Solidity Contracts

How Solidity Works

Get A Receipt

```
curl  \
-H "Content-Type: application/json" \
-X POST \
--data '{"jsonrpc":"2.0", "method":"eth_getTransactionReceipt","params":\
   ["0x3f3aa792dd4a76d6feea51d57fc6543e97031cb4fb53e76642243eab0dfdb343"],"id":1}' \
http://192.168.0.199:8545/
```

Get an Account Blaance

```
curl  \
-H "Content-Type: application/json" \
-X POST \
--data '{"jsonrpc":"2.0", "method":"eth_getBalance", "params":[\
    "0x6ffba2d0f4c8fd7961f516af43c55fe2d56f6044", "latest"], "id":1}' \
http://192.168.0.199:8545
```

Unlock an Account

```
#!/bin/bash

# UTC--2018-03-11T01-56-42.511489695Z--9a6446d642d76a3ac1baf6c6d8c1e5179c58d87f

geth attach "http://192.154.97.75:8545/" <<XXxx
personal.unlockAccount( "0x9a6446d642d76a3ac1baf6c6d8c1e5179c58d87f", "xtwHdVIsKNFdMOcI
exit
XXxx
```

Storage & Events

A simple PayFor contract that will accept payments and create events.

```
pragma solidity >=0.4.21 <0.6.0;

```

```
import "openzeppelin-solidity/contracts/ownership/Ownable.sol";

contract PayFor is Ownable {

  address payable owner_address;
  event ReceivedFunds(address sender, uint256 value, uint256 application, uint256 l
  event Withdrawn(address to, uint256 amount);

  uint256 internal nPayments;
  uint256 internal paymentID;

  address[] private listOfPayedBy;
  uint256[] private listOfPayments;
  uint256[] private payFor;

  mapping (address => uint256) internal totalByAccount;

  constructor() public {
    owner_address = msg.sender;
    nPayments = 0;
  }

  function ReceiveFunds(uint256 forProduct) public payable returns(bool) {
    nPayments++;
    uint256 pos;
    pos = listOfPayments.length;
    listOfPayedBy.push(msg.sender);
    listOfPayments.push(msg.value);
    payFor.push(forProduct);
    uint256 tot;
    tot = totalByAccount[msg.sender];
    totalByAccount[msg.sender] = tot + msg.value;
    emit ReceivedFunds(msg.sender, msg.value, forProduct, pos);
    return true;
  }

  function getNPayments() public onlyOwner payable returns(uint256) {
    return ( nPayments );
  }

  function getPaymentInfo(uint256 n) public onlyOwner payable returns(address, uint
    return ( listOfPayedBy[n], listOfPayments[n], payFor[n] );
  }

  function withdraw( uint256 amount ) public onlyOwner returns(bool) {
    require(amount <= address(this).balance, "Insufficient funds for witdrawl");
    address(owner_address).transfer(amount);
    emit Withdrawn(owner_address, amount);
    return true;
  }

  function getBalanceContract() public view onlyOwner returns(uint256){
    return address(this).balance;
  }
}
```

Testing for Contract

Dennis Ritchie: “Software that is not tested is broken.”

Overview

1. How do you know that your results are correct?
2. What will testing tell you
3. What will it not tell you
4. Code Review
5. Testing and the real world
6. Different ways of testing
    1. Think Testing
    2. Unit Testing
    3. TDD - Test Driven Development
    4. Integration Testing
    5. Formally Proven

Solidity

```
pragma solidity >=0.4.21 <0.6.0;

import "truffle/Assert.sol";
import "truffle/DeployedAddresses.sol";
import "../contracts/PayFor.sol";

contract PayFor_Test {

    PayFor token = new PayFor();

    function testGetNPayments() public {
        uint tmp = token.getNPayments();
        Assert.equal(0, tmp, "No Payments Yet.");
    }
    function testGetBalanceContract() public {
        uint tmp = token.getBalanceContract();
```

```
            Assert.equal(0, tmp, "Random address has 0 tokens.");
        }

    }
```

Mocha / Web3 / JS

```

const PayFor = artifacts.require('./PayFor.sol');

contract('PayFor', function(accounts) {

  let payForItem,
    account_two = accounts[1];

  beforeEach(async () => {
    payForItem = await PayFor.new();
  });

  it("should pay for an item", async function() {
    // event ReceivedFunds(address sender, uint256 value, uint256 application, uint
    // emit ReceivedFunds(msg.sender, msg.value, forProduct, pos);
    const ReceivedFunds = payForItem.ReceivedFunds();

    await payForItem.ReceiveFunds( 42, {from: account_two, value:1000});
      ReceivedFunds.get(function(error, result){
      assert.equal(result[0].event, 'ReceivedFunds', "ReceivedFunds event should oc
    });

    var nPayments = await payForItem.getNPayments();
    assert.equal(nPayments, 1, "Shoudl be 1 Payment Made.");

    let [addr_to, amount, for] = await payForItem.getPaymentInfo(0);
    assert.equal(amount, 1000, "Shoudl be 1000 Payed Made.");

  });

});
```

Truffle Test

```
$ truffle test
```

Test Output

```
Using network 'development'.

Compiling ./contracts/PayFor.sol...
Compiling ./test/PayFor_Test.sol...
Compiling ./test/helpers/ThrowProxy.sol...
Compiling openzeppelin-solidity/contracts/ownership/Ownable.sol...
Compiling truffle/Assert.sol...
Compiling truffle/DeployedAddresses.sol...


  PayFor_Test
    ✓ testGetNPayments (62ms)
    ✓ testGetBalanceContract (54ms)

  Contract: PayFor
    ✓ should pay for an item (86ms)


  3 passing (4s)
```