Web3 JS API:

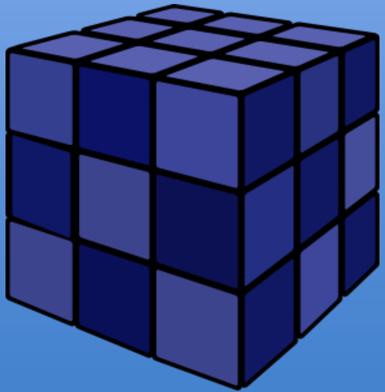
- Connecting from DAPP to Node
- Download setup Workbench DAPP (sample)
- Workbench implementation

This deck is part of a online course on "Ethereum: Design and Development of Decentralized Apps.

raj@acloudfan.com

@acloudfan

http://ACloudFan.com



Dapp Libraries

https://docs.web3j.io/getting_started.html

https://github.com/pipermerriam/web3.py

- Multiple libraries available for connecting to Ethereum

web3 js

web3J

nethereum

Web3.py

python™

Big Numbers

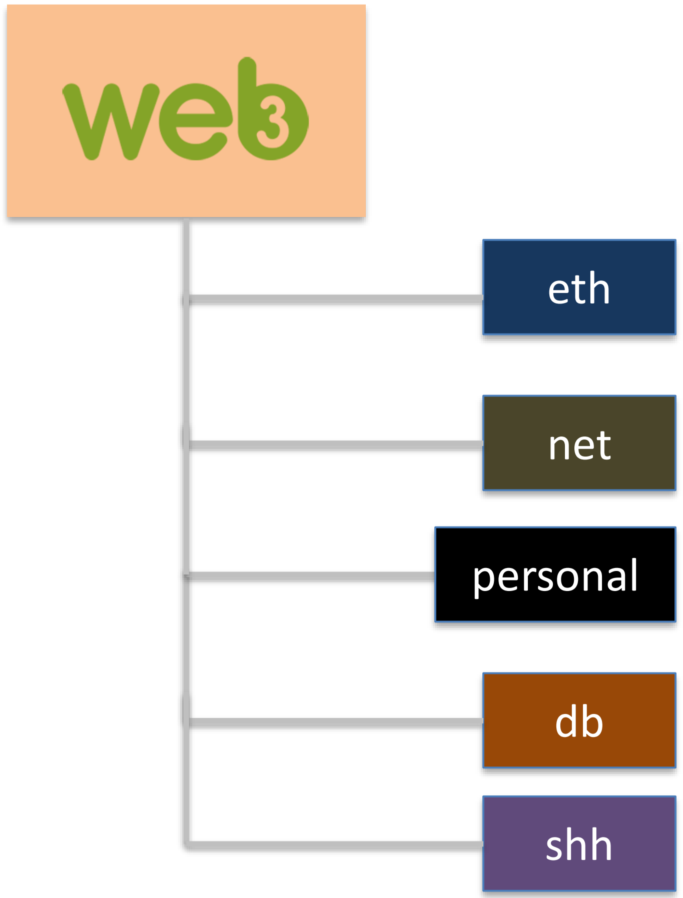
- Javascript cannot handle big number values correctly
- web3JS uses the *BigNumber* library
  - Even BigNumber cannot handle more than 20 floating points

Solution: Manage the balances in WEI

https://github.com/MikeMcl/bignumber.js/

Web3 API Overview

https://github.com/ethereum/wiki/wiki/JavaScript-API



Ethereum blockchain related methods

Node's network status

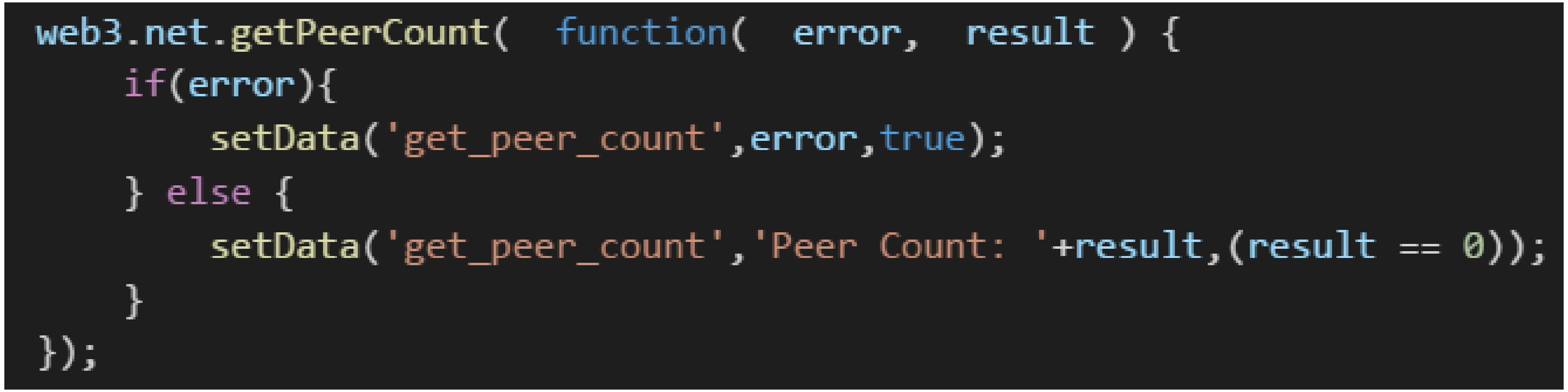
Account functions and sending

Get/put for local *LevelDB*

P2P messaging using *Whisper*

Web3 JS Asynchronous Calls

- A number of API have Synchronous & Asynchronous flavor
- Asynchronous: *Error-First Callback*



```
web3.net.getPeerCount(  function(  error,  result ) {
    if(error){
        setData('get_peer_count',error,true);
    } else {
        setData('get_peer_count','Peer Count: '+result,(result == 0));
    }
});
```

Install NodeJS Tools/Components

1 Install Yeoman

YEOMAN

```
> npm install –g yo
```

Install Yeoman webapp template

```
> npm install –g   generator-webapp
```

3 Install Gulp

Gulp

```
> npm install –g gulp
```

4 Install Bower

Bower

```
> npm install –g bower
```

Setup Dapp

1 Create a folder for application

2 Create the application

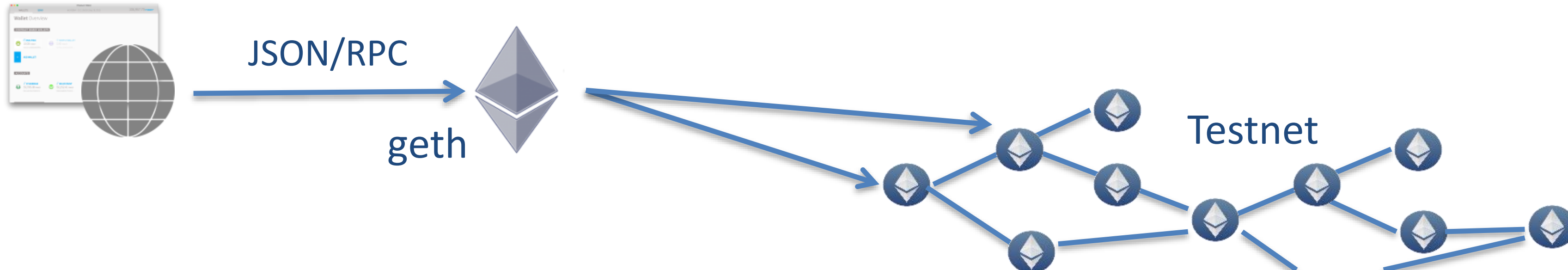
```
> yo  webapp
```

3 Install web3 library

```
> bower install web3  --save
```

Web3 Sample App: Available @ http://TheDapps.com

- Single page application (HTML, Javascript)
  - Not using any Javascript framework + Minimal error handling
  - Minimilistic UI as Focus is on use of web3JS API
  - Developed against Ehereum client : *geth*



Web3 Workbench

- Decentralized application
  - Will show the use of web3 API

http://The**D**apps.com

Workbench DAPP will work with:

- Local (or Remote) Ethereum node e.g., geth
- TestRPC
- MetaMask

Download & Setup Sample Application Locally

1. Download the application
   - http://acloudfan.com/download-files
2. Unpack the zip file in a directory
3. Run > npm install
4. Run > gulp serve

On Load Processing

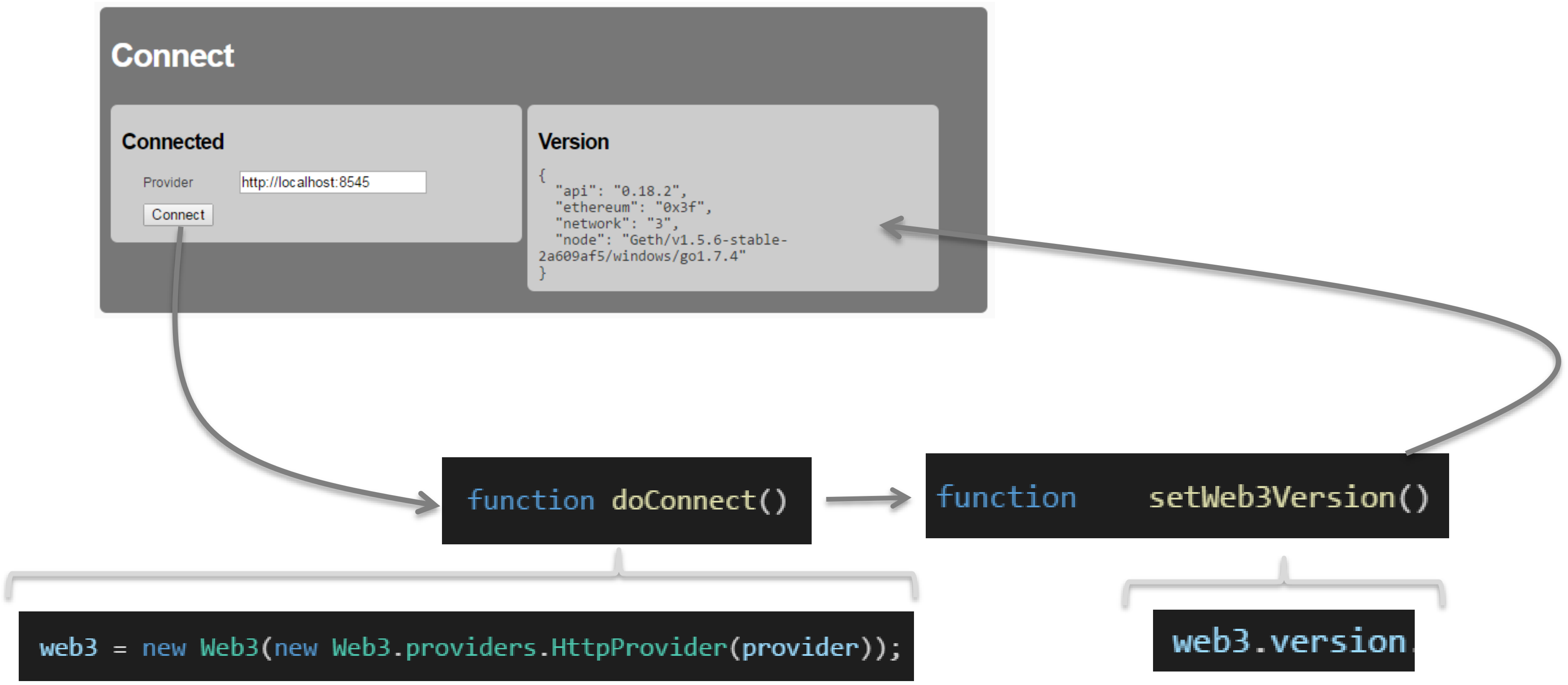
1. Checks if MetaMask has injected the web3 object

2. If web3 is not found then app tries to connect with local node

App Structure

index.html	• HTML UI Components
main.js	• All web3 JS API calls + some UI related code
utils.js	• UI related utility functions

Connect & Version



Connect

Connected

Provider http://localhost:8545

Connect

Version

```
{
  "api": "0.18.2",
  "ethereum": "0x3f",
  "network": "3",
  "node": "Geth/v1.5.6-stable-
2a609af5/windows/go1.7.4"
}
```

```
function doConnect()
```

```
function    setWeb3Version()
```

```
web3 = new Web3(new Web3.providers.HttpProvider(provider));
```

```
web3.version
```

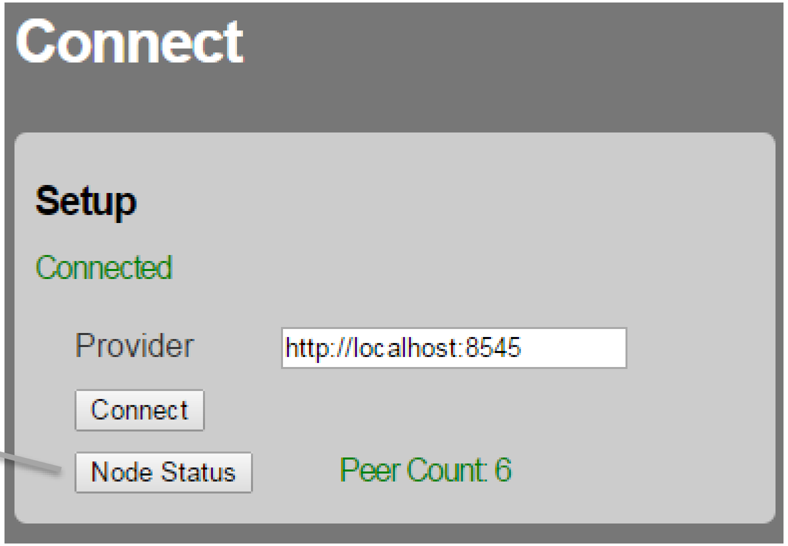
web3.net

- web3.net

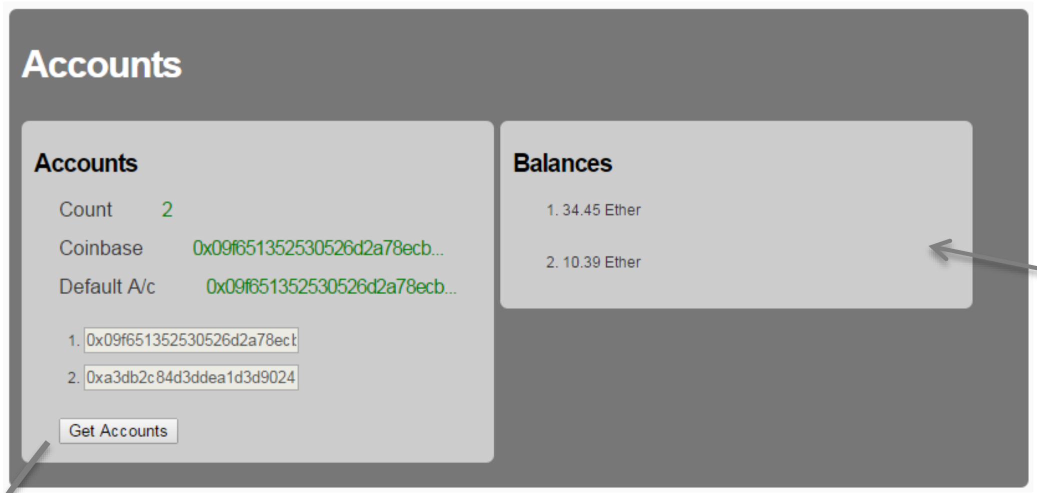
listening / getListening

peerCount/ getPeerCount

```
function    doGetNodeStatus()
```



Get Accounts & Balances



Accounts

Accounts

Count 2

Coinbase 0x09f651352530526d2a78ecb...

Default A/c 0x09f651352530526d2a78ecb...

1. 0x09f651352530526d2a78ecb
2. 0xa3db2c84d3ddea1d3d9024

Balances

1. 34.45 Ether
2. 10.39 Ether

```
function    doGetAccounts()
```

```
function    doGetBalances(accounts)
```

```
web3.eth.getAccounts(function (error, result)
```

```
web3.eth.defaultAccount
```

```
web3.eth.coinbase
```

```
web3.eth.getBalance(accounts[i]);
```

web3.eth.coinbase

- Account for mining rewards
- Read only
- Cannot be set using web3 eth object

```
web3.miner.setEtherbase(web3.eth.accounts[1])

> geth   --address   coinbase_address
```

web3.eth.defaultAccount

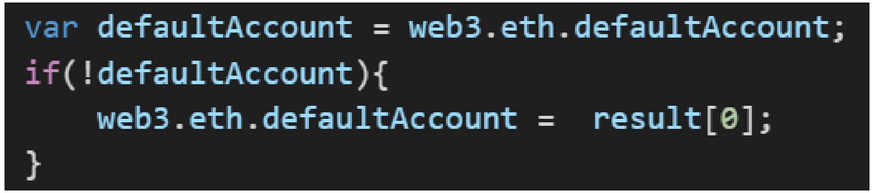
- Read/Write

- Used in these methods if *<u>from:</u>* not specified

  web3.eth.sendTransaction()

  web3.eth.call()

- May be undefined *(depending on implementation)*



```
var defaultAccount = web3.eth.defaultAccount;
if(!defaultAccount){
    web3.eth.defaultAccount =  result[0];
}
```

web3.eth.getBalance

- Gets the balance for the account

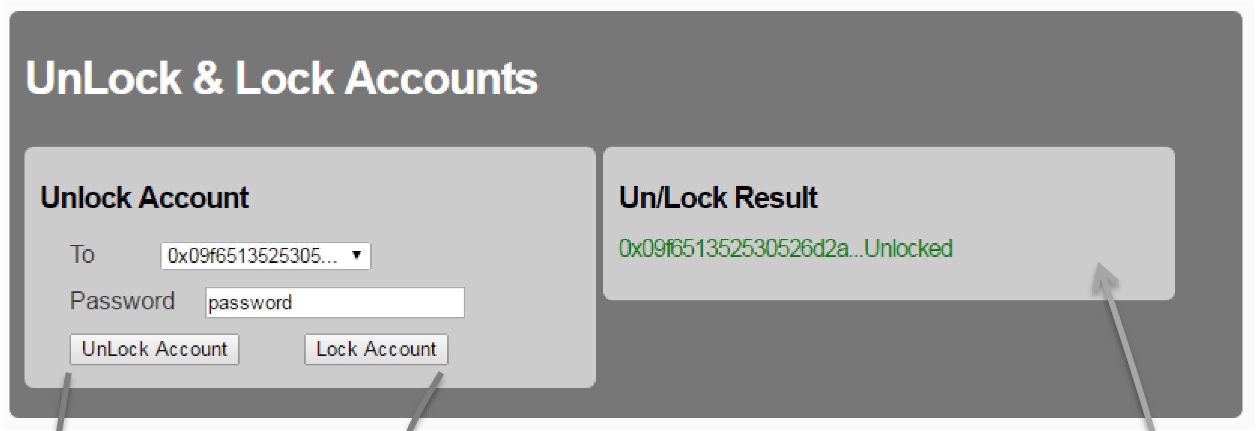
web3.eth.getBalance (address)

Result: Balance in ***wei***

var balance_in_ethers = web3.fromWei(balance_in_wei, ‘ether’)

USE THE Asynchronous version as synchronous version NOT supported by MetaMask

Lock/Unlock Accounts



UnLock & Lock Accounts

Unlock Account

To 0x09f6513525305...

Password password

Un/Lock Result

0x09f651352530526d2a...Unlocked

```
function    doLockAccount()
```

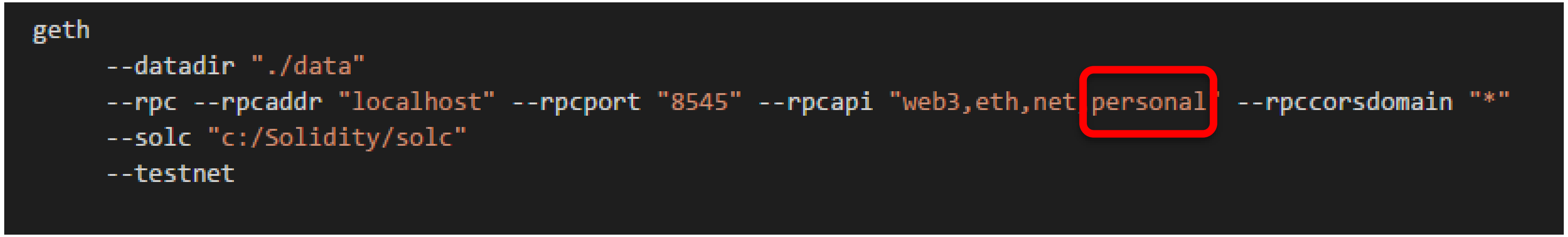
```
web3.personal.lockAccount(account, function(error, result){
```

```
function    doUnlockAccount()
```

```
web3.personal.unlockAccount(account, password,function(error, result)  {
```

RPC API

- Ensure that “personal” API is enabled for RPC



```
geth
     --datadir "./data"
     --rpc --rpcaddr "localhost" --rpcport "8545" --rpcapi "web3,eth,net,personal" --rpccorsdomain "*"
     --solc "c:/Solidity/solc"
     --testnet
```

Unlock Accounts

- Unlock API

```
web3.personal.unlockAccount(account, password, duration)

web3.personal.unlockAccount(account, password, callback_func)

                    Success:  result = true
                    Failure:  error = “Reason for failure”
```

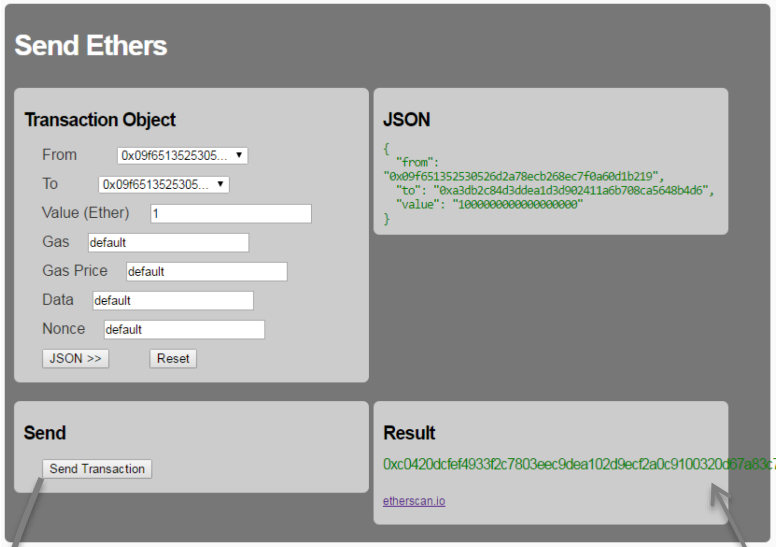
Lock Account

- Lock Account API

```
web3.personal.lockAccount(account)

web3.personal.lockAccount(account, callback_func)
```

Send Ethers



Send Ethers

Transaction Object

From 0x09f6513525305...
To 0x09f6513525305...
Value (Ether) 1
Gas default
Gas Price default
Data default
Nonce default

JSON

```
{
  "from":
"0x09f651352530526d2a78ecb268ec7f0a60d1b219",
  "to": "0xa3db2c84d3ddea1d3d902411a6b708ca5648b4d6",
  "value": "1000000000000000000"
}
```

Send

Result

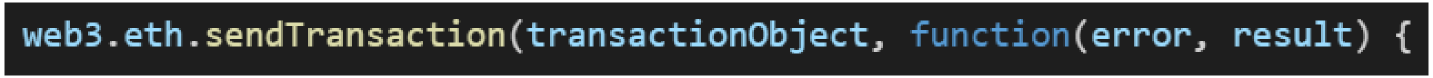
0xc0420dcfef4933f2c7803eec9dea102d9ecf2a0c9100320d67a83c

etherscan.io

```
function    doSendTransaction()
```

```
web3.eth.sendTransaction(transactionObject, function(error, result) {
```

sendTransaction



```
web3.eth.sendTransaction(transactionObject, function(error, result) {
```

- Sending ethers

- Invoking contracts

Success: result = Transaction Hash
Failure: error

Transaction Object

If not specified then *web3.eth.defaultAccount*

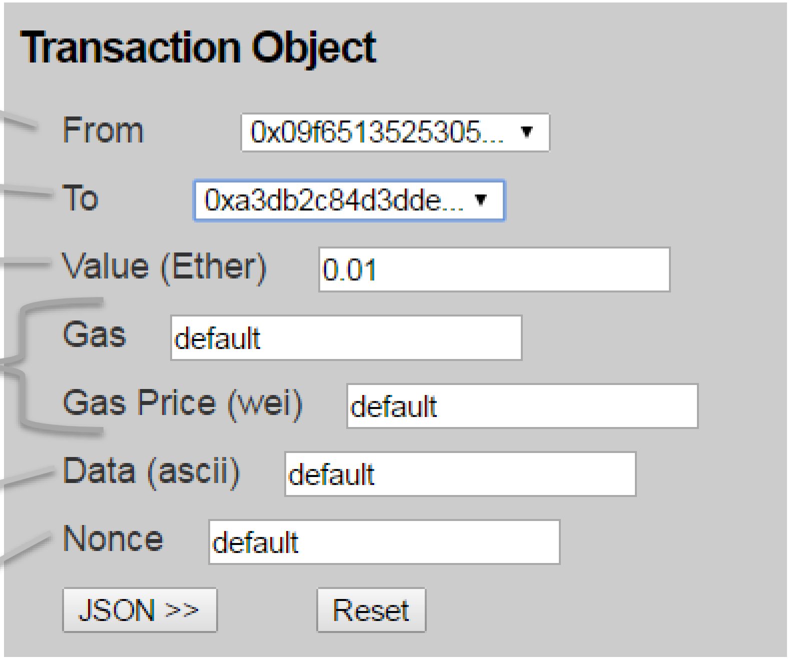
To Account

Value in Wei

Txn fee paid by originator
Fee=gas*gasPrice

Data | Contract call
In Hex

Overwrite pending



Transaction Object

From	0x09f6513525305...
To	0xa3db2c84d3dde...
Value (Ether)	0.01
Gas	default
Gas Price (wei)	default
Data (ascii)	default
Nonce	default

Web3 JS API:

- Deployment

Discount Coupon Links to UDEMY courses:

https://www.udemy.com/hyperledger/?couponCode=DKHLF1099

https://www.udemy.com/ethereum-dapp/?couponCode=DKETH1099

https://www.udemy.com/rest-api/?couponCode=DKRST1099

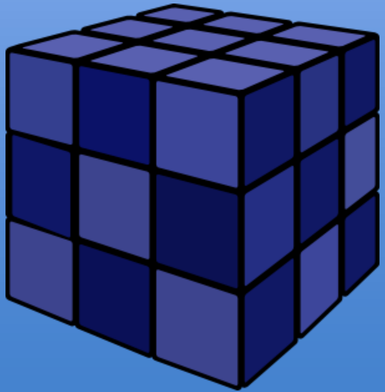
mentoring, seeking Blockchain part time work, project guidance, advice … …
http://www.bcmentors.com

This deck is part of a online course on "Ethereum: Design and Development of Decentralized Apps.

raj@acloudfan.com

@acloudfan

http://ACloudFan.com



Contract deployment

- Deployment is recorded as a transaction on the chain/ledger
- Contract available after its been mined
- Deployment is not free
  - Originator of the deployment transaction pays
- Bytecode deployed to all nodes

Contract Object

var contract = web3.eth.**contract**(*abiDefinition Array*)

1. Deploying the contract code to EVM
2. Invoking a contract function
3. Watch for events from contract instance

Deployment using **new(...)**

- Synchronous

```
var contractInstance = contract.new ( Constructor_Param1, Constructor_Param2 ....,
                                      { from: web3.eth.coinbase,
                                        data: bytecode,
                                        gas: gas    }
                                                                  )
```

- contractInstance.transactionHash << Transaction created
- contractInstance.address << Filled after the txn is mined

Deployment using **new(…)**

- Asynchronous

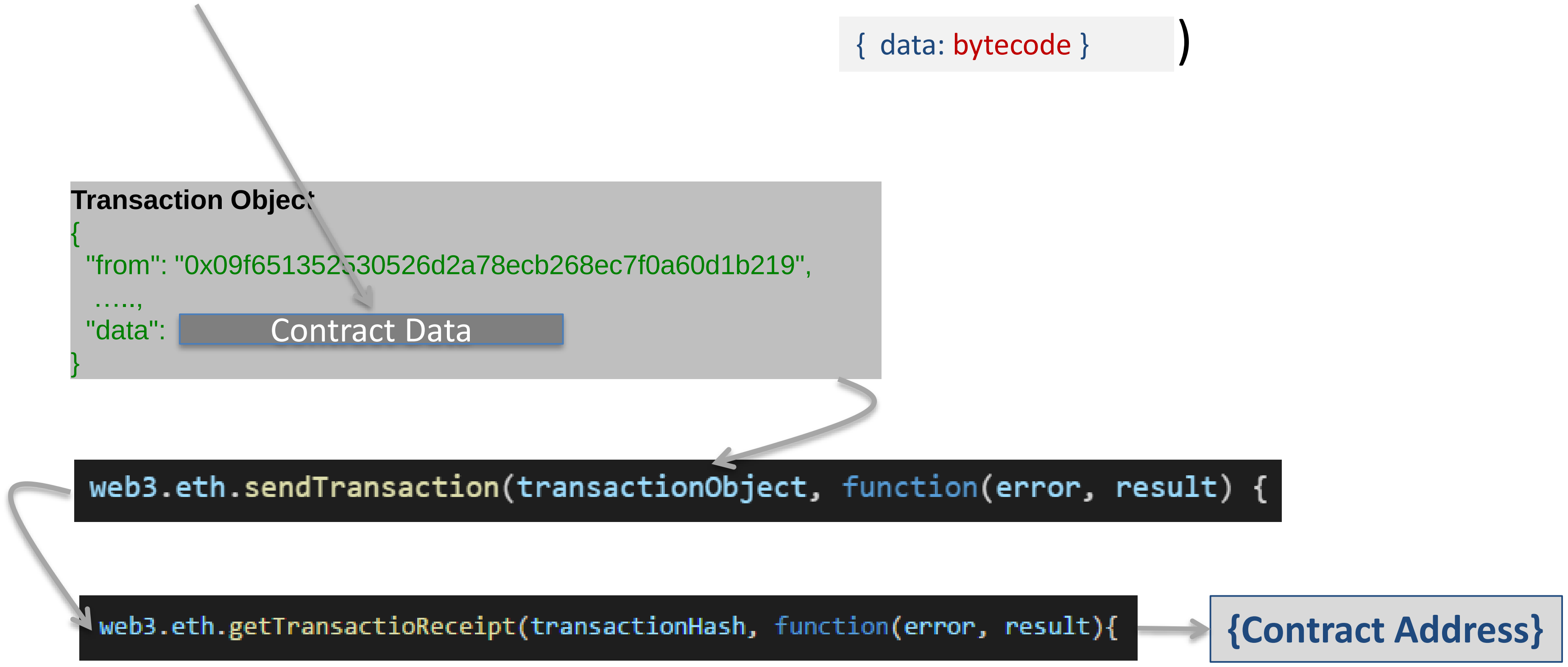
```
contract.new ( Constructor_Param1, Constructor_Param2 ….,
               {  from: web3.eth.coinbase,
                  data: bytecode,
                  gas: gas    } ,
                Callback(error, result){….} )
```

- Callback function gets called 2 times in case of success

1. Result = Transaction Hash
2. Result = Contract Instance Address

Deployment using **sendTransaction**

```
var conData = contract.new.getData( Constructor_Param1, Constructor_Param2 ….,
                                    { data: bytecode } )
```



Transaction Object

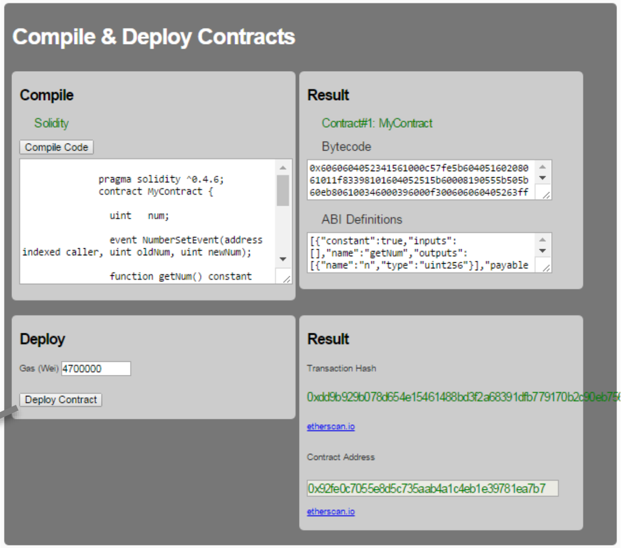
```
{
  "from": "0x09f651352530526d2a78ecb268ec7f0a60d1b219",
   …..,
  "data": Contract Data
}
```

```
web3.eth.sendTransaction(transactionObject, function(error, result) {
```

```
web3.eth.getTransactioReceipt(transactionHash, function(error, result){
```

{Contract Address}

Deploy Contract



Compile & Deploy Contracts

Compile

Solidity

Compile Code

```
            pragma solidity ^0.4.6;
            contract MyContract {

              uint   num;

              event NumberSetEvent(address
indexed caller, uint oldNum, uint newNum);

              function getNum() constant
```

Result

Contract#1: MyContract

Bytecode

```
0x606060405234156100c57fe5b604051602080
61011f83398101604052515b60008190555b505b
60eb806100346000396000f300606060405263ff
```

ABI Definitions

```
[{"constant":true,"inputs":
[],"name":"getNum","outputs":
[{"name":"n","type":"uint256"}],"payable
```

Deploy

Gas (Wei) 4700000

Deploy Contract

Result

Transaction Hash

0xdd9b929b078d654e15461488bd3f2a68391dfb779170b2c90eb75

etherscan.io

Contract Address

0x92fe0c7055e8d5c735aab4a1c4eb1e39781ea7b7

etherscan.io

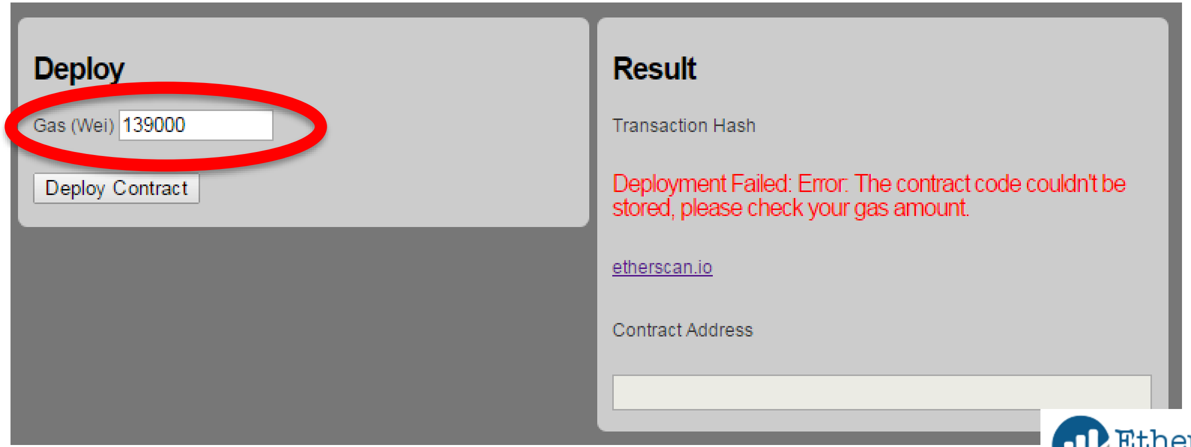
```
contract.new(constructor_param,params,function(error,result){
    // CALLBACK Gets called 2 time
});
```

#1 result >> Transaction Hash

#2 result >> Contract Address

```
function    doDeployContract()
```

Deployment Cost



Deploy

Gas (Wei) 139000

Deploy Contract

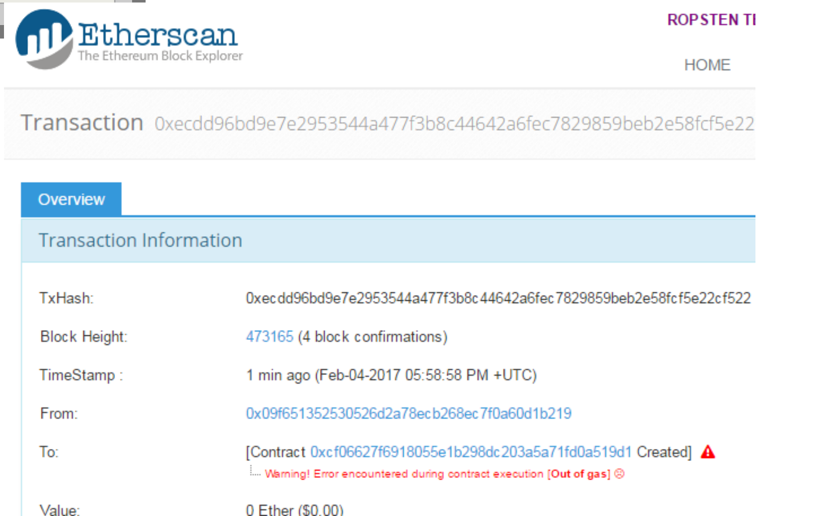
Result

Transaction Hash

Deployment Failed: Error: The contract code couldn't be stored, please check your gas amount.

etherscan.io

Contract Address



Etherscan
The Ethereum Block Explorer

ROPSTEN TI

HOME

Transaction 0xecdd96bd9e7e2953544a477f3b8c44642a6fec7829859beb2e58fcf5e22

Overview

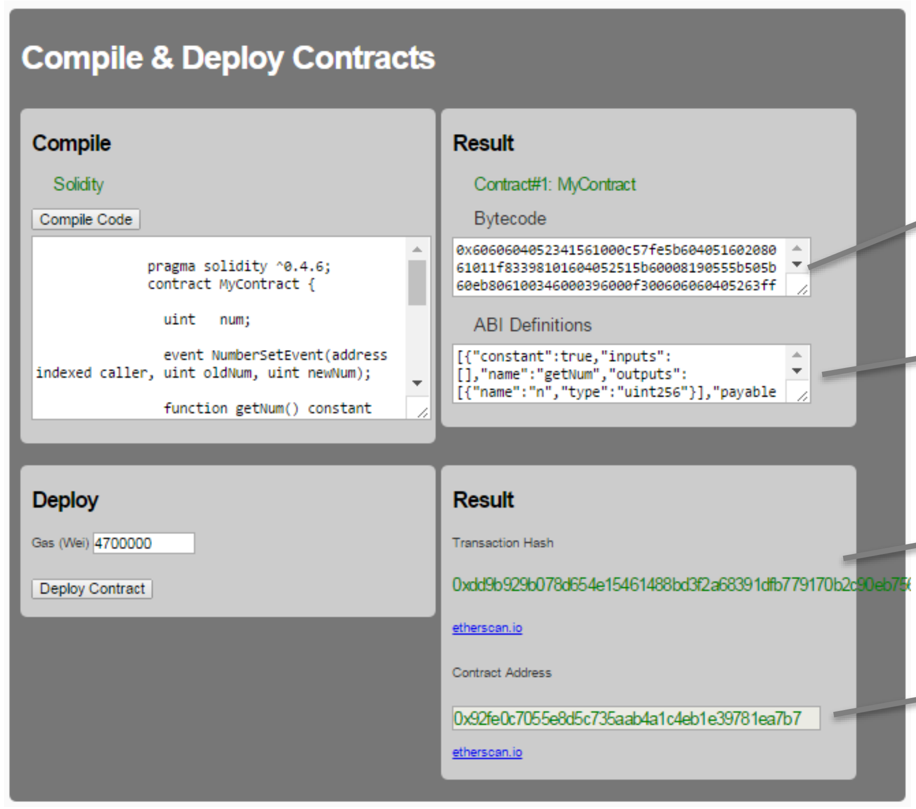
Transaction Information

TxHash:	0xecdd96bd9e7e2953544a477f3b8c44642a6fec7829859beb2e58fcf5e22cf522
Block Height:	473165 (4 block confirmations)
TimeStamp :	1 min ago (Feb-04-2017 05:58:58 PM +UTC)
From:	0x09f651352530526d2a78ecb268ec7f0a60d1b219
To:	[Contract 0xcf06627f6918055e1b298dc203a5a71fd0a519d1 Created] ⚠ Warning! Error encountered during contract execution [Out of gas] ☹
Value:	0 Ether ($0.00)

#1 result >> Transaction Hash

#2 **Error**

Deploy Contract



Compile & Deploy Contracts

Compile

Solidity

Compile Code

```
              pragma solidity ^0.4.6;
              contract MyContract {

                uint   num;

                event NumberSetEvent(address
indexed caller, uint oldNum, uint newNum);

                function getNum() constant
```

Result

Contract#1: MyContract

Bytecode

```
0x6060604052341561000c57fe5b604051602080
61011f83398101604052515b60008190555b505b
60eb806100346000396000f30060606040526 3ff
```

ABI Definitions

```
[{"constant":true,"inputs":
[],"name":"getNum","outputs":
[{"name":"n","type":"uint256"}],"payable
```

Deploy

Gas (Wei) 4700000

Deploy Contract

Result

Transaction Hash

0xdd9b929b078d654e15461488bd3f2a68391dfb779170b2c90eb75

etherscan.io

Contract Address

0x92fe0c7055e8d5c735aab4a1c4eb1e39781ea7b7

etherscan.io

Bytecode (Data) deployed on chain

Needed by the caller of functions

Transaction on the chain

Address of contract

Contract Instance

1. ABI Definition

   *var contract = web3.eth.contract(**abiDefinition**)*

2. Address of the contract

   *var contractInstance = contract.**at**(address)*

Web3 JS API:

- Call()
- sendTransaction()

Discount Coupon Links to UDEMY courses:

https://www.udemy.com/hyperledger/?couponCode=DKHLF1099

https://www.udemy.com/ethereum-dapp/?couponCode=DKETH1099

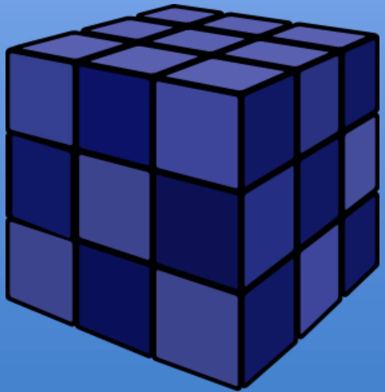
https://www.udemy.com/rest-api/?couponCode=DKRST1099

mentoring, seeking Blockchain part time work, project guidance, advice … …
http://www.bcmentors.com

raj@acloudfan.com

@acloudfan

http://ACloudFan.com



This deck is part of a online course on "Ethereum: Design and Development of Decentralized Apps.

Method Invocation

1. Call(…)

   Cost of Call = 0 ETH

   contractInstance.Method.call(…)

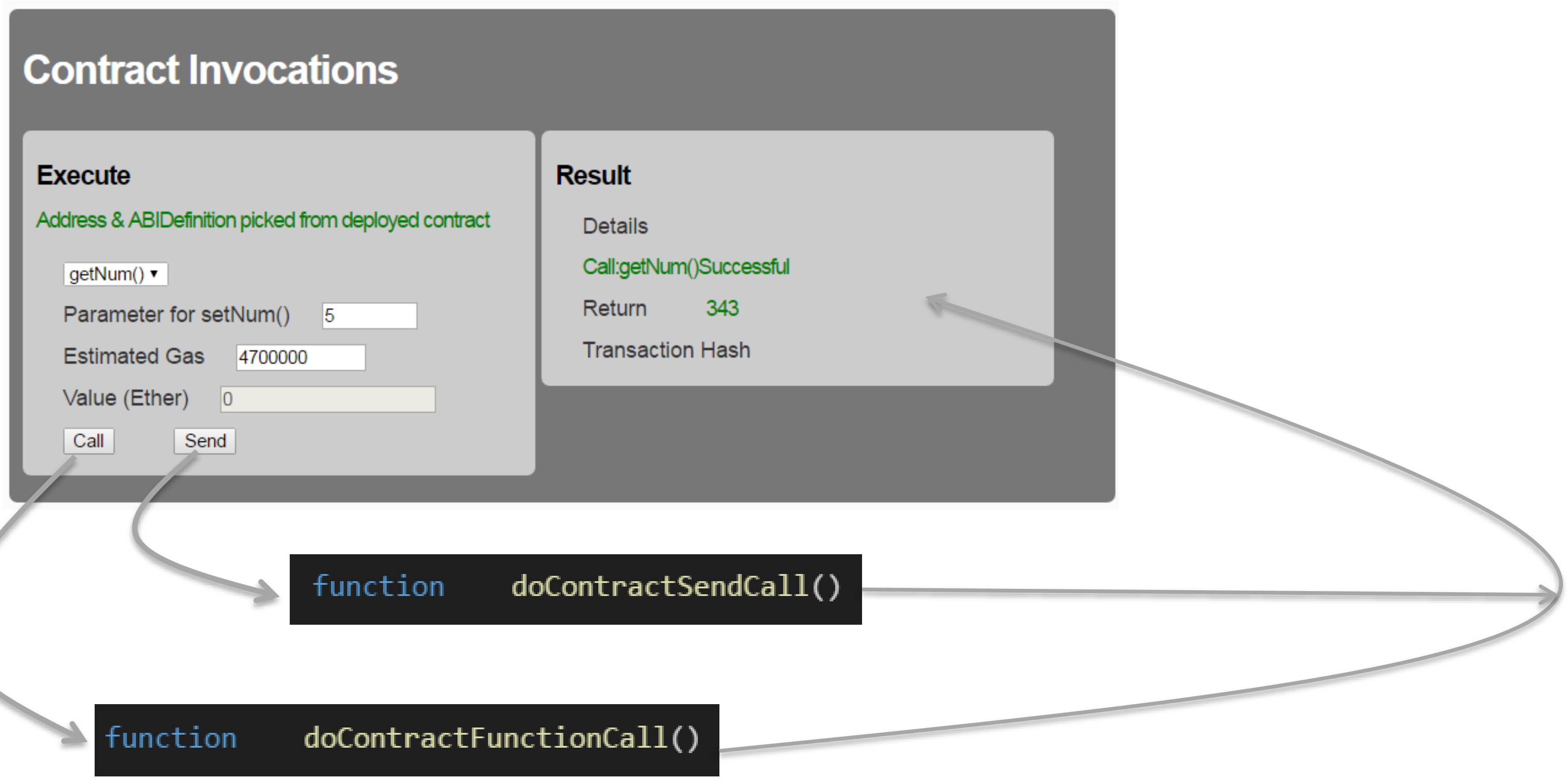
2. sendTransaction(…)

   Cost of Send = Gas paid by caller

   contractInstance.Method.sendTransaction(…)

Method.call(…)	*Method.sendTransaction(…)*
• Executed locally on the node	• Executed on miner nodes
• Value= Return value from function	• Value= Transaction hash
• No state changes in contract	• State changes in contracts
• 0 execution fee	• Gas paid by caller

Call() & sendTransaction()


Contract Invocations

Execute

Address & ABIDefinition picked from deployed contract

getNum()

Parameter for setNum() 5

Estimated Gas 4700000

Value (Ether) 0

Call Send

Result

Details

Call:getNum()Successful

Return 343

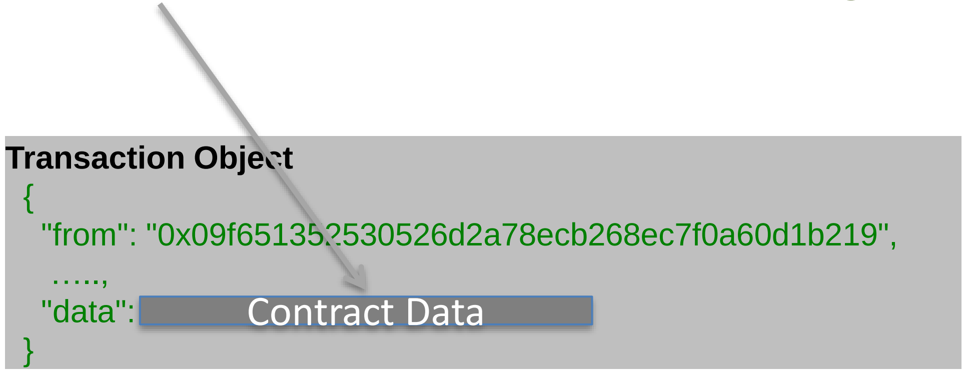
Transaction Hash

```
function    doContractSendCall()
```

```
function    doContractFunctionCall()
```

web3.eth.call web3.eth.sendTransaction

*Var conData = contractInstance. Method. **getData**(param1, param2 …)*



Transaction Object

```
{
  "from": "0x09f6513522530526d2a78ecb268ec7f0a60d1b219",
   .....,
  "data": Contract Data
}
```

var result = web3.eth.call(transaction_object, [default block], [callback])

var result = web3.eth.sendTransaction(transaction_object…)

call()

From: is optional

var result = web3.eth.call(transaction_object, [default block], [callback])

"latest" by default

var result = contractInstance.Method. call(params,...,
[transaction_object],
[default block],
[callback])