测试

运行环境

- Node Package Manager (NPM)
- Truffle Framework （NPM 的一个工具包）
- Ganache (windows 应用，或其他测试链)
- Metamask (Chrome 插件，用于账户登陆)

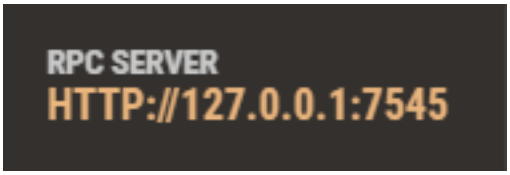
合约部署

- 执行 `npm install` 安装 npm 依赖包：

```
C:\Users\user\Desktop\blockchain  (peppa-insurance@1.0.0)
λ npm install
npm WARN peppa-insurance@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4:
wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

added 68 packages in 10.118s
```

- 打开 Ganache，将项目加入 Ganache 项目中：

同时获得测试连地址：



- 将./truffle-config.js 与./src/js/app.js 中的区块链地址更换 localhost:7545

```
development: {
 host: "127.0.0.1",     // Localhost (default: none)
 port: 7545,            // Standard Ethereum port (default: none)
 network_id: "*",       // Any network (default: none)
},
```

```
App =
{
    web3Provider: null,
    contracts: {},
    account: '0x0',
    instance: null,
    chainAddress: "http://localhost:7545",
    info: {},
```

- 修改./migrations/2_deploy_contracts.js 中的参数，将一个可用的账户地址作为认证机构的地址填入第四个参数：

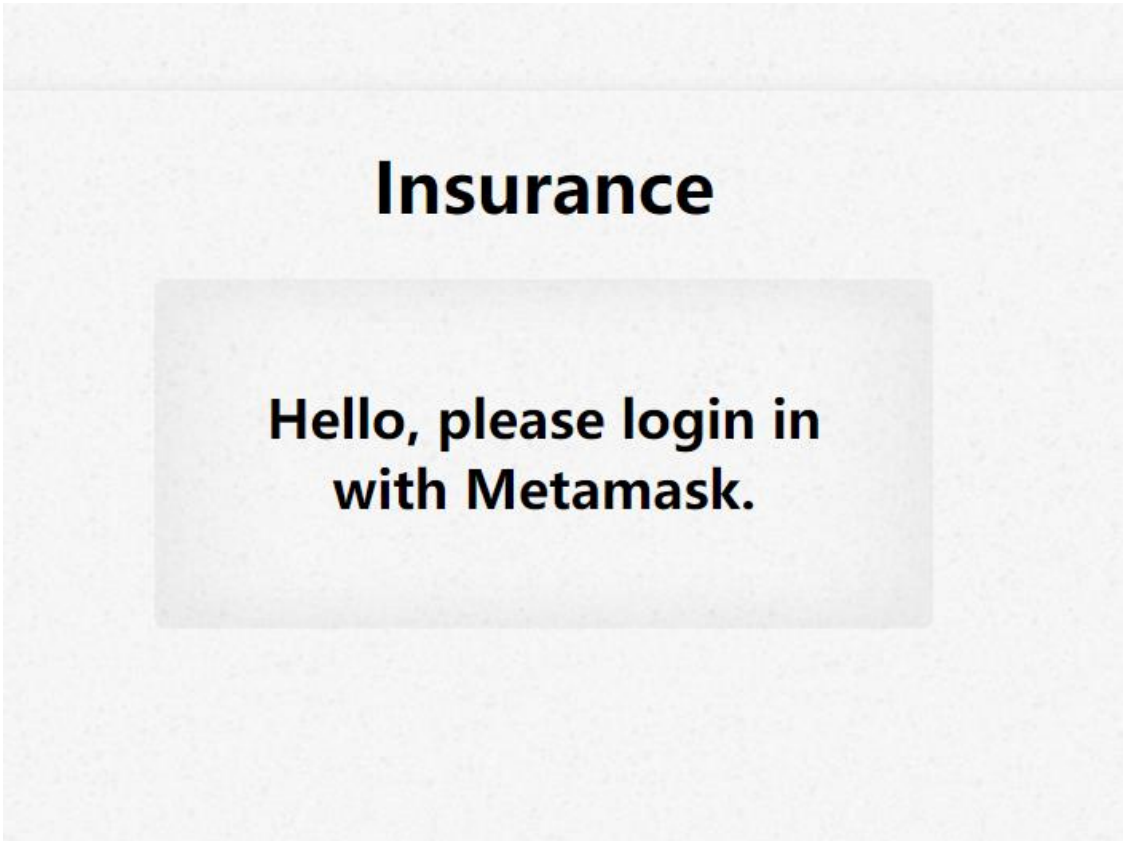
```
module.exports = function(deployer)
{
    //argument:
    //arg1: price(in ehter)
    //arg2: compensation(in ehter)
    //arg3: valid period(in day)
    //arg4: address of authority agency
    deployer.deploy(insuranceContract, 1, 2, 1, "0xA1Dd1C57DEAE9f50BbaCC020213097DCAC3810b2");
};
```

- 使用 `truffle migrate --reset` 部署合约, 在 Ganache 中可以看到合约已经部署:

- 然后使用 `npm run dev` 开启服务器:

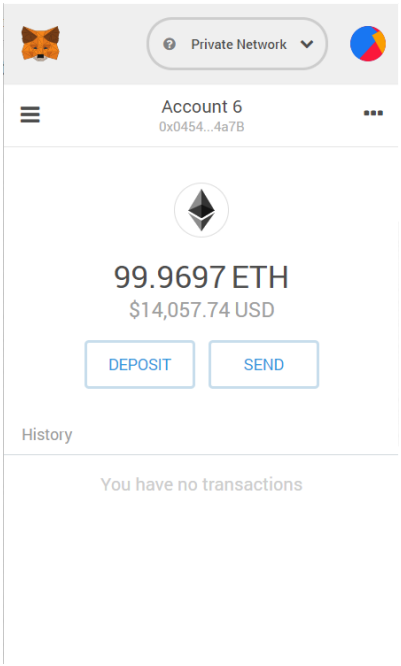
```
C:\Users\user\Desktop\blockchain  (peppa-insurance@1.0.0)
λ npm run dev

> peppa-insurance@1.0.0 dev C:\Users\user\Desktop\blockchain
> lite-server

** browser-sync config **
{ injectChanges: false,
  files: [ './**/*.{html,htm,css,js}' ],
  watchOptions: { ignored: 'node_modules' },
  server:
   { baseDir: [ './src', './build/contracts' ],
     middleware: [ [Function], [Function] ] } }
[Browsersync] Access URLs:
 ------------------------
 Local: http://localhost:3000
 ------------------------
    UI: http://localhost:3001
 ------------------------
[Browsersync] Serving files from: ./src
[Browsersync] Serving files from: ./build/contracts
[Browsersync] Watching files...
18.12.29 21:50:02 304 GET /index.html
18.12.29 21:50:02 304 GET /css/style.css
18.12.29 21:50:02 304 GET /js/jquery-3.2.1.min.js
18.12.29 21:50:02 304 GET /js/bootstrap.min.js
18.12.29 21:50:02 304 GET /js/web3.min.js
18.12.29 21:50:02 304 GET /js/truffle-contract.js
18.12.29 21:50:02 200 GET /js/app.js
18.12.29 21:50:02 304 GET /images/bg.jpg
18.12.29 21:50:02 304 GET /images/bgshadow.png
18.12.29 21:50:02 200 GET /insurance.json
```

- 在浏览器中输入 localhost:3000 访问交互式界面:

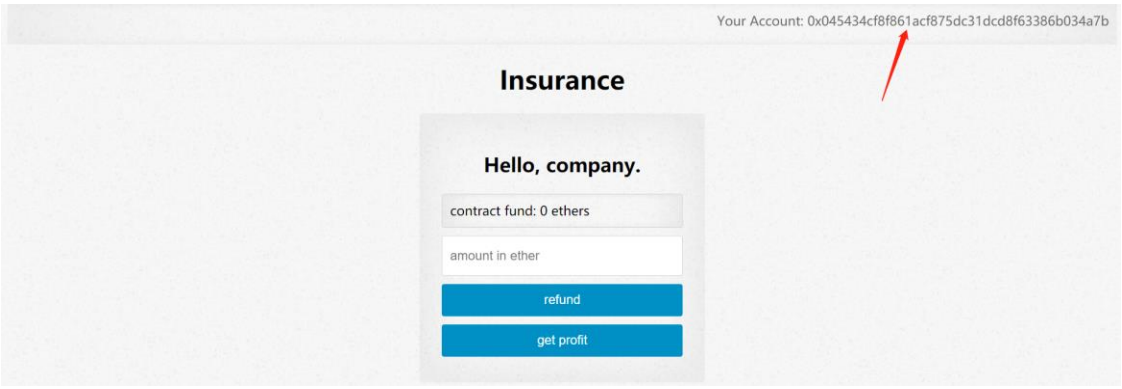


合约使用

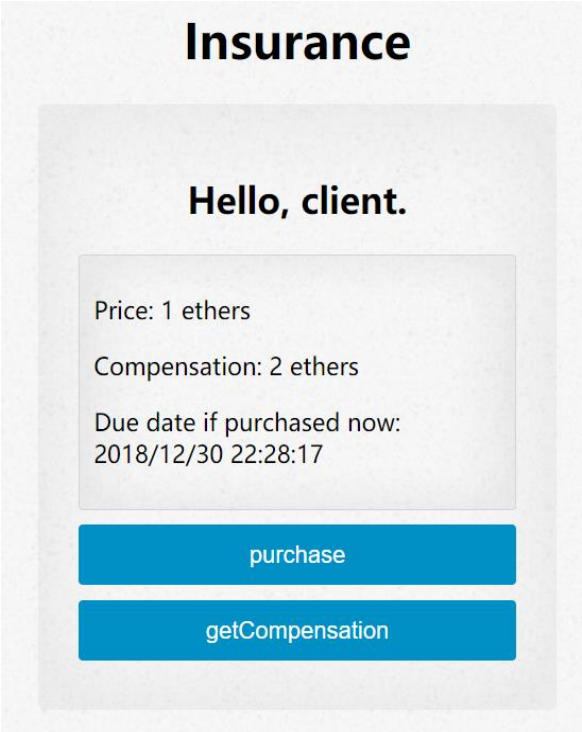
- 使用 Chome 上的插件 Metamask 进行登陆：将 Metamask 链接到测试链上，并将在 Ganache 上获取私钥并导入到 Metamask 即可登陆：



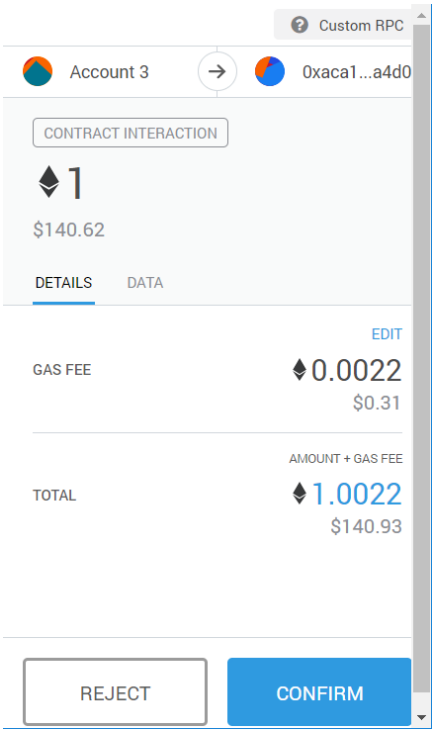
登陆后刷新页面，右上角显示登陆的账户地址，不同角色登陆页面会显示不同内容。测试链第一个账户，即部署合约时所用的用户为公司账户；部署合约时填入参数中的账户为认证机构账户；其余用户为客户。现登陆的是公司账户（合约里一分钱也没有）：



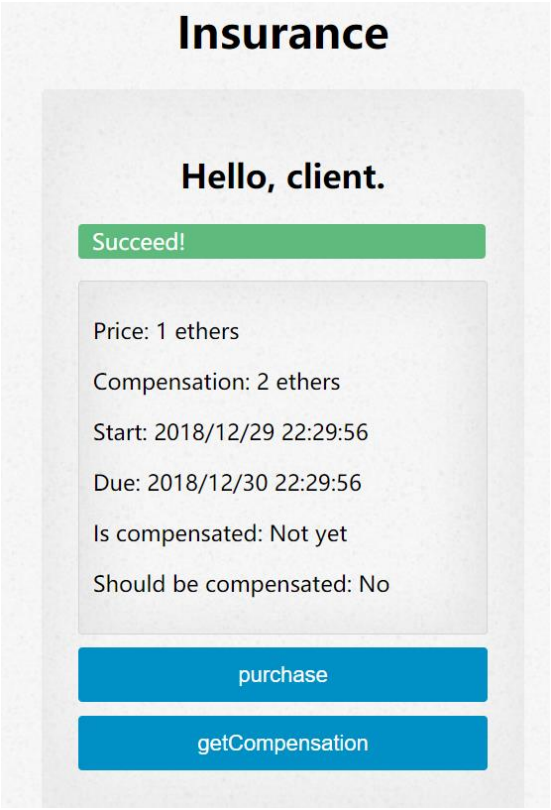
- 现在先切换到一个客户账户，可以看到合约的标价、赔付金额，预测到期时间。



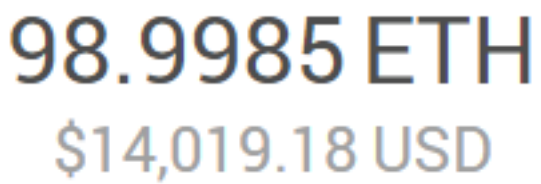
点击 purchase 按钮进行购买，这时会弹出一个 metamask 窗口让我们确认交易（后续的这个弹框将不再截图）：



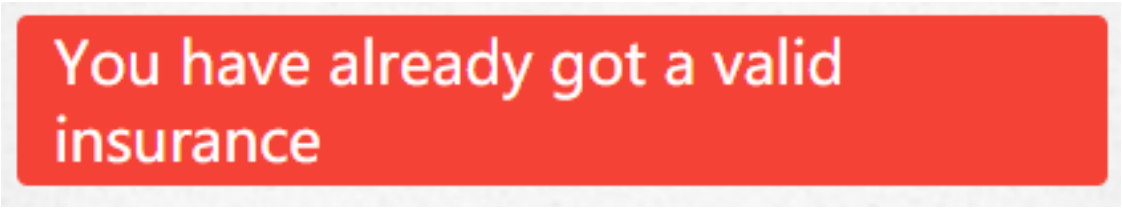
购买成功后弹出成功提示框，并显示详细的保险内容，包括保险的购买时间、到期时间、是否获得赔付、是否有资格获得赔付等信息：



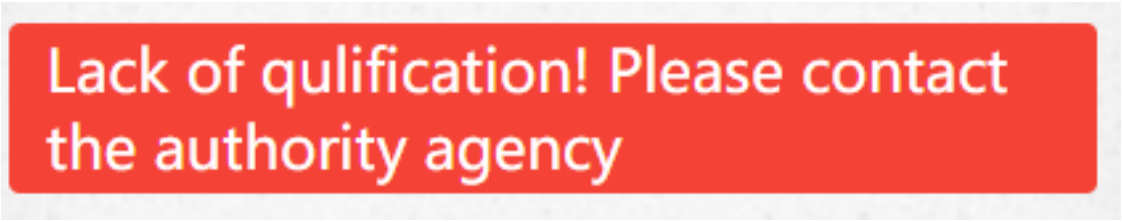
付款后账户里的钱的确被扣掉了，扣了 1 ether+手续费:



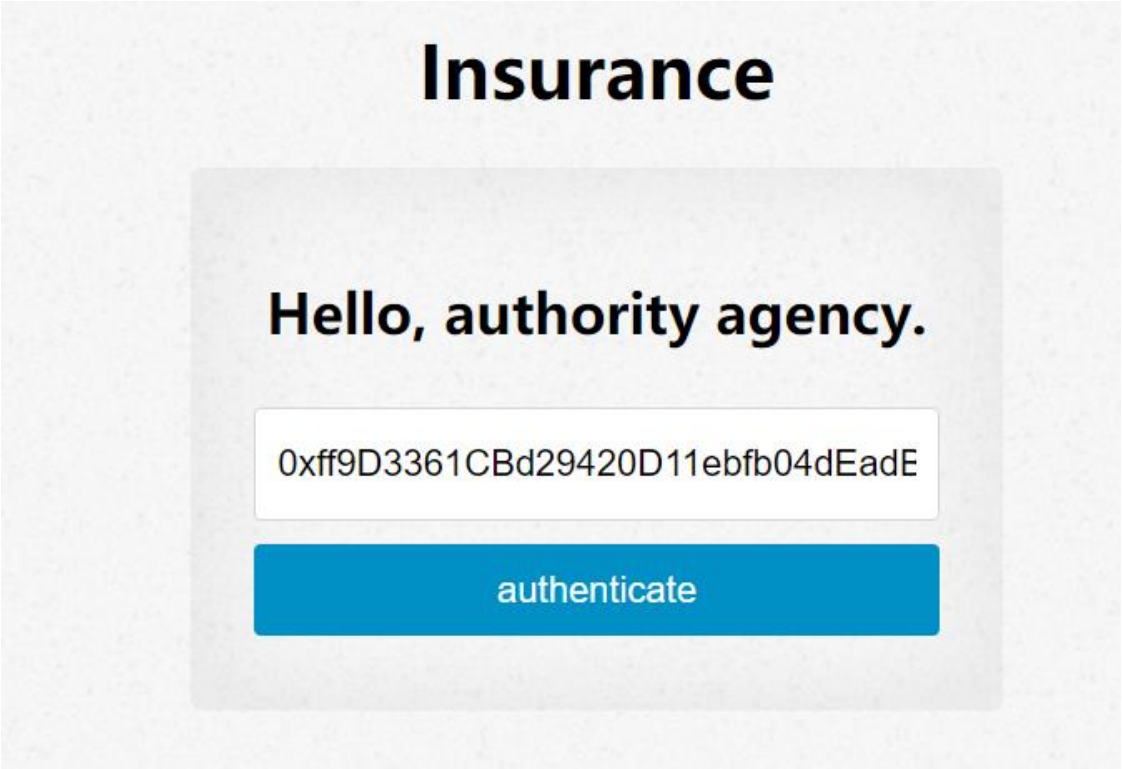
再点击购买按钮会出现该弹框，表明用户已有一个合法的保险了，不能重复购买，合法的是指未过期、未赔付的保险：



这时点击 getCompensation 按钮，会弹出无资格获赔提示框：

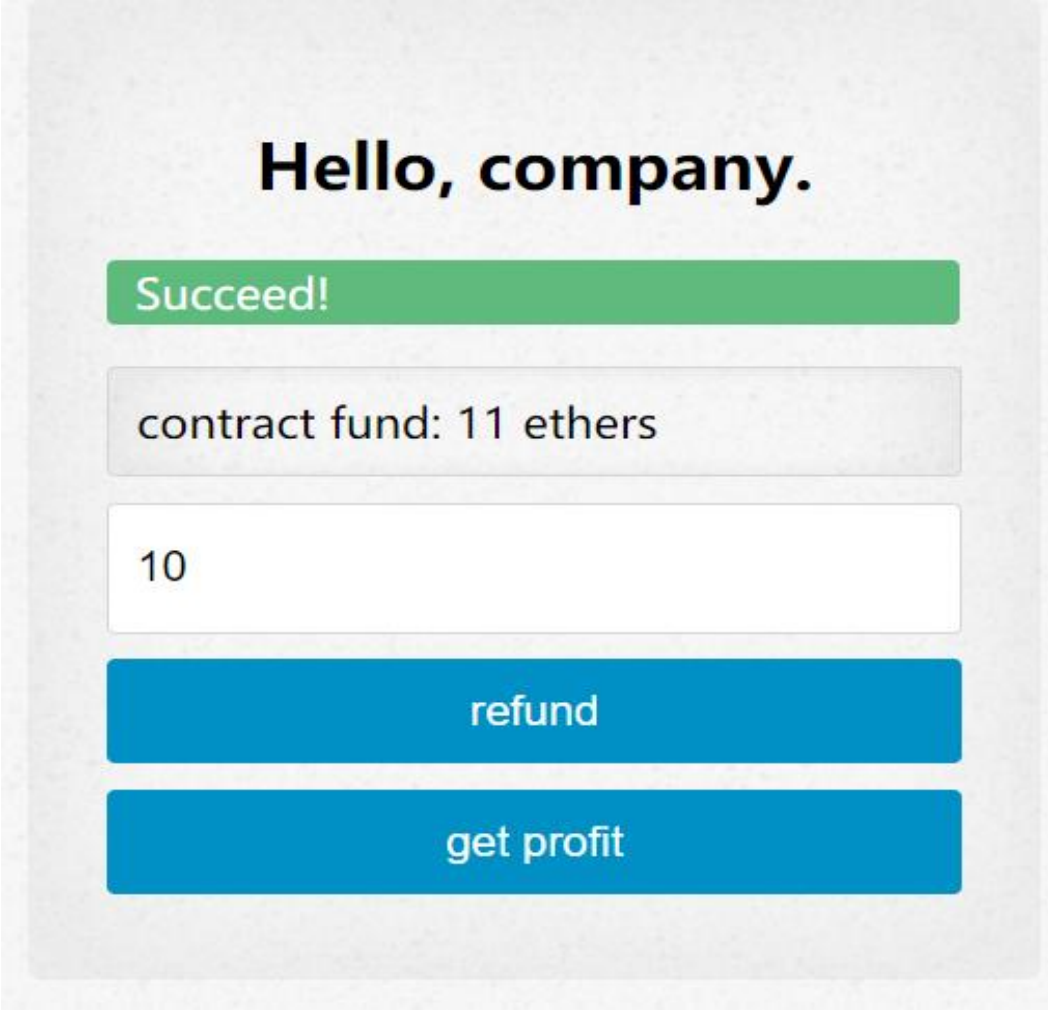


- 下一步我们假设上述用户确实满足赔付条件，可以获得赔偿。我们切换到认证机构账户,通过了上述用户的获赔申请，现给予该用户获得赔付的权利：

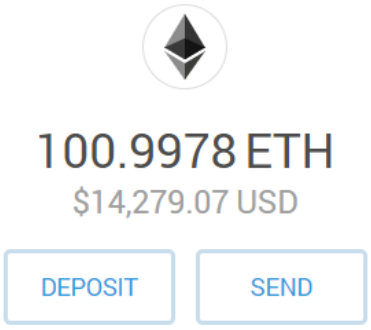


- 成功后客户前去获赔，发现合约的钱并不够，因为合约里只有客户自己付的一个以太币：

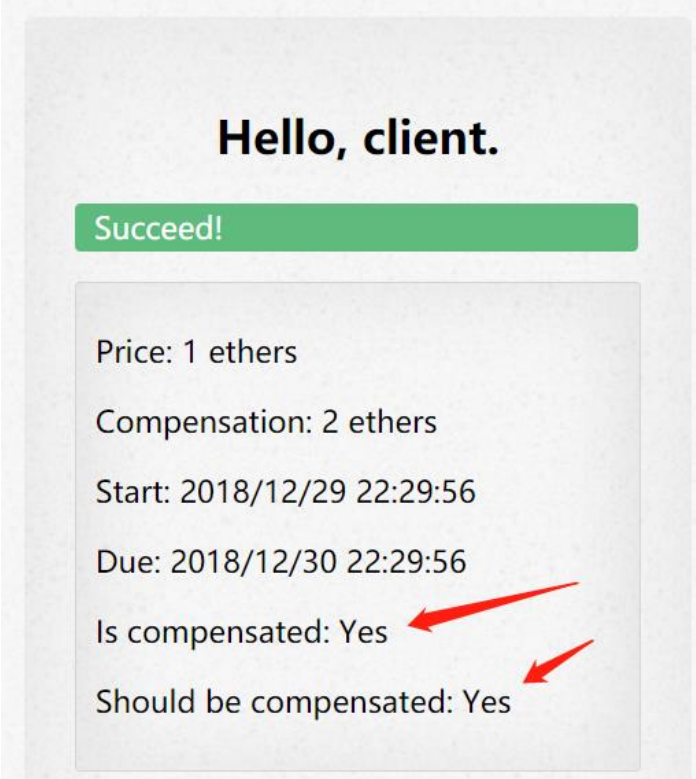
- 这时切换到公司账户，为合约注入资金（refund），一口气将 10 个以太币汇入合约中：



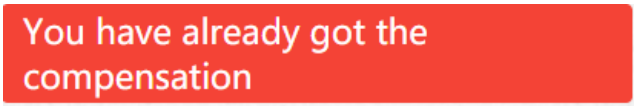
- 这时用户就能获得赔款了，可以看到账户上的钱是比初始状态（100 个币）多将近 1 个以太币的，当然这个过程当中被扣掉了一些手续费。



合约状态也发生了改变：



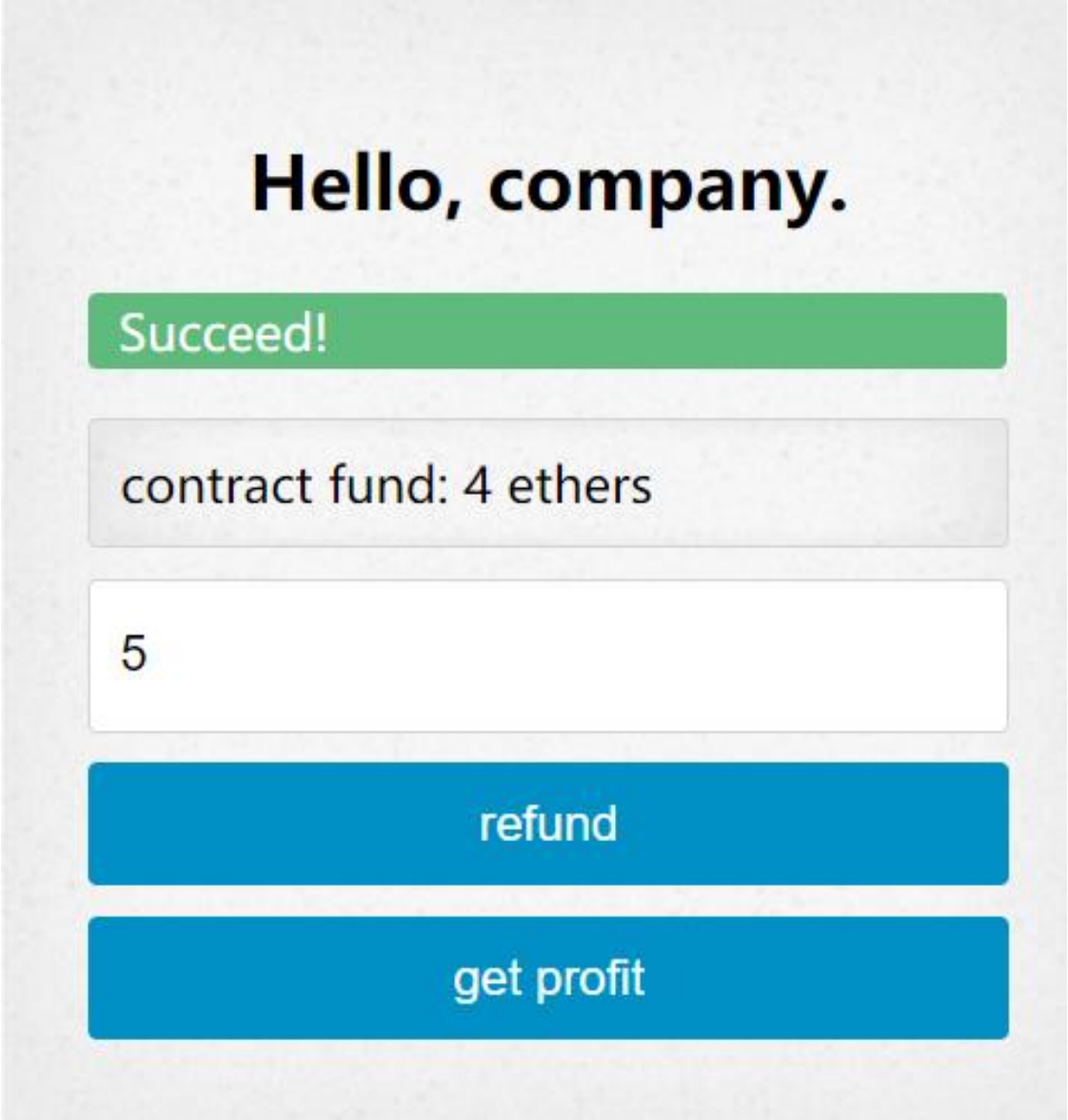
这时再点击获赔按钮会获得如下提示：



但因为上一份保险已被赔付，这时可以重新购买保险。

- 假设很多人买了保险，保险公司觉得合约里的钱已经足够支付一定数额的赔款了，此时保险公司可以从合约中拿钱出来，进行盈利：

  合约原有 9 个以太币，现公司取走 5 个以太币，合约中还剩下 4 个以太币，而公司账户上也多出了 5 个以太币。



- 以上就是合约调用的正常流程。

控制台攻击防御

- 现假设有一个恶意用户 dinosaur 想绕开前端 js 检查，通过控制台调用合约函数进行攻击：

- dinosaur 强行调用获赔函数想获得赔款，但被智能合约拒绝，偷鸡不成还蚀把手续费：

```
> App.instance.getCompensation().then(function(message)
  {
      console.log("I made it")
  }).catch(function(error)
  {
      console.warn(error)
  })
◂ ▶ Promise {<pending>}
✕ ▶ MetaMask - RPC Error: Error: Error: [ethjs-rpc]   inpage.js:1
  rpc error with payload
  {"id":976424524107,"jsonrpc":"2.0","params":
  ["0xf86b808504a817c800836170d494aca10ecf72c20ede9c1bb465aac82cd
  e70faa4d08084076aa026822d46a07bb31ba6146042b947b0025ec391000573
  59c90834b76924491715e1123f6273a04228c885cf6d7d84eb6475a3ff65cbd
  8246d3ca2a448e7d8a1f305437a80bb04"],"method":"eth_sendRawTransa
  ction"} Error: VM Exception while processing transaction:
  revert You can not get the compensation
    {code: -32603, message: "Error: Error: [ethjs-rpc] rpc error w
  ▶ ith payload {…nsaction: revert You can not get the compensatio
    n"}
```

- dinosaur 假冒认证机构给予自己的账户获赔权力，但被告知账户不符，没有认证机构账户的私钥就无法调用该函数：

```
> App.instance.authenticateCompensation(App.account,
{from:"0xA1Dd1C57DEAE9f50BbaCC020213097DCAC3810b2",
value:0}).then(function(message)
{
    console.log("I made it")
}).catch(function(error)
{
    console.warn(error)
})
< ▶ Promise {<pending>}
✕ ▶ MetaMask - RPC Error: Error: WalletMiddleware -    inpage.js:1
Invalid "from" address.
    at l (chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn/b
ackground.js:1:1071487)
  {code: -32603, message: "Error: WalletMiddleware - Invalid "fr
▶ om" address. ↵…ogaeaoehlefnkodbefgpgknn/background.js:1:107148
  7)"}
```

- 假冒其他账户、绕过 js 检测非法调用合约函数都会被 metamask 或智能合约拒绝，例子与上述两个大同小异，在之前的“合约部署报告”中就写得比较详细了，在此就不一一列举了。