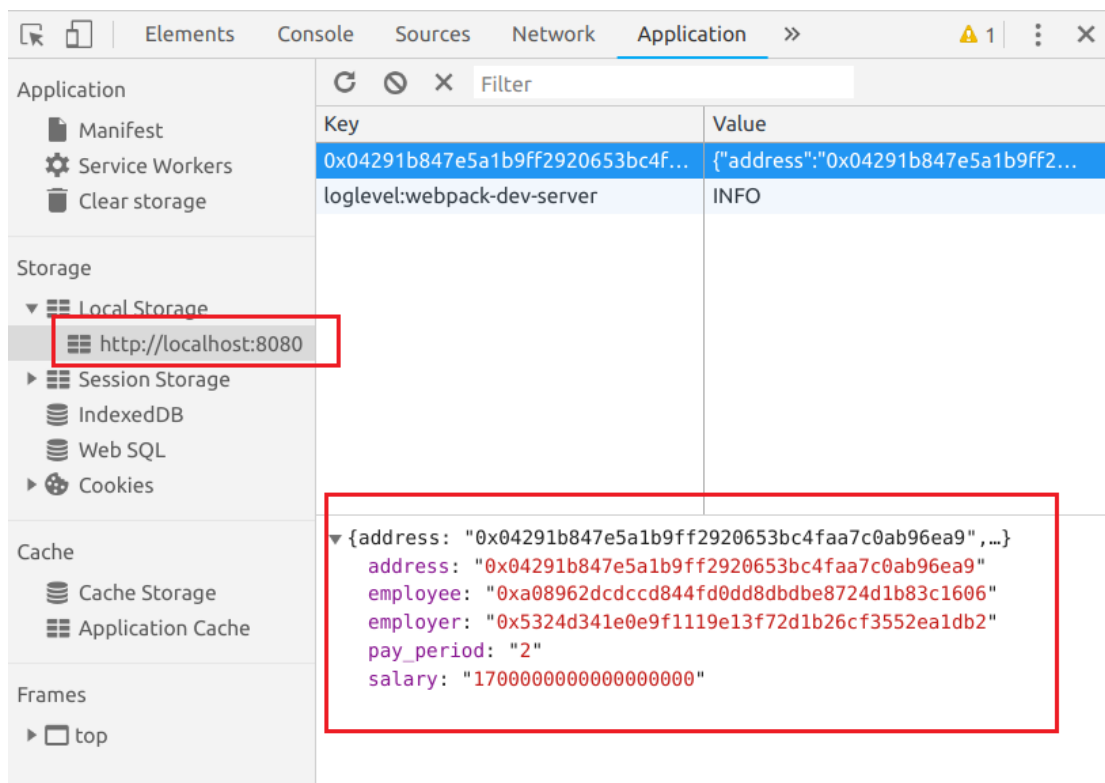
deposit in contract:

### (3) 合约存储功能:

新创建的合约将会以键值对的形式存储在浏览器的 localStorage 中, 永久保存:



我们截取详细内容:

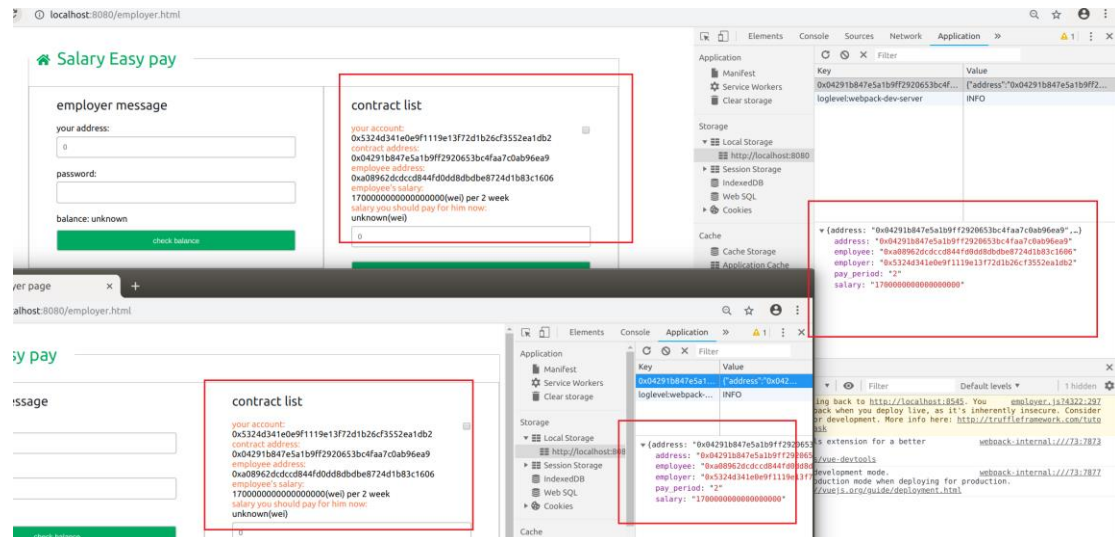


这个内容和合约列表中的合约信息一致:

your account:  
0x5324d341e0e9f1119e13f72d1b26cf3552ea1db2  
contract address:  
0x04291b847e5a1b9ff2920653bc4faa7c0ab96ea9  
employee address:  
0xa08962dcdcc844fd0dd8dbdbe8724d1b83c1606  
employee's salary:  
17000000000000000(wei) per 2 week  
salary you should pay for him now:  
unknown(wei)

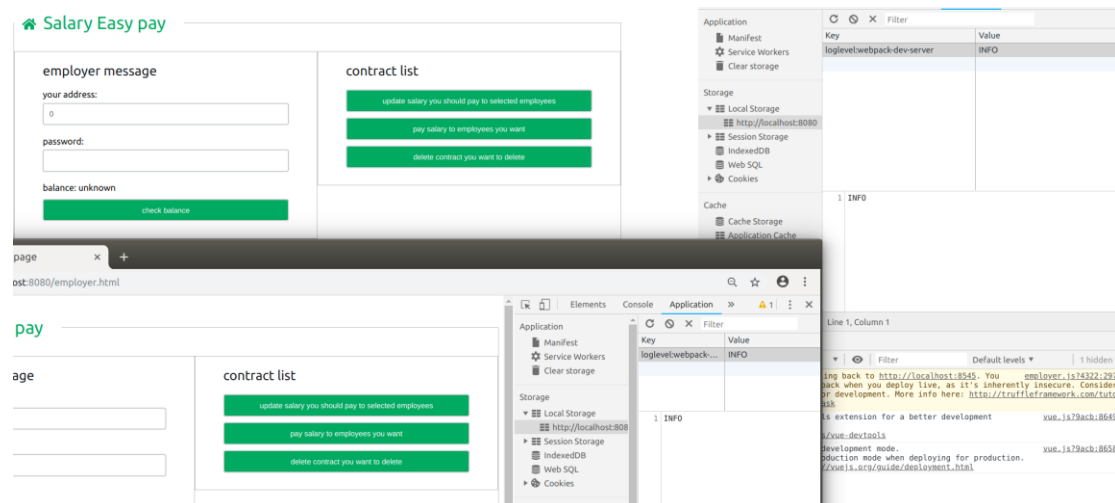
说明存储信息正确

可以看到在浏览器的 localStorage 中存储了之前创建的合约信息，我们关闭浏览器/webpack 打开的 localhost:8080，然后重启浏览器/webpack 打开的 8080 端口服务，仍然能够看到已经存储的合约信息被重新加载：



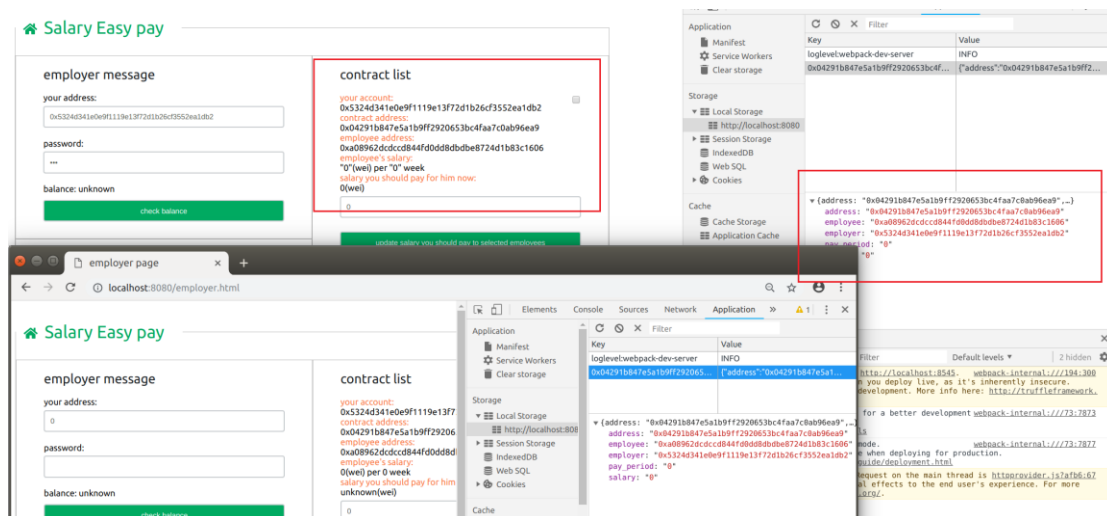
(两个页面都能看到合约，说明合约加载成功)

删除合约 (localStorage 数据也被删除)，合约列表将没有该合约：



(两个页面都没有该合约了，localStorage 也没有该条目了，)  
即正确从 localStorage 删除合约

重新通过 add contract exist 功能还可以重新添加回来：



我们看到合约被添加回来，信息与之前被创建的合约一致  
说明正确添加到 localStorage

以上是全部测试内容，时间有限，更多的详细信息无法一一展示

本程序还提供各种各样的报错信息（提示用户正确操作）、更多的列表操作（通过选中想要操作的列表元素，执行相对应的操作）

详细情况推荐下载项目资源进行考察。后续如果有时间也可能会在 github 项目的 readme 中添加更多的测试截图（代码部分已经确定，不会再改了）

感谢阅读！