Introduction to Blockchain

DApp Academy

DAY 2

Instructors: Vinoth Kumar, Abhishek Chowdhury

PEERBUDS INNOVATION LAB

Web3 JS API:

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

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

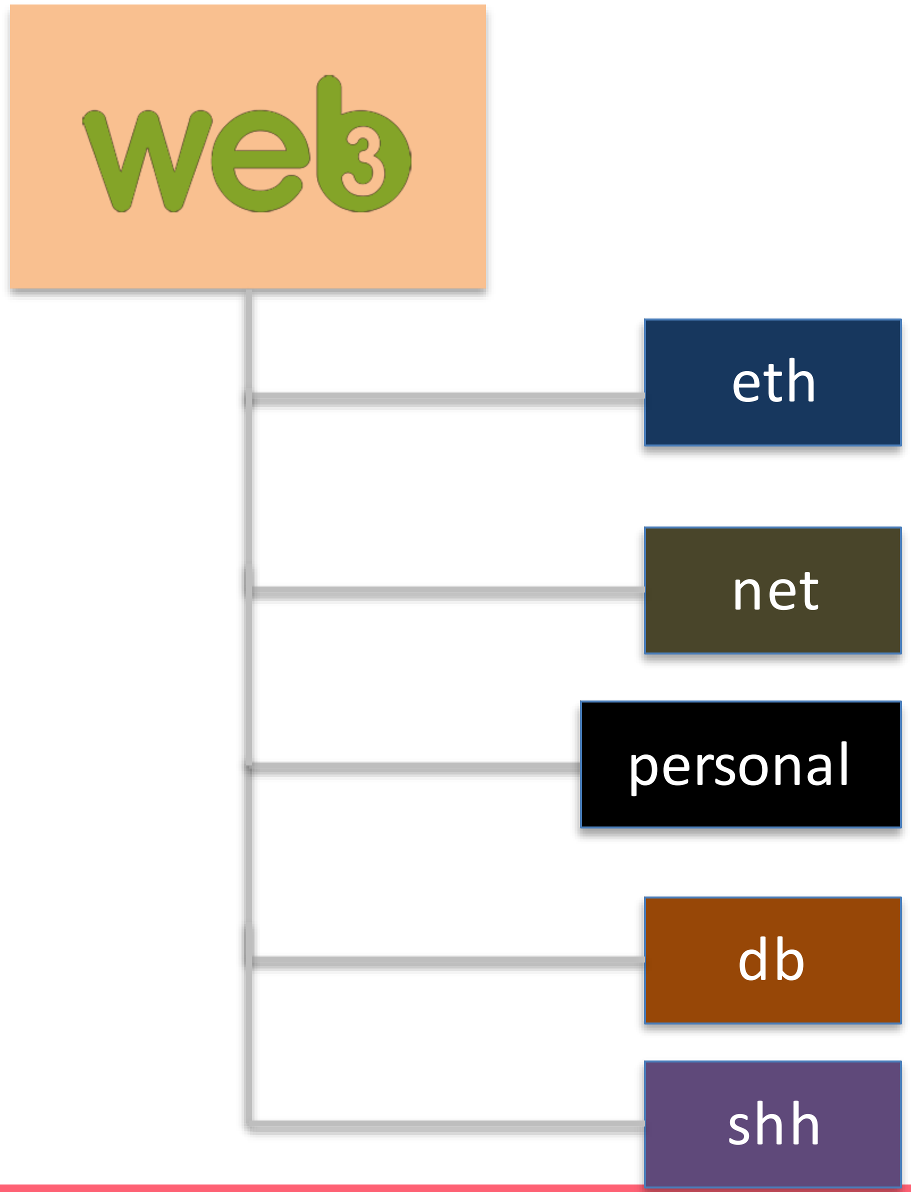
  - Even BigNumber cannot handle more than 20 floating points

  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

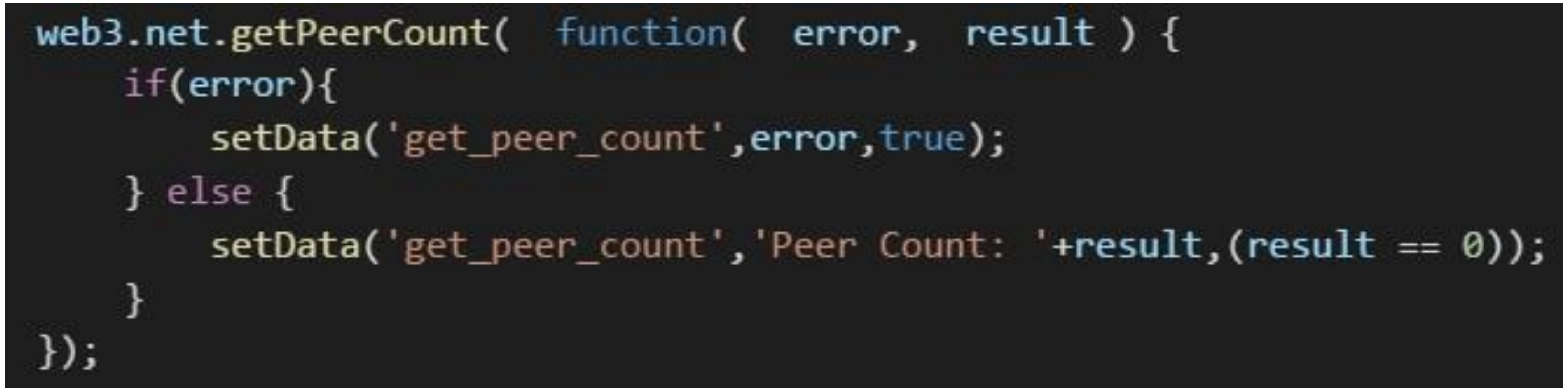
Account functions and sending

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

```
> npm install – g yo
```

Install Yeoman webapp template

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

3 Install Gulp

```
> npm install – g gulp
```

4 Install Bower

```
> npm install – g bower
```

Setup Dapp

1 Create a folder for application

2 Create the application

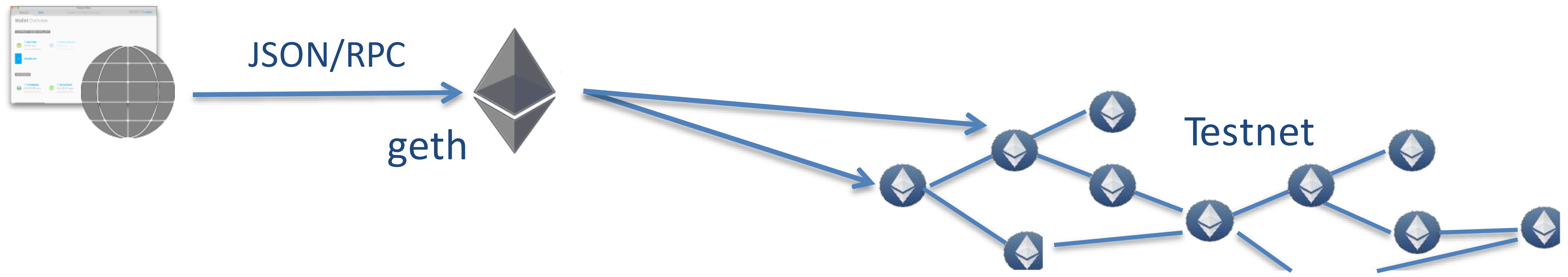
```
> yo  webapp
```

3 Install web3 library

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

Web3 Sample App

- 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*



Workbench DAPP will work with:

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

Download & Setup Sample Application Locally

1. Download the application
2. Unpack the zip file in a directory

https://github.com/abhishekover9000/web3-dapp-demo.git

3. Run
```
> npm install
```

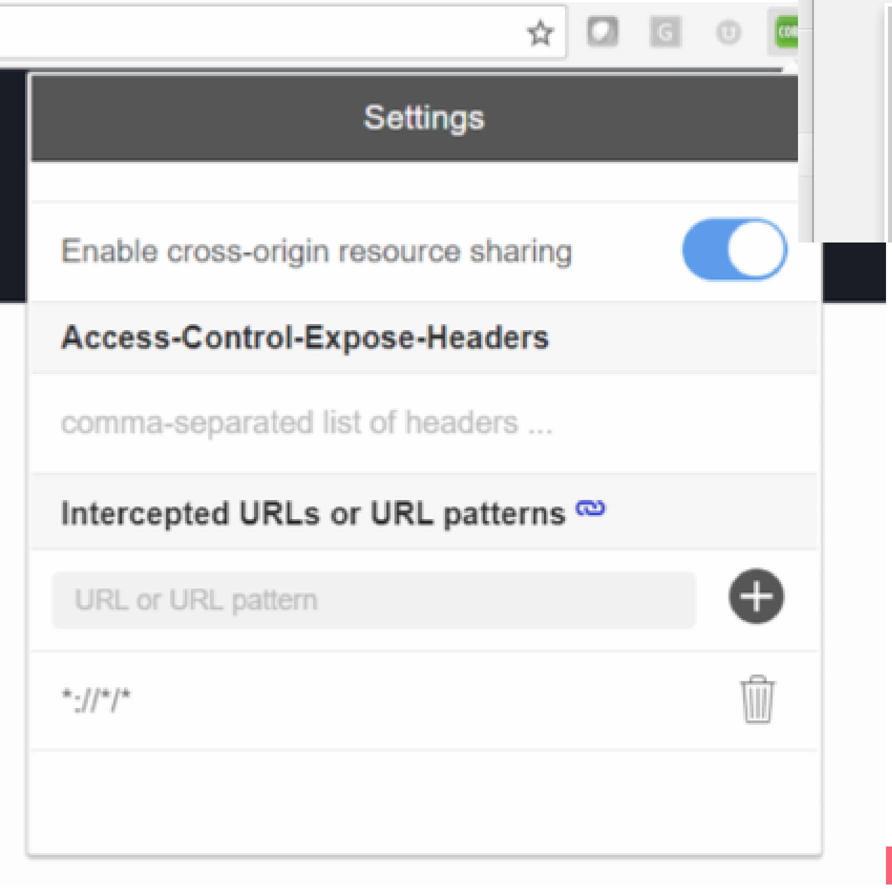
4. Run
```
> gulp serve
```

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

Enable CORS (Cross original resource sharing)



Settings

Enable cross-origin resource sharing

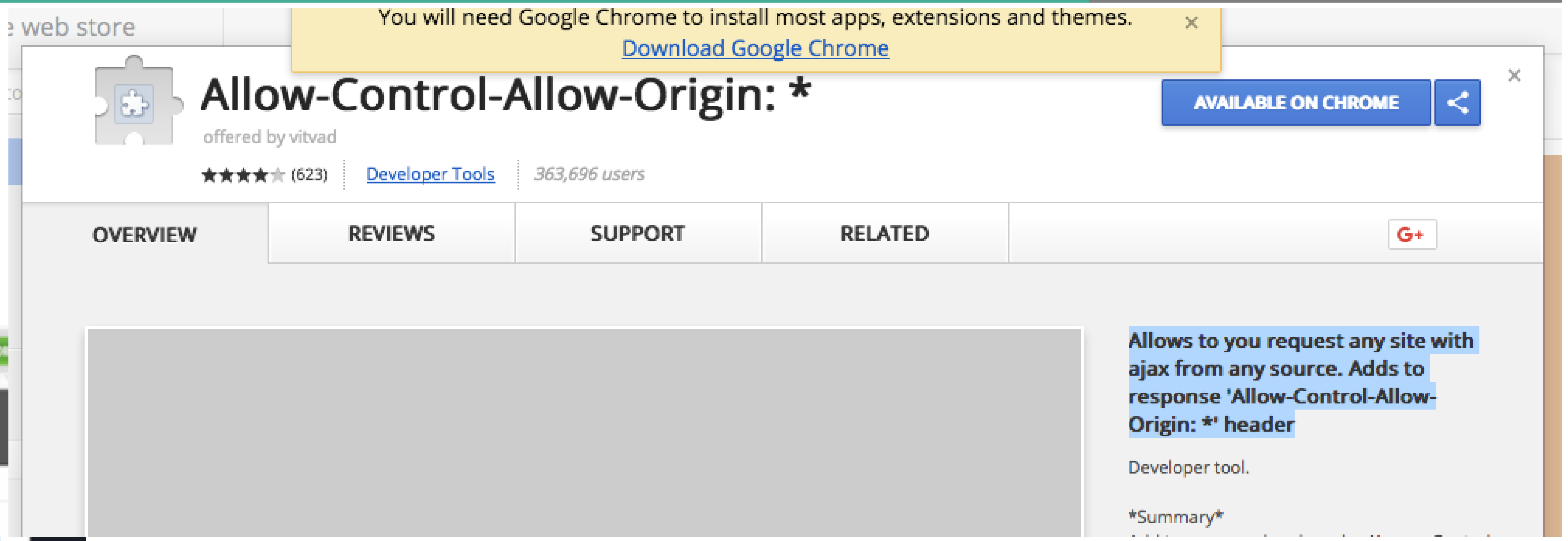
Access-Control-Expose-Headers

comma-separated list of headers ...

Intercepted URLs or URL patterns

URL or URL pattern

:///*



web store

You will need Google Chrome to install most apps, extensions and themes.
Download Google Chrome

Allow-Control-Allow-Origin: *

offered by vitvad

★★★★☆ (623) | Developer Tools | 363,696 users

AVAILABLE ON CHROME

OVERVIEW | REVIEWS | SUPPORT | RELATED

Allows to you request any site with ajax from any source. Adds to response 'Allow-Control-Allow-Origin: *' header

Developer tool.

Summary

App Structure

inde

- HTML UI Components

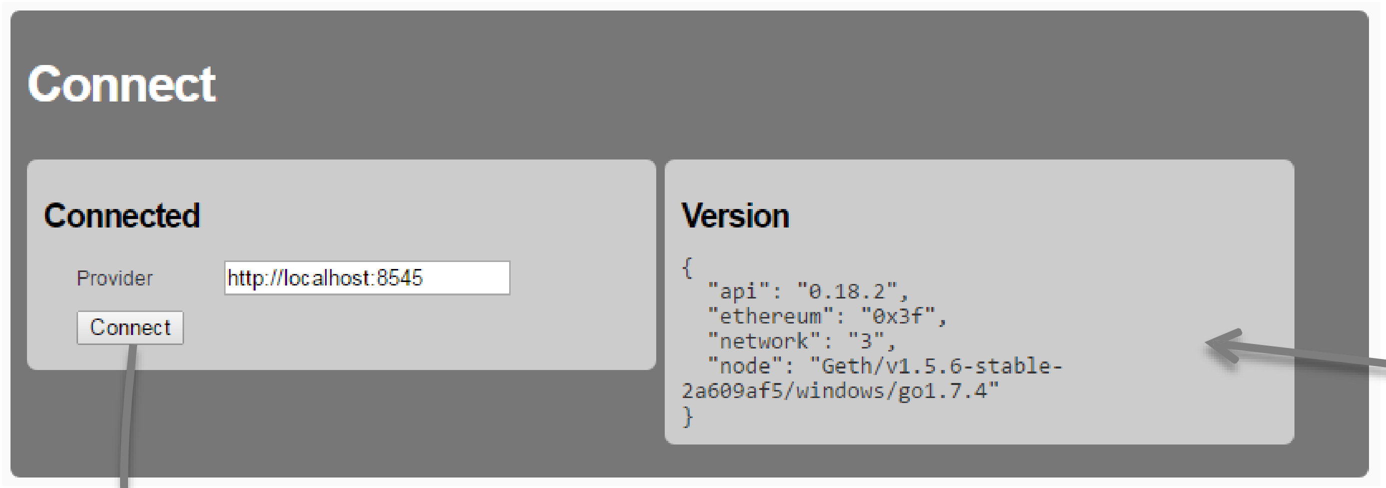
main.js

- 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));
```

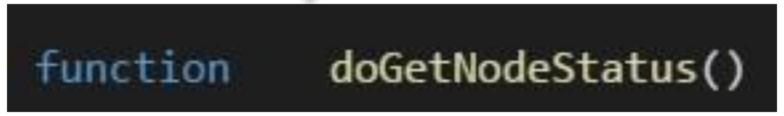
```
web3.version
```

web3.net

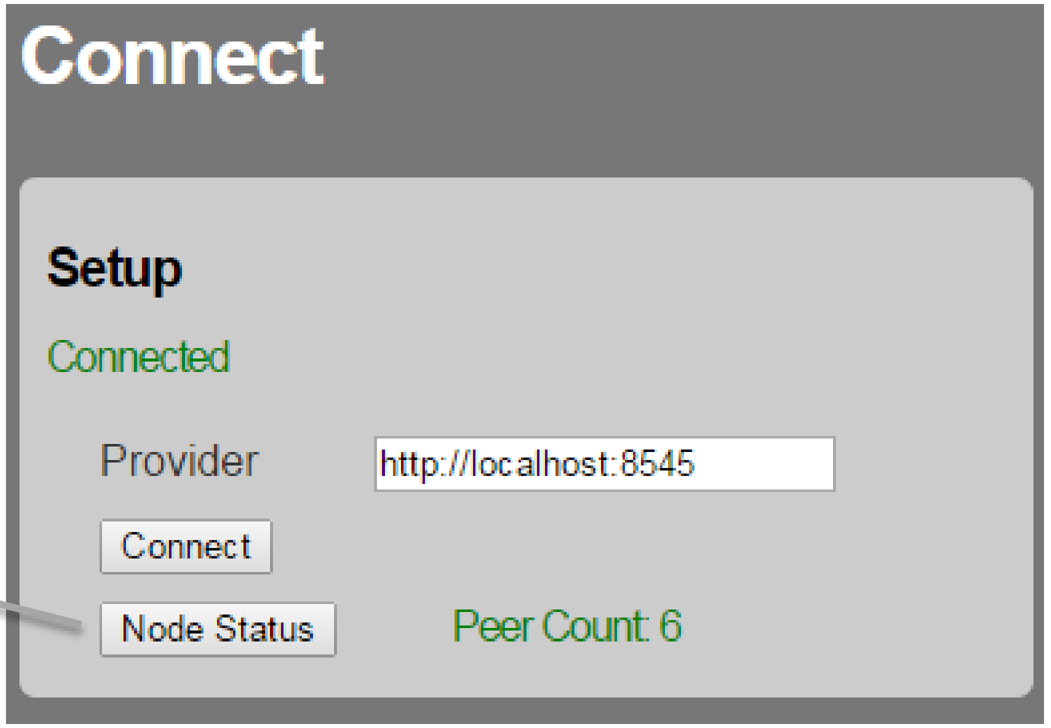
- web3.net

listening / getListening

peerCount/ getPeerCount



```
function    doGetNodeStatus()
```



Connect

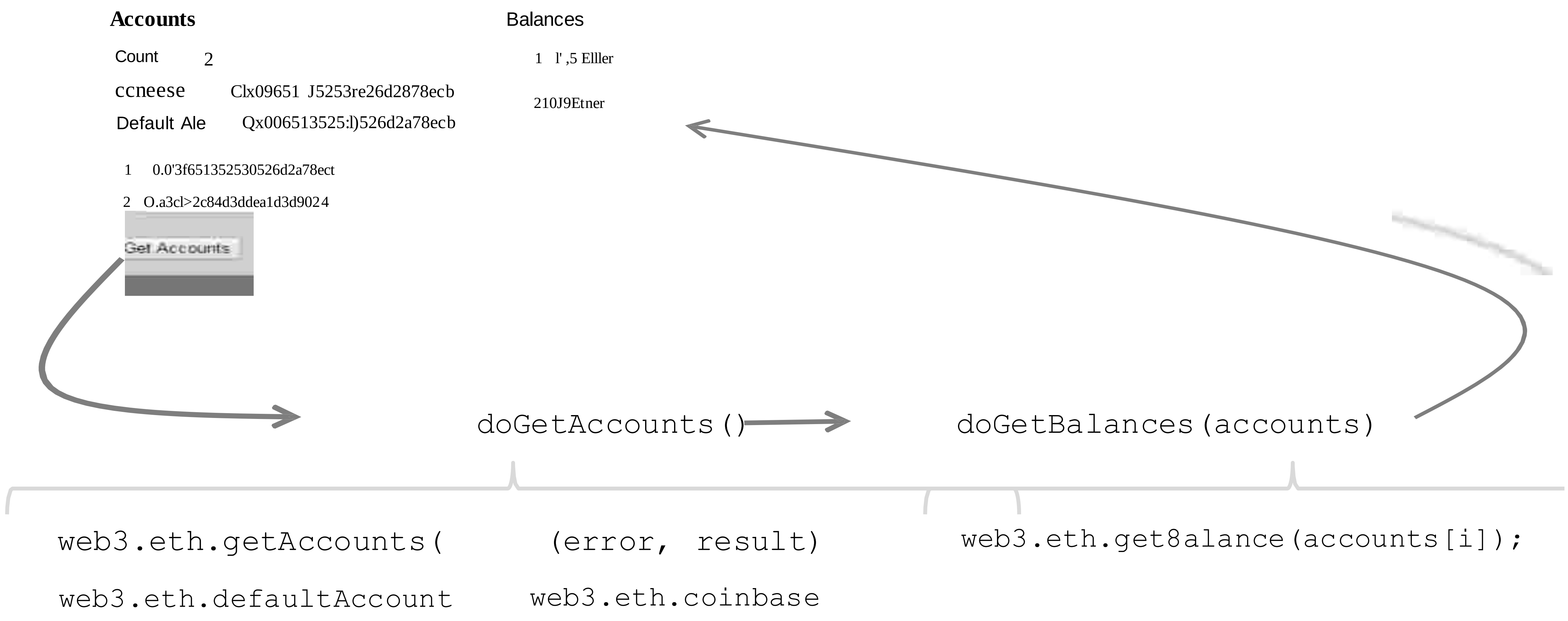
Setup

Connected

Provider http://localhost:8545

Peer Count: 6

Get Accounts & Balances



Accounts

Count	2
ccneese	Clx09651 J5253re26d2878ecb
Default Ale	Qx006513525:l)526d2a78ecb

1 0.0'3f651352530526d2a78ect

2 O.a3cl>2c84d3ddea1d3d9024

Balances

1 l' ,5 Elller

210J9Etner

```
doGetAccounts()          doGetBalances(accounts)
```

```
web3.eth.getAccounts(      (error,  result)        web3.eth.get8alance(accounts[i]);

web3.eth.defaultAccount    web3.eth.coinbase
```

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
```

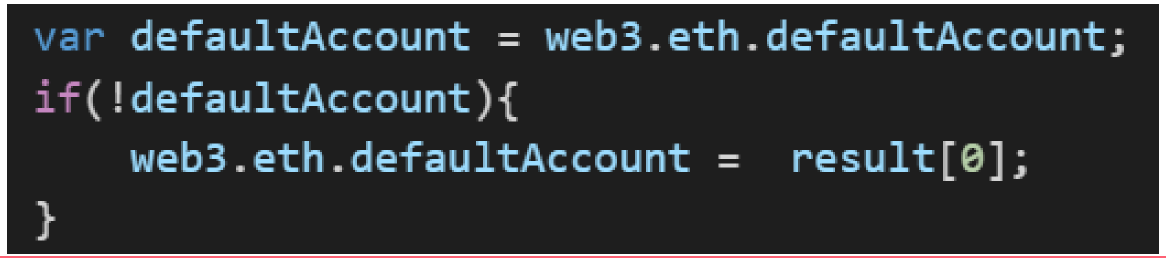
web3.eth.defaultAccount

- Read/Write
- Used in these methods if *from:* 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

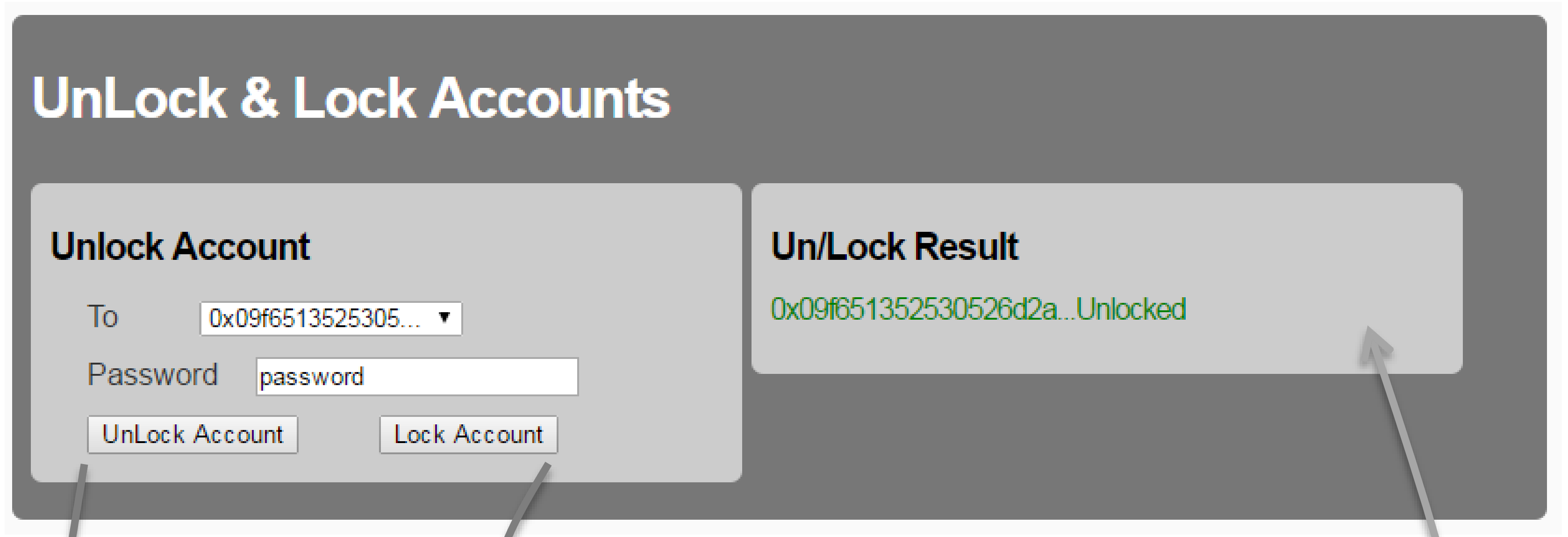
```
web3.eth.getBalance (address)
```

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

UnLock Account   Lock Account

Un/Lock Result

0x09f651352530526d2a...Unlocked

```
function    doLockAccount()
```

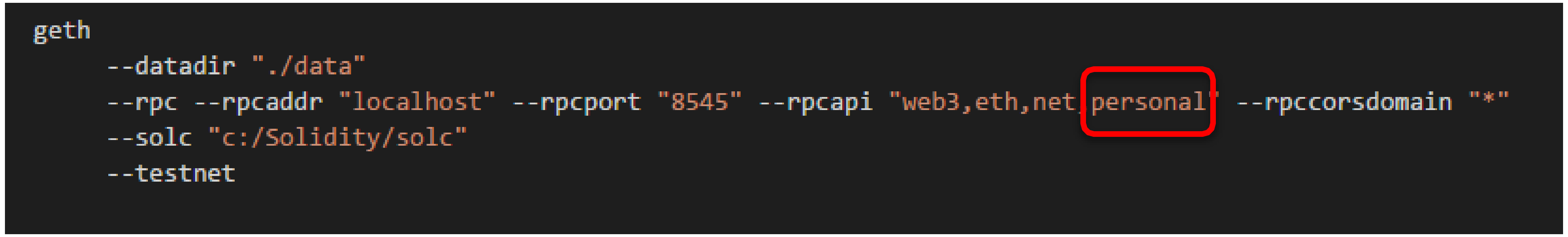
```
web3.personal.lockAccount(account, function(error, result){
```

```
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)

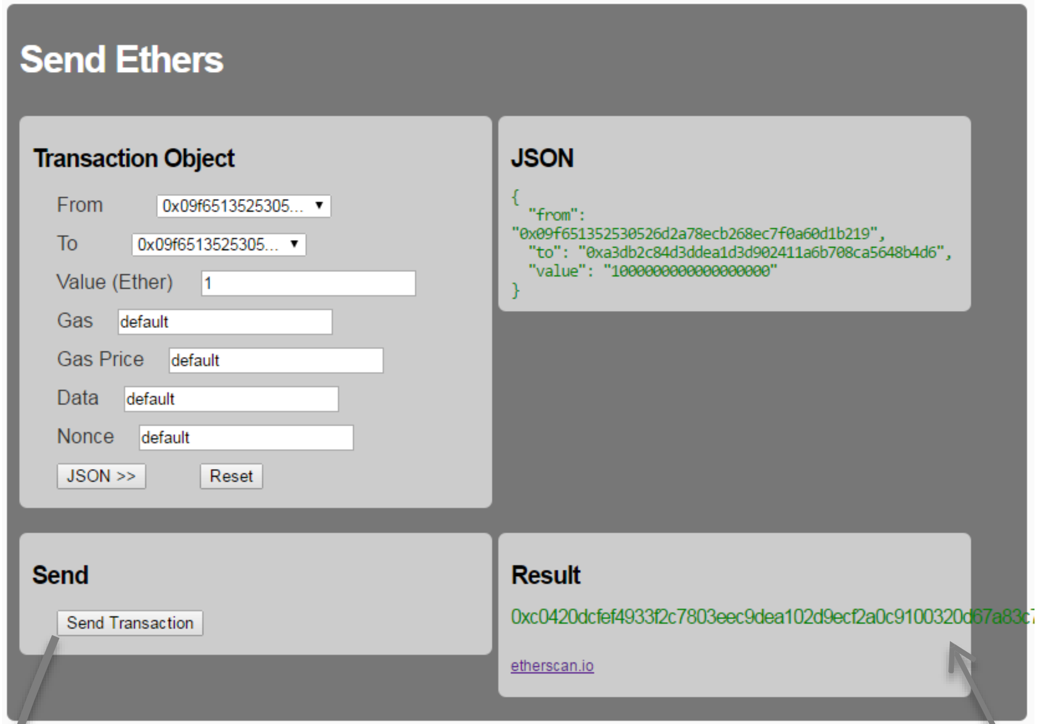
Success: result = true
Failure: error = “Reason for failure”

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

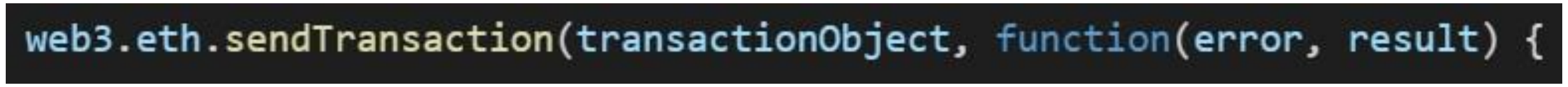
0xc0420dcfef4933f2c7803eec9dea102d9ecf2a0c9100320d67a83c

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*

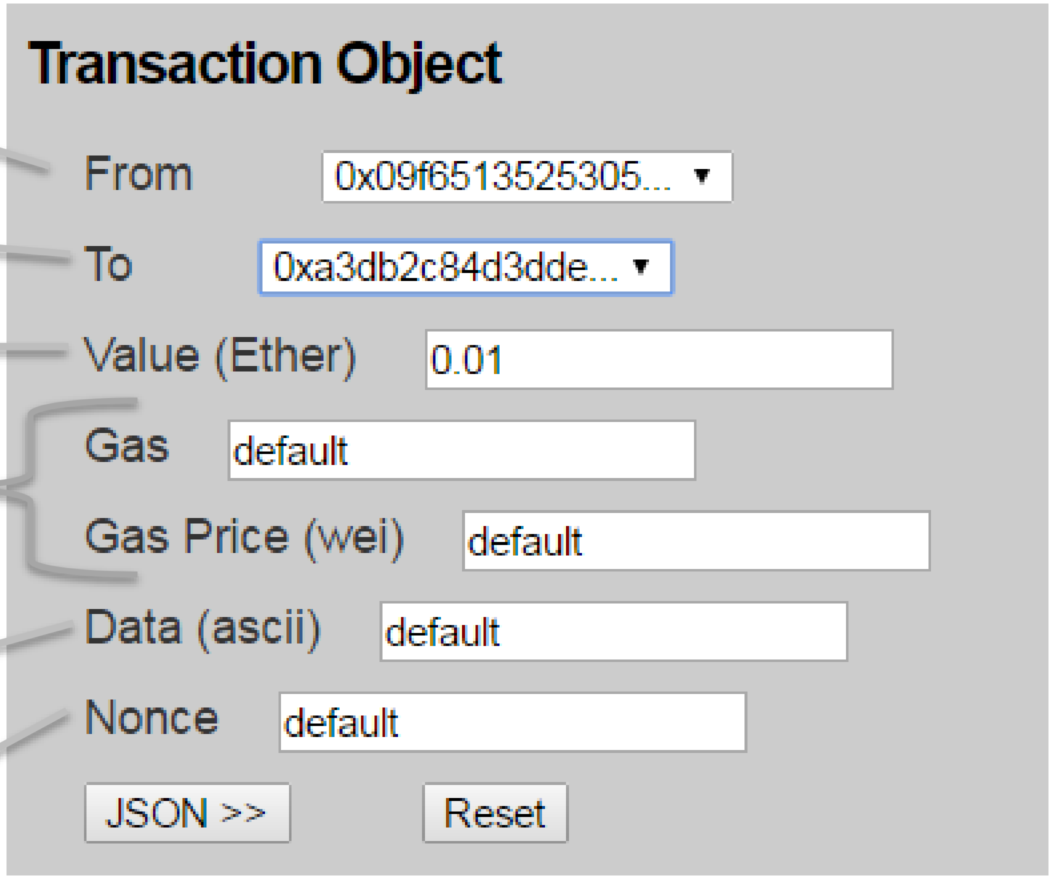
To Account

Value in W i

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

Introduction to Blockchain

DApp Acadamy

Break

Instructors: Vinoth Kumar, Abhishek Chowdhury

Web3 JS API:

- Contract Deployment

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

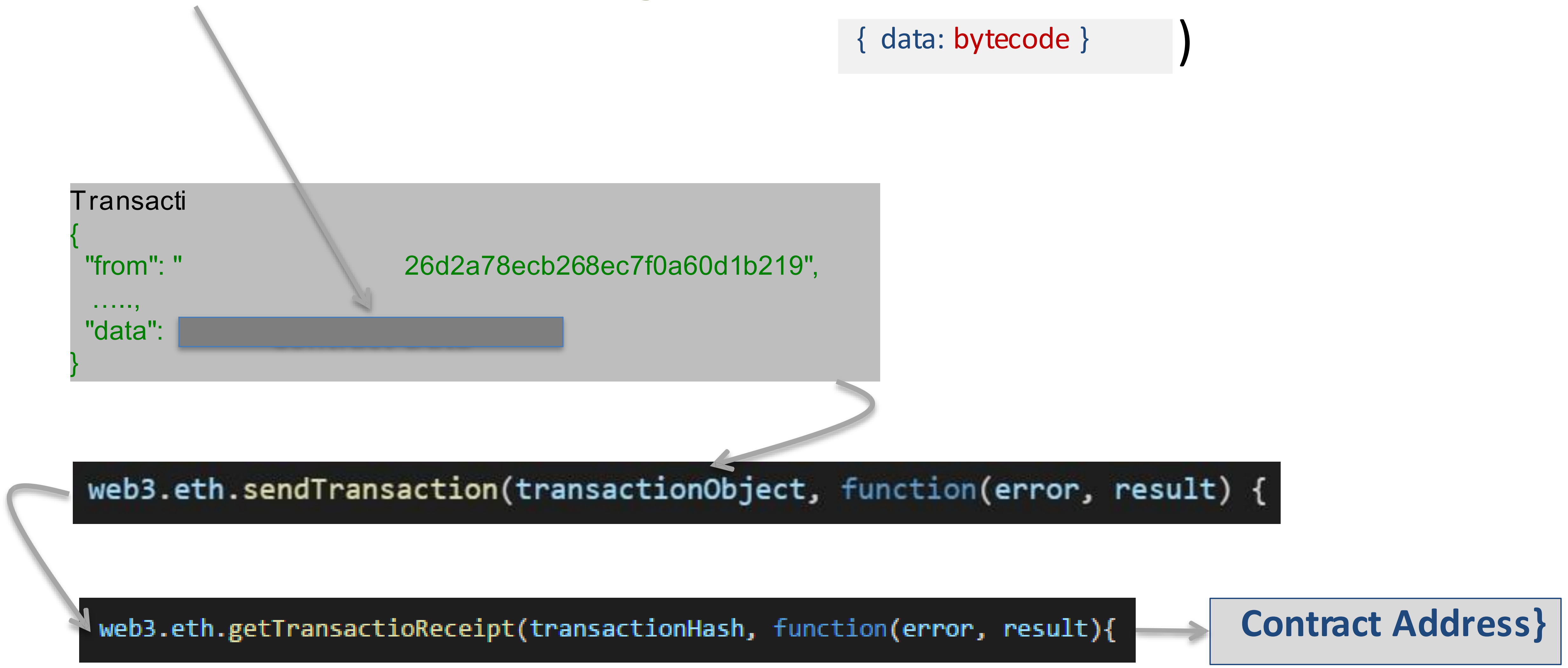
```
contract.new ( Constructor_Param1, Constructor_Param2 ....,
               { from: web3.eth.coinbase,
                 data: bytecode,
                 gas:gas   },
               Callback(error, result){....} )
```

- 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 })



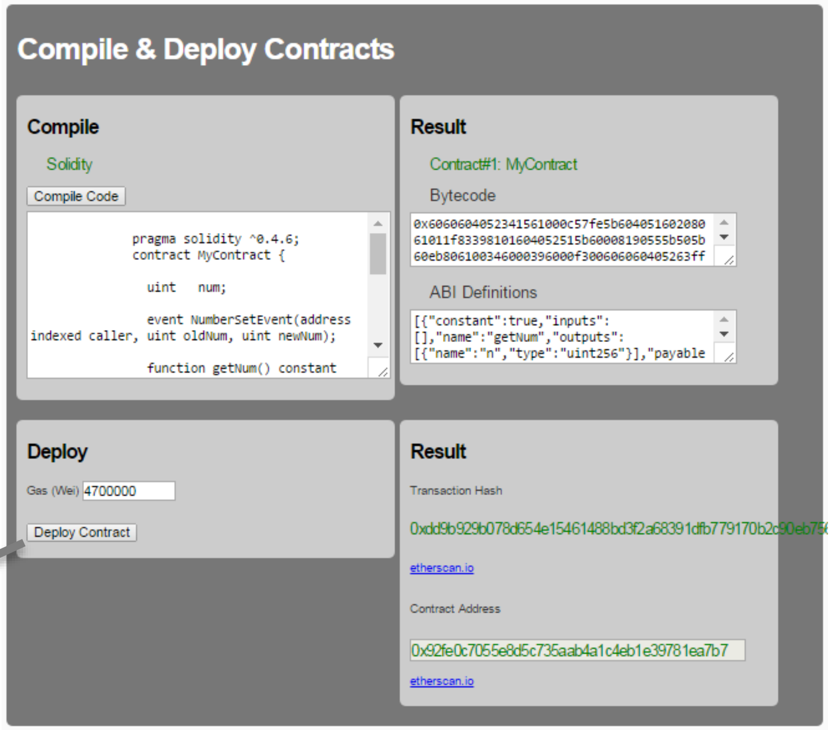
```
Transacti
{
  "from": "                    26d2a78ecb268ec7f0a60d1b219",
   …..,
  "data":
}
```

```
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

```
0x6060604052341561000c57fe5b604051602080
61011f8339810160405251b60008190555b505b
60eb80610034600039600f300606060405263ff
```

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

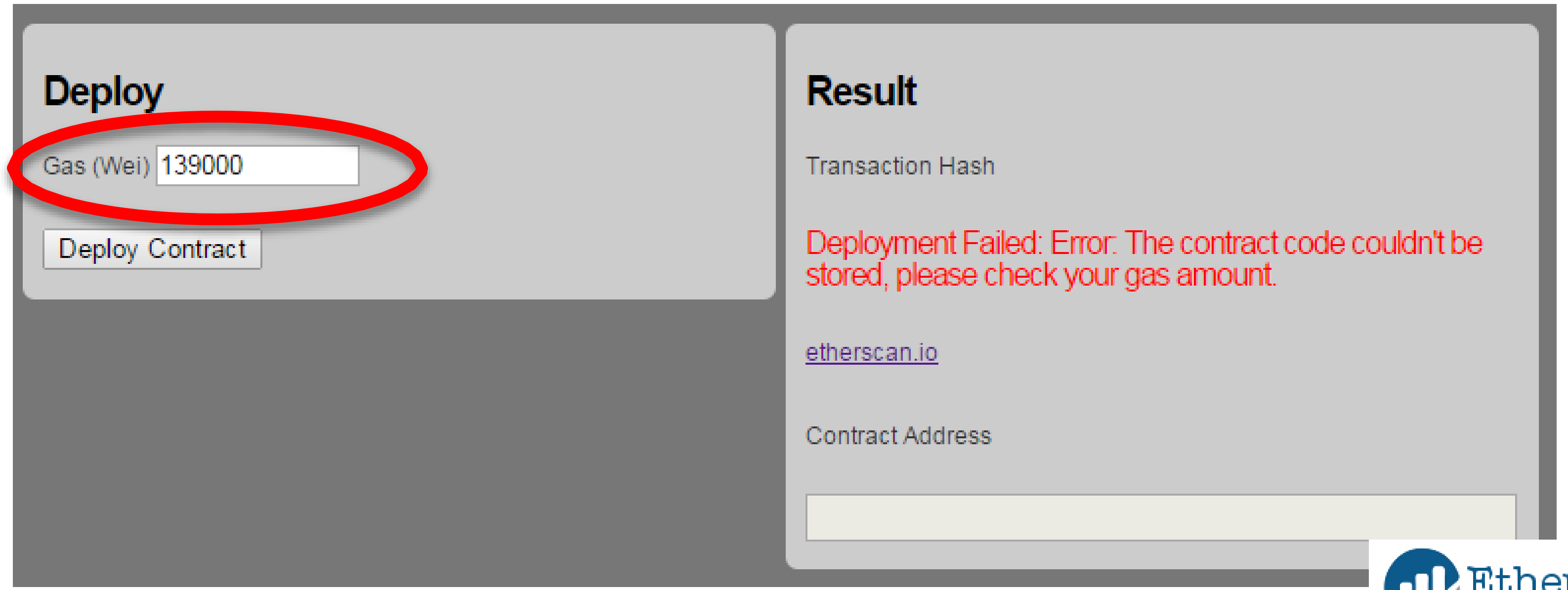
```
contract.new(constructor_param,params,function(error,result){
    // CALLBACK Gets called 2 time
});
```

```
function    doDeployContract()
```

#1 result >> Transaction Hash

#2 result >> Contract Address

Deployment Cost



Deploy

Gas (Wei) 139000

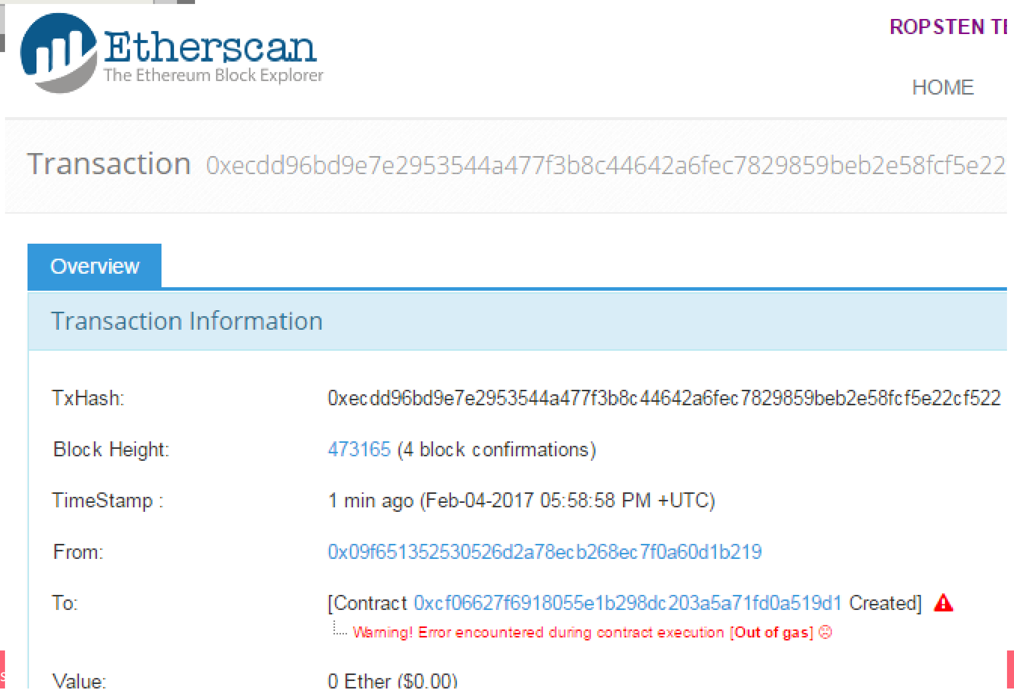
Result

Transaction Hash

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

etherscan.io

Contract Address



Transaction 0xecdd96bd9e7e2953544a477f3b8c44642a6fec7829859beb2e58fcf5e22

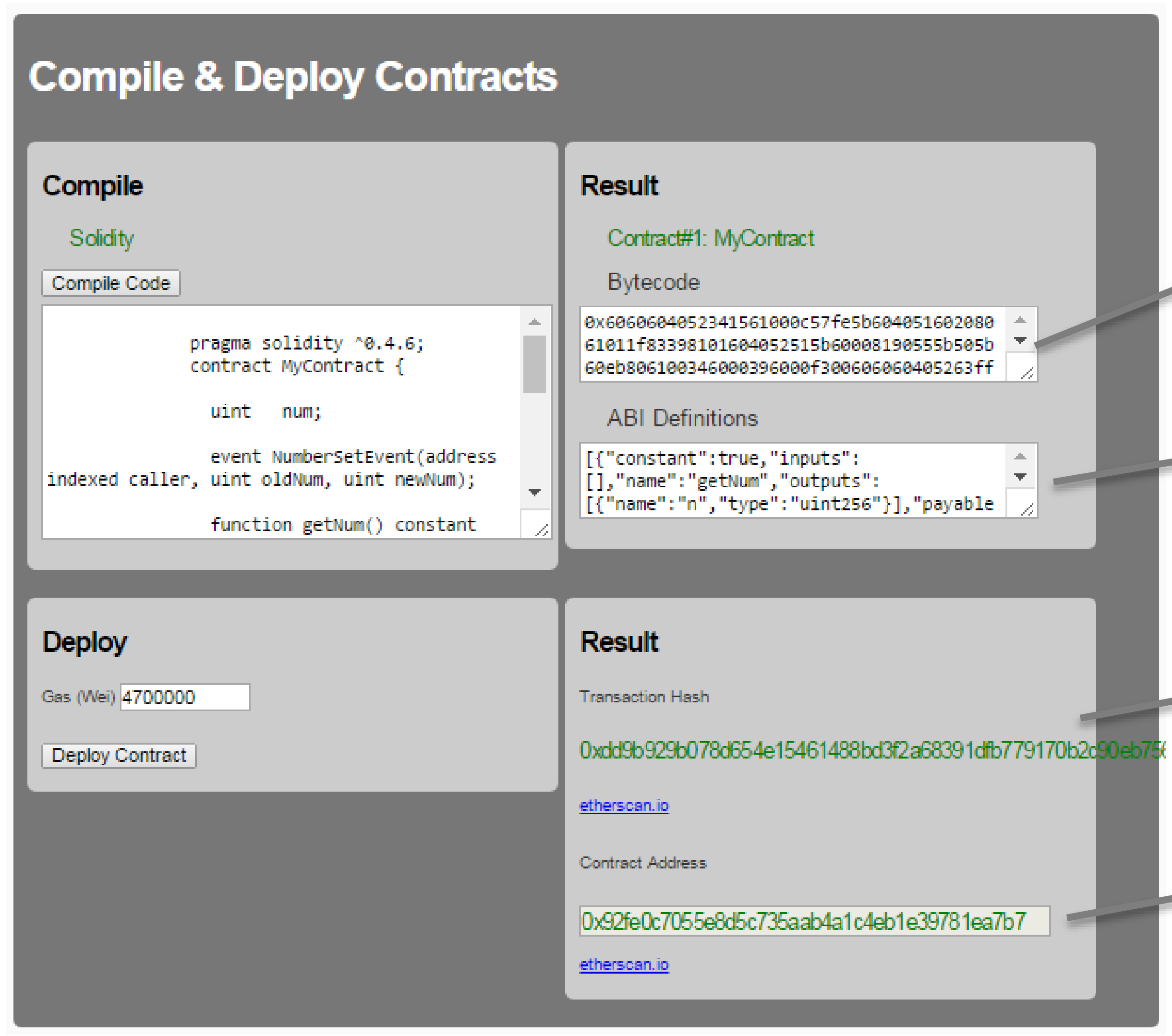
Transaction Information

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()

Method Invocation

1. Call(...)

   Cost of Call = 0 ETH

   contractInstance.Method.call(...)

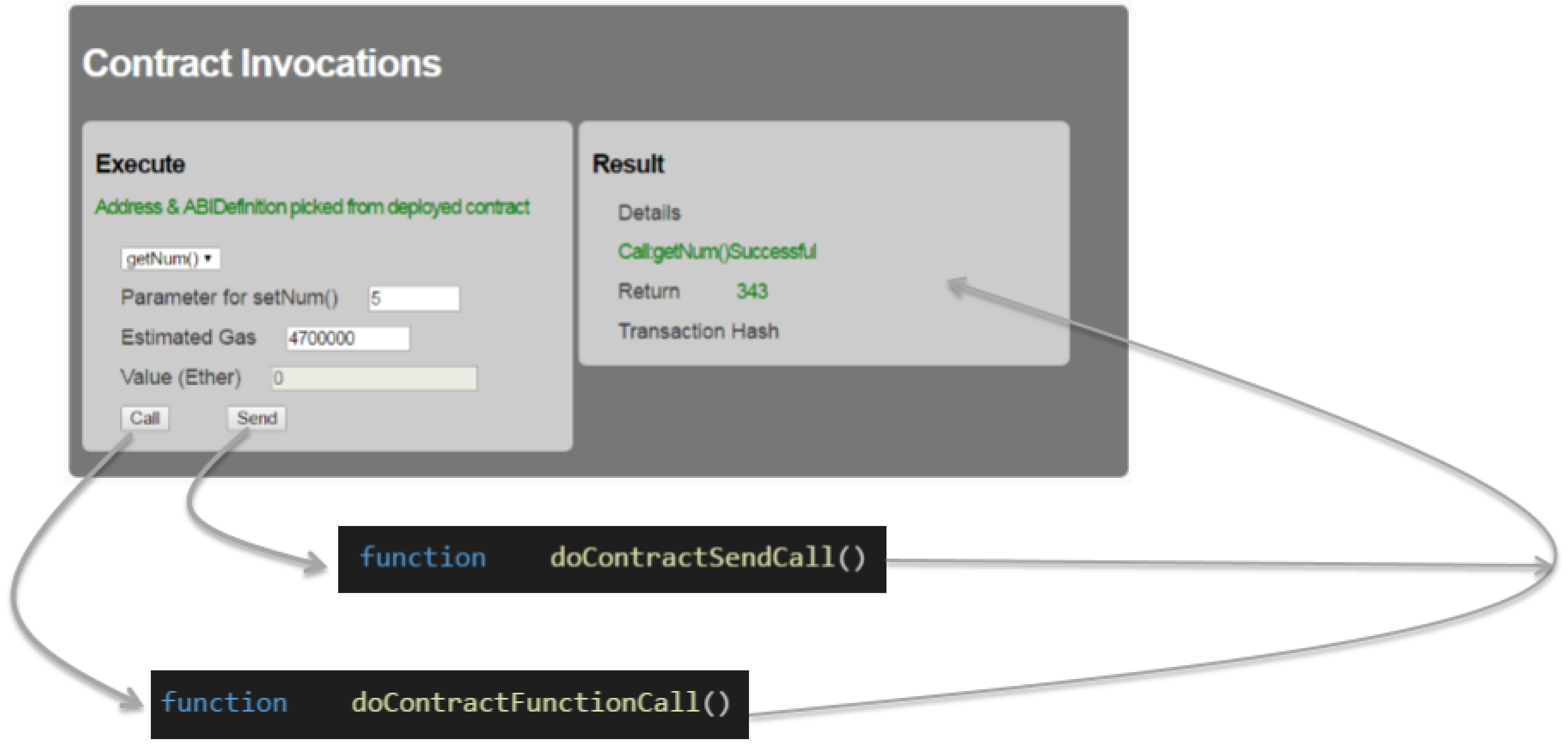
2. sendTransaction(...)

   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

Parameter for setNum() 5

Estimated Gas 4700000

Value (Ether) 0

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 ...)*

```
Transacti
 {
  "from":                    526d2a78ecb268ec7f0a60d1b219",
   .....,
  "data" [redacted]
 }
```

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

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

call()

From: is optional

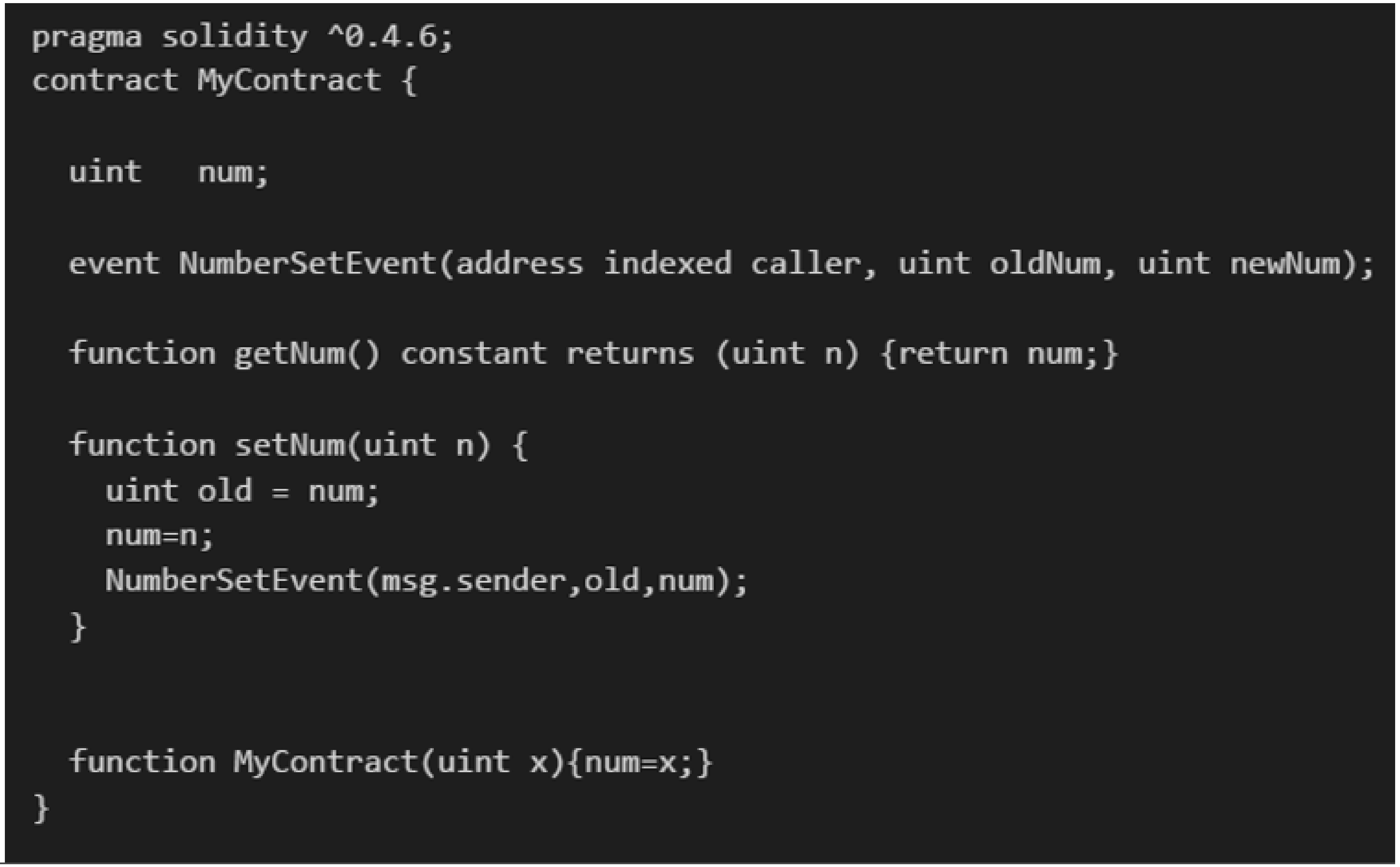
var result = web3.eth.call(transaction_object, [default block], [callback])

"latest " by default

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

Contract Events

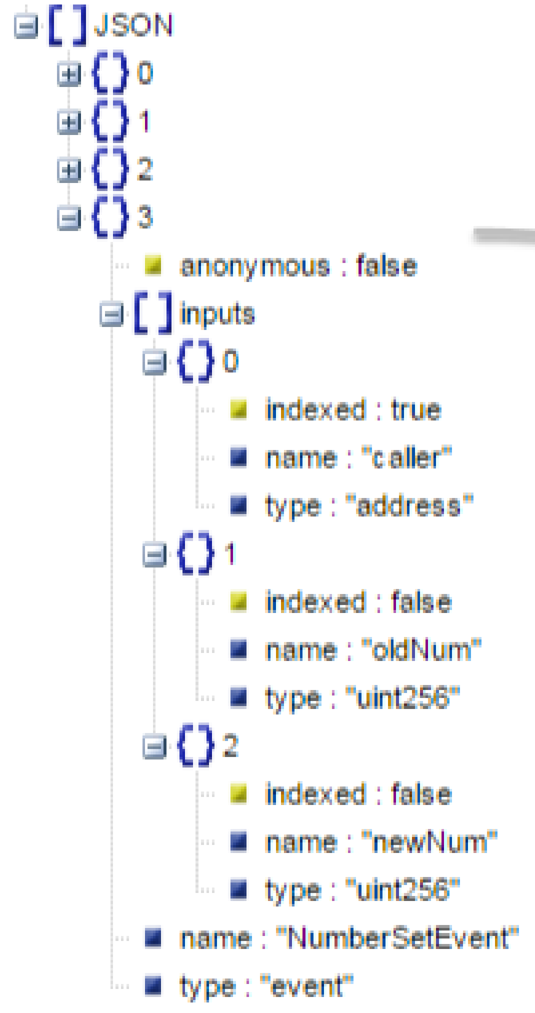
Contracts may emit events



```
pragma solidity ^0.4.6;
contract MyContract {

  uint   num;

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

  function getNum() constant returns (uint n) {return num;}

  function setNum(uint n) {
    uint old = num;
    num=n;
    NumberSetEvent(msg.sender,old,num);
  }


  function MyContract(uint x){num=x;}
}
```

Contract Events

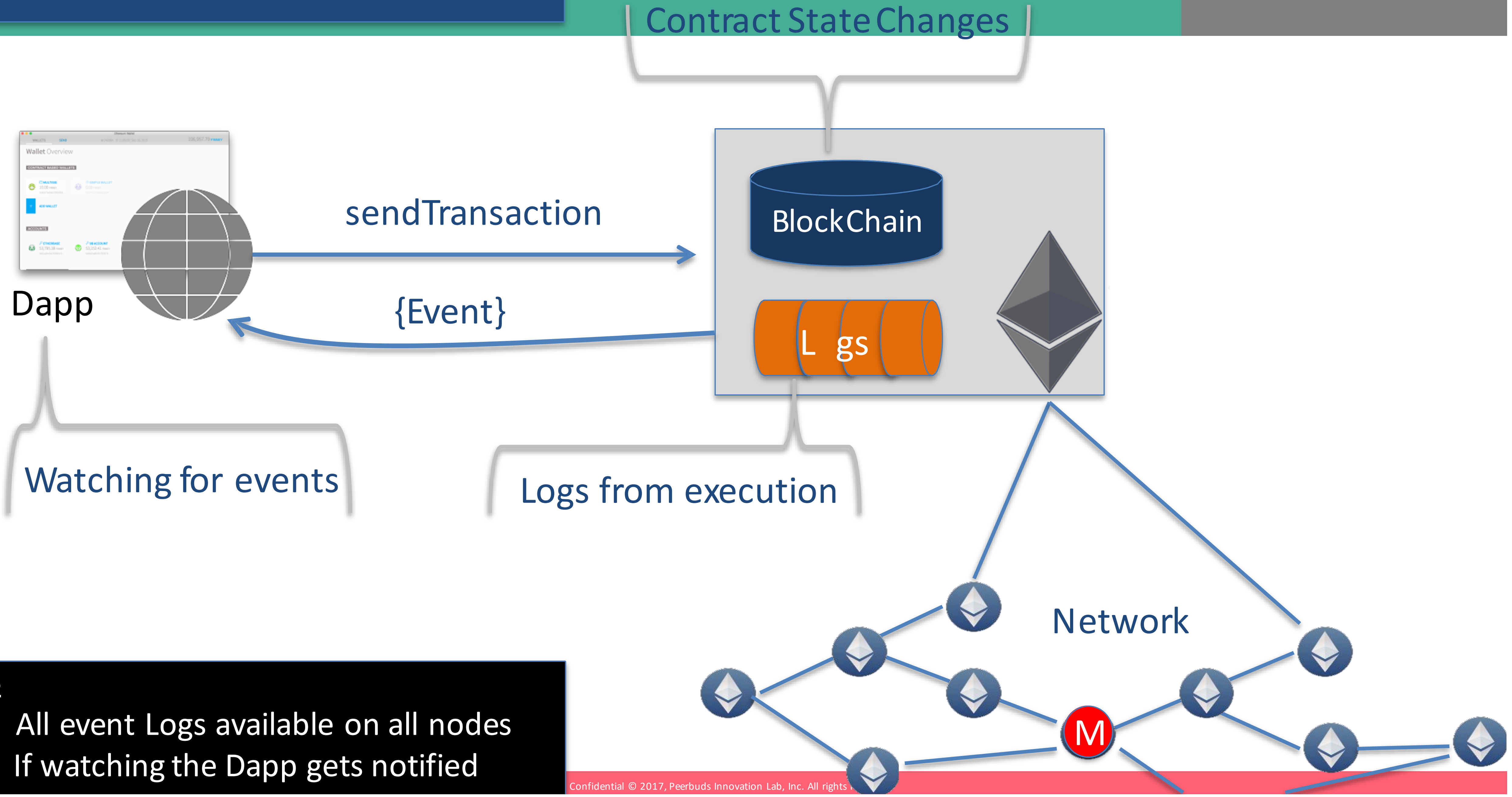
Contracts may emit events



```
[ ] JSON
  { } 0
  { } 1
  { } 2
  { } 3
    anonymous : false
    [ ] inputs
      { } 0
        indexed : true
        name : "caller"
        type : "address"
      { } 1
        indexed : false
        name : "oldNum"
        type : "uint256"
      { } 2
        indexed : false
        name : "newNum"
        type : "uint256"
    name : "NumberSetEvent"
    type : "event"
```

```
pragma solidity ^0.4.6;
contract MyContract {

  uint   num;

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

  function getNum() constant returns (uint n) {return num;}

  function setNum(uint n) {
    uint old = num;
    num=n;
    NumberSetEvent(msg.sender,old,num);
  }


  function MyContract(uint x){num=x;}
}
```

Ethereum Logs & Events



PS

- All event Logs available on all nodes
- If watching the Dapp gets notified

Web3 JS API:

- Logs
- Events

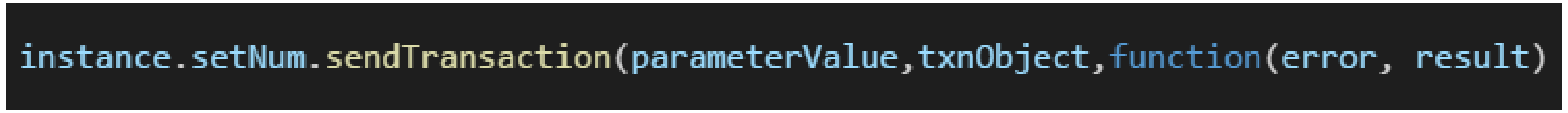
Event/Log usage patterns

1. Receive data for transaction

2. Asynchronous trigger

3. Cheap data storage

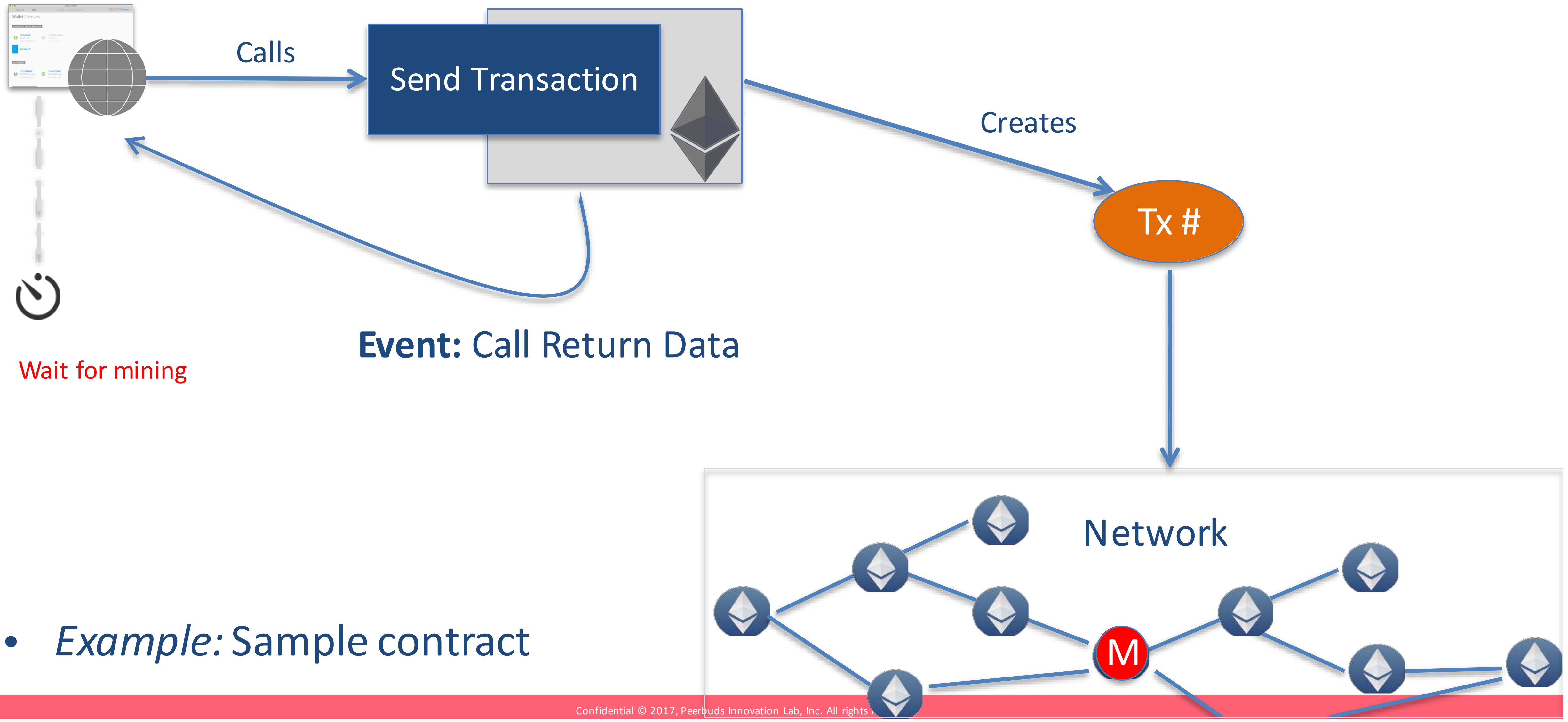
1. Receives data for transaction



```
instance.setNum.sendTransaction(parameterValue,txnObject,function(error, result)
```

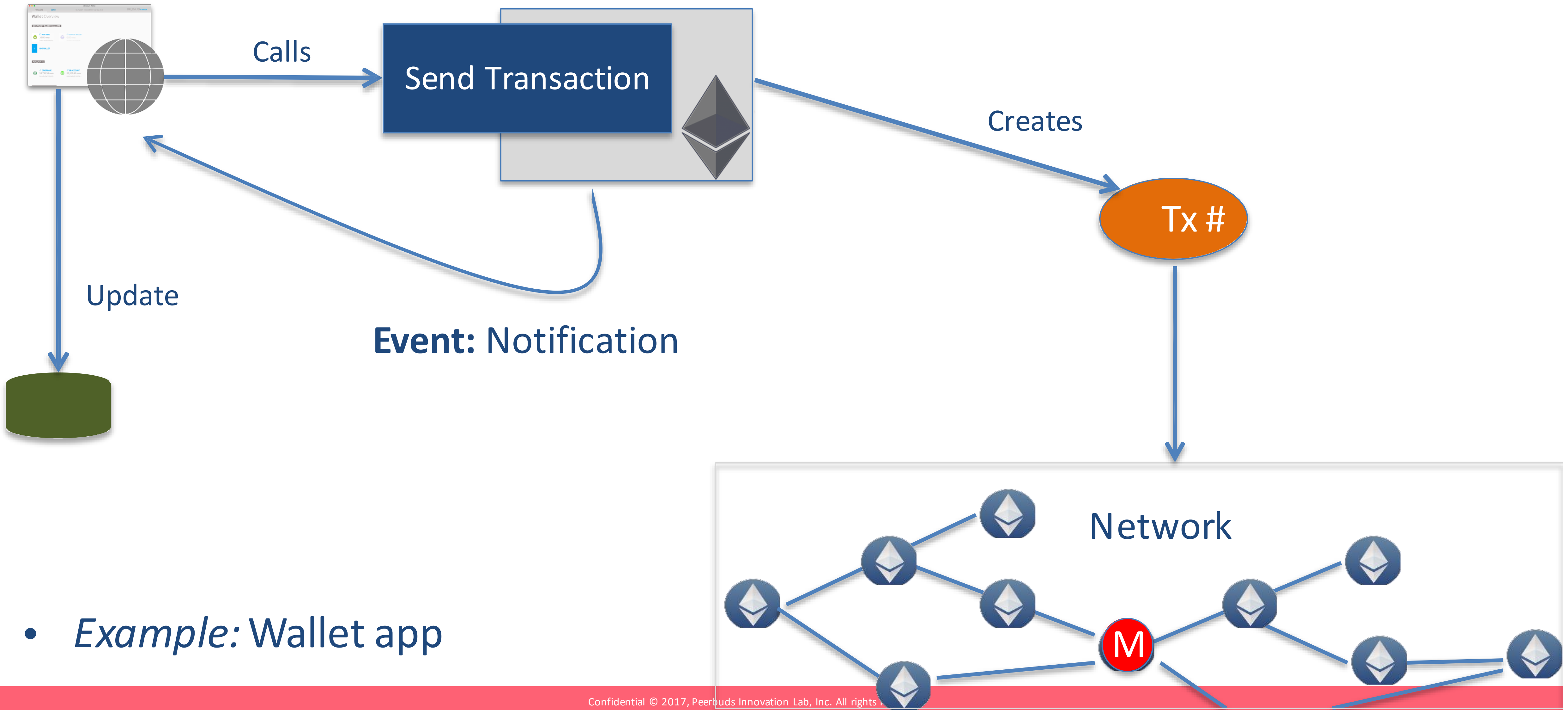
- Call returns a transaction hash and not a return value
  - Method execution result is not available till transaction is mined
  - Contract (methods) may return data using **events**

1. Receives data for transaction



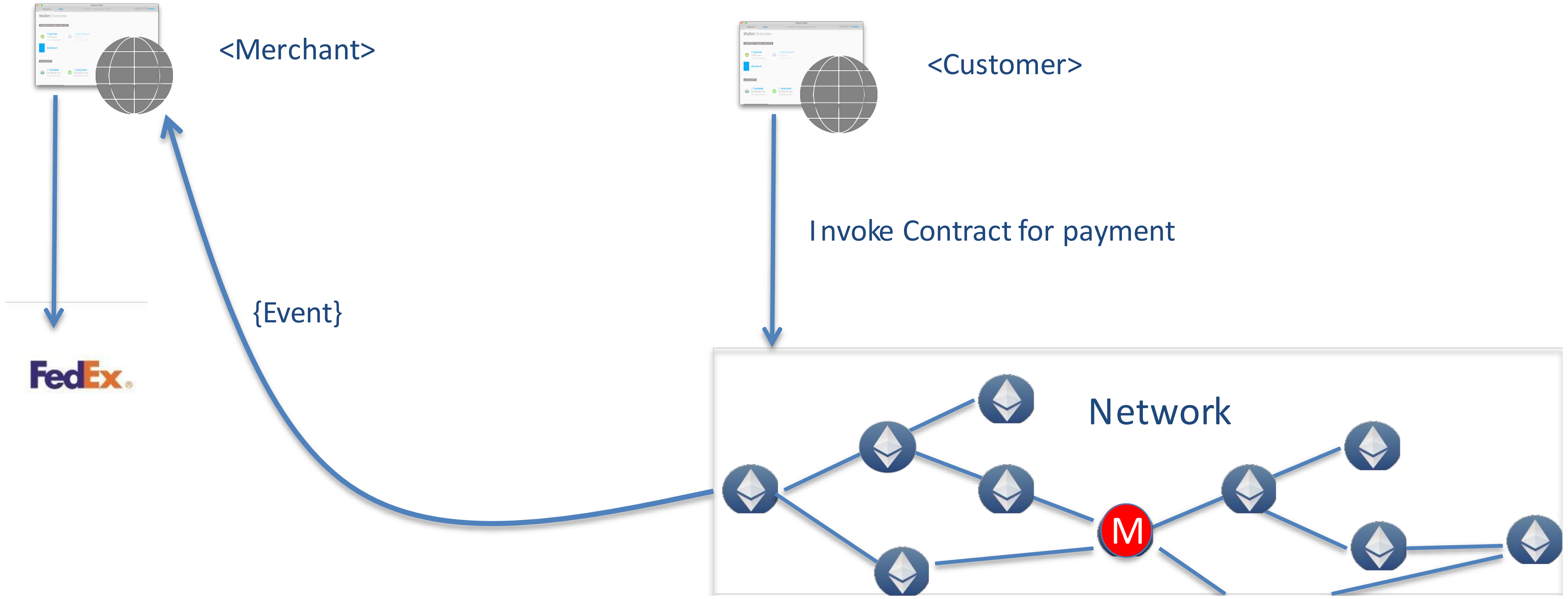
- *Example:* Sample contract

2. Asynchronous Notifications



- *Example:* Wallet app

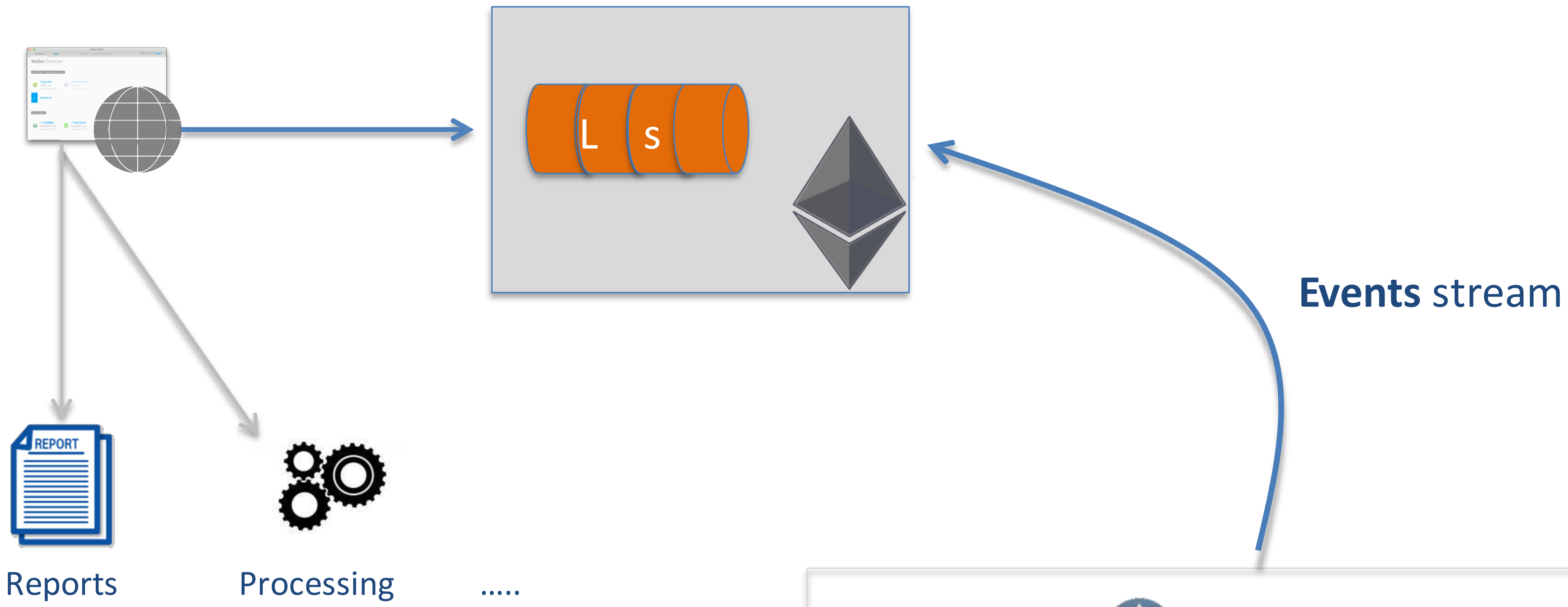
2. Asynchronous Notifications

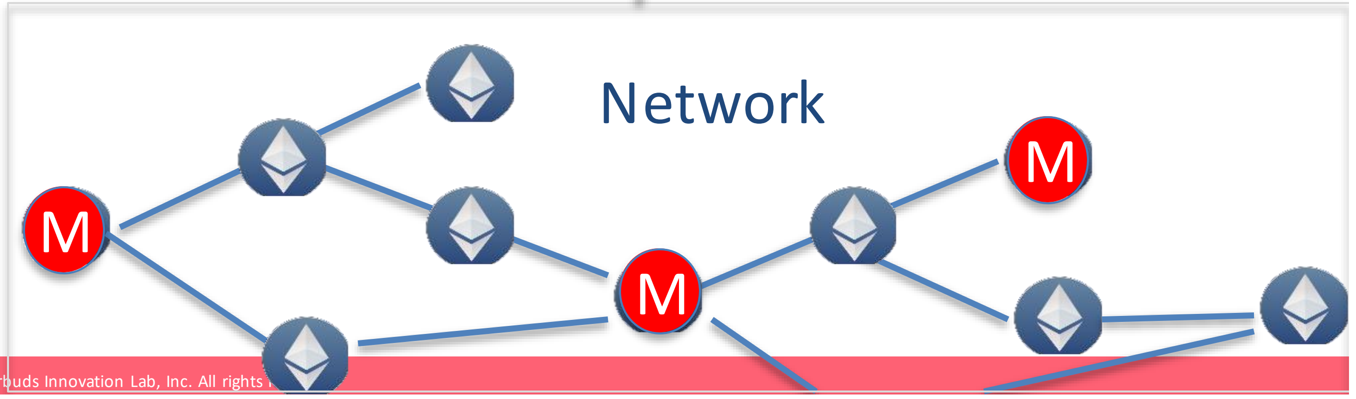


2. Asynchronous processing

- Front end (Dapp) can **watch** for events of interest
  - Example: Wallet app receives notification on receiving ethers
  - Example: Multisig contract shows transactions waiting for approval

3. Data storage





- *Example:*
  Encrypted customer identity

3. Data storage

- Cheaper than contract storage

• Log data storage cost	8 Gas/byte
• Contract data storage cost	20,000 Gas/32-byte

- Logs are **NOT** accessible from contracts

Watch & Get

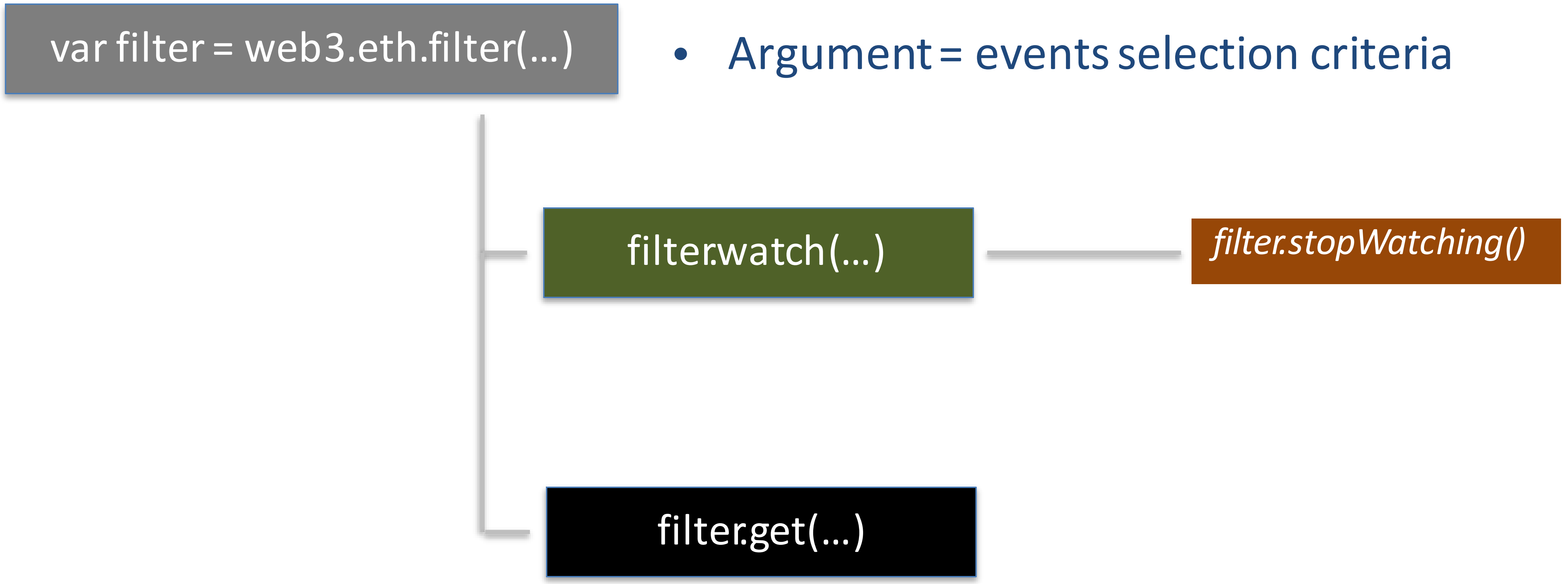
- Watch
  - Listens for incoming events
- Get
  - Gets the log data

2 ways to watch & get

1. Using the Filter API
2. 2. Using the contract instance

Using Filter

- Argument = events selection criteria



```
var filter = web3.eth.filter(...)
filter.watch(...)
filter.stopWatching()
filter.get(...)
```

web3.eth.filter(...)

1. *web3.eth.filter*(string)

"latest"

Result=Block Hash of latest Block

"pending"

Result=Transaction Hash of latest txn

2. *web3.eth.filter*(options_object)

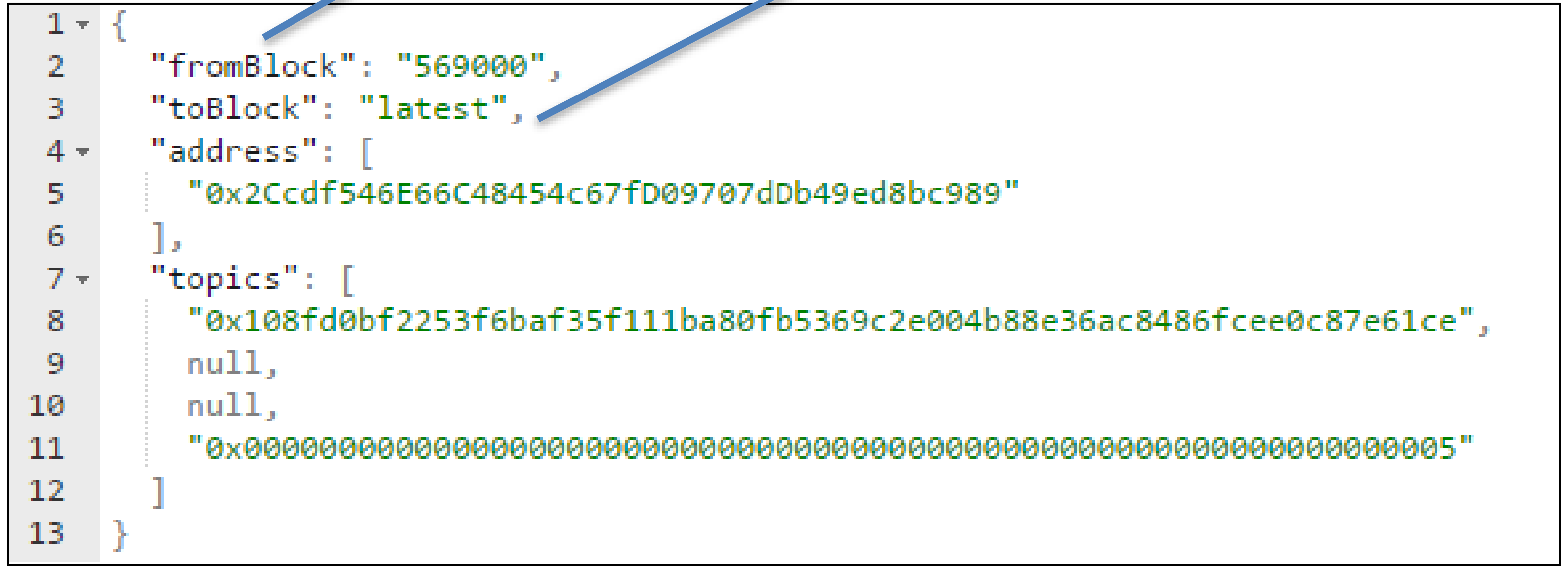
Options_object

- Block range *fromBlock, toBlock*
- Specific contract instance *[address]*
- Event data
  - Data in the log fields topic: *[‘event-signature’, ‘data1’, ‘data2’, ‘data3’]*
    - Fields marked *indexed* used in topics
    - Maximum of 3 indexed fields & order is important

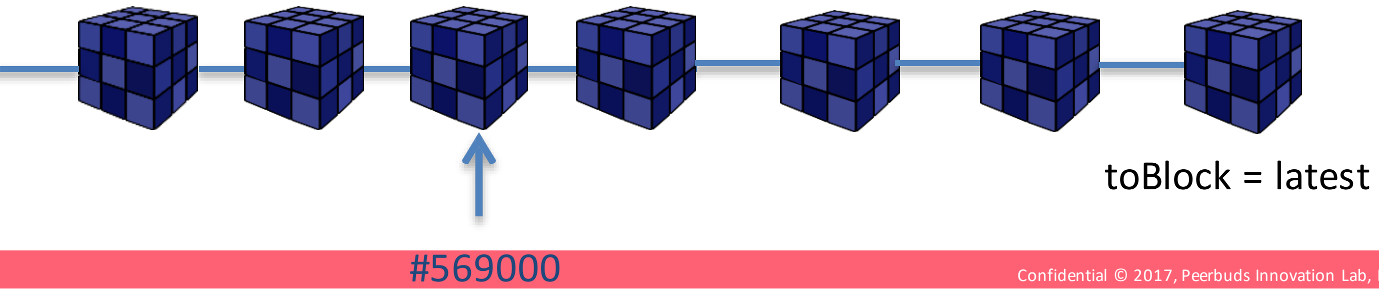
Options JSON

Get events starting from block# 569000

- For get() ; get events from 569000 to the current block
- For watch() continue to receive events for all blocks

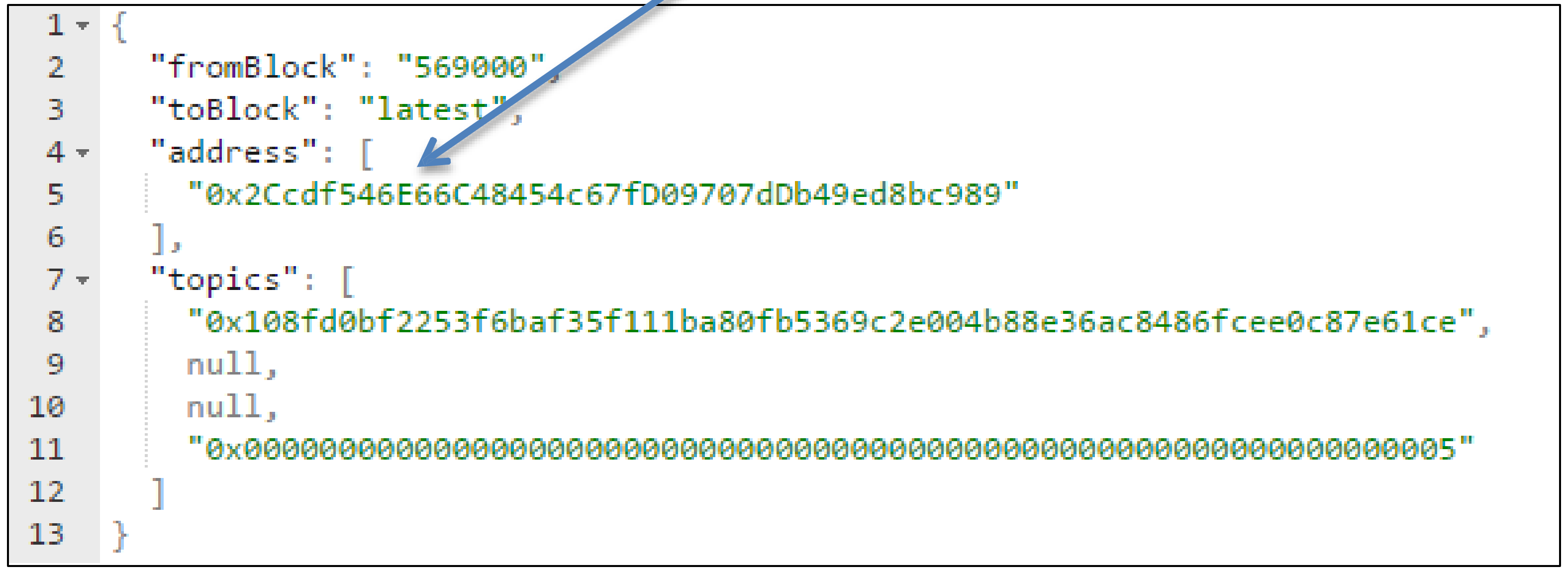


```
{
  "fromBlock": "569000",
  "toBlock": "latest",
  "address": [
    "0x2Ccdf546E66C48454c67fD09707dDb49ed8bc989"
  ],
  "topics": [
    "0x108fd0bf2253f6baf35f111ba80fb5369c2e004b88e36ac8486fcee0c87e61ce",
    null,
    null,
    "0x0000000000000000000000000000000000000000000000000000000000000005"
  ]
}
```



Options JSON

Array of contract addresses

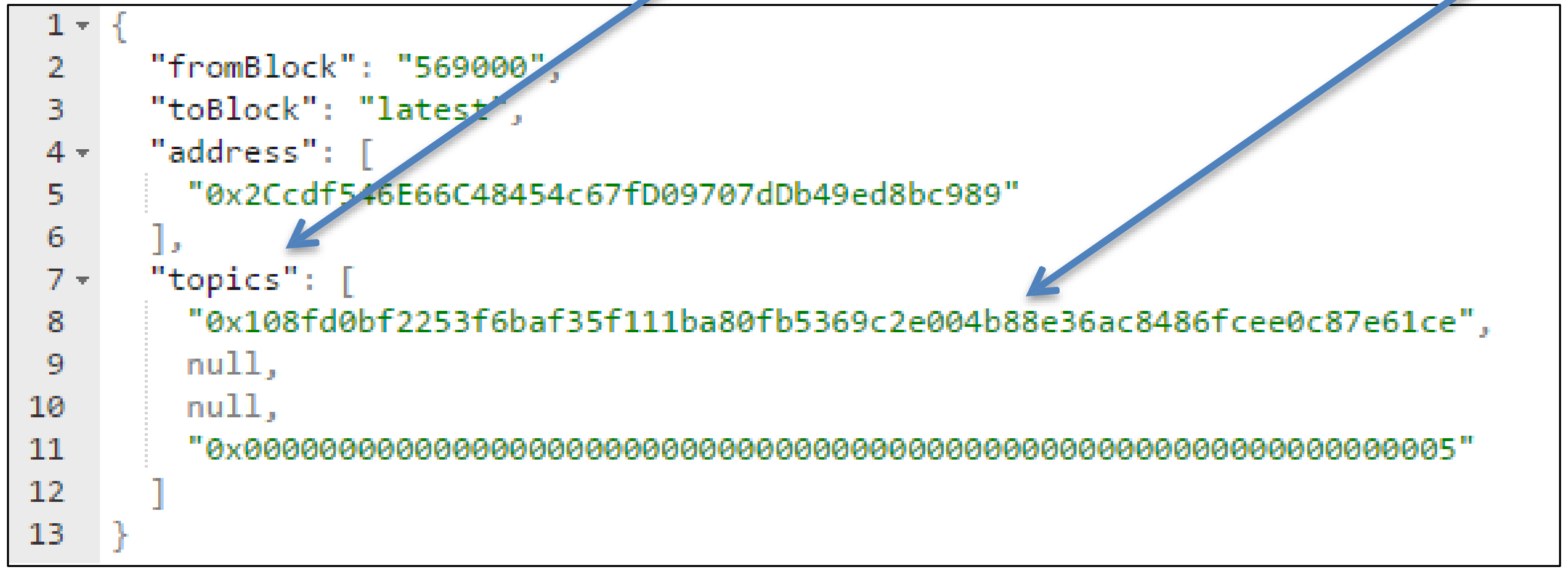


```
{
  "fromBlock": "569000",
  "toBlock": "latest",
  "address": [
    "0x2Ccdf546E66C48454c67fD09707dDb49ed8bc989"
  ],
  "topics": [
    "0x108fd0bf2253f6baf35f111ba80fb5369c2e004b88e36ac8486fcee0c87e61ce",
    null,
    null,
    "0x0000000000000000000000000000000000000000000000000000000000000005"
  ]
}
```

Options JSON

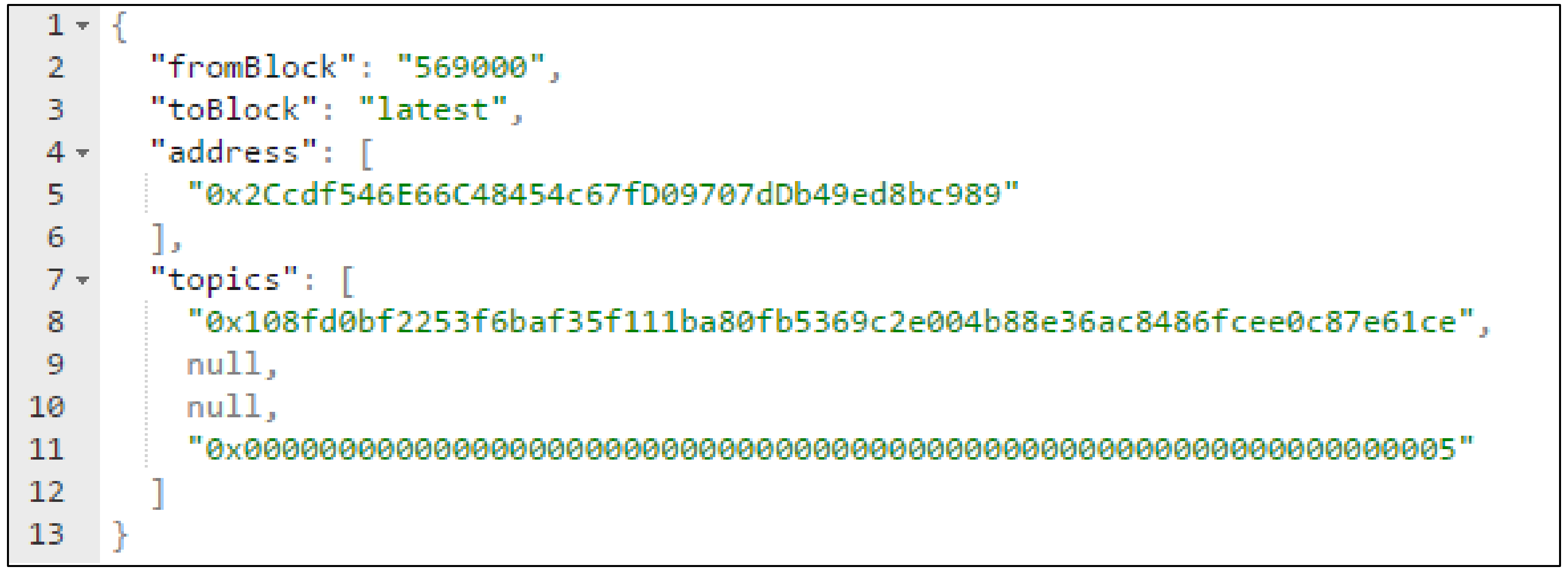
topics = event data criteria

topics[0] = Event Signature



```json
{
  "fromBlock": "569000",
  "toBlock": "latest",
  "address": [
    "0x2Ccdf546E66C48454c67fD09707dDb49ed8bc989"
  ],
  "topics": [
    "0x108fd0bf2253f6baf35f111ba80fb5369c2e004b88e36ac8486fcee0c87e61ce",
    null,
    null,
    "0x0000000000000000000000000000000000000000000000000000000000000005"
  ]
}
```

Options JSON



```json
{
  "fromBlock": "569000",
  "toBlock": "latest",
  "address": [
    "0x2Ccdf546E66C48454c67fD09707dDb49ed8bc989"
  ],
  "topics": [
    "0x108fd0bf2253f6baf35f111ba80fb5369c2e004b88e36ac8486fcee0c87e61ce",
    null,
    null,
    "0x0000000000000000000000000000000000000000000000000000000000000005"
  ]
}
```

event NumberSetEvent(address <u>indexed</u> **caller**, bytes32 <u>indexed</u> **oldNum**, bytes32 <u>indexed</u> **newNum**);

setNum(**5**) **{Event Received}**

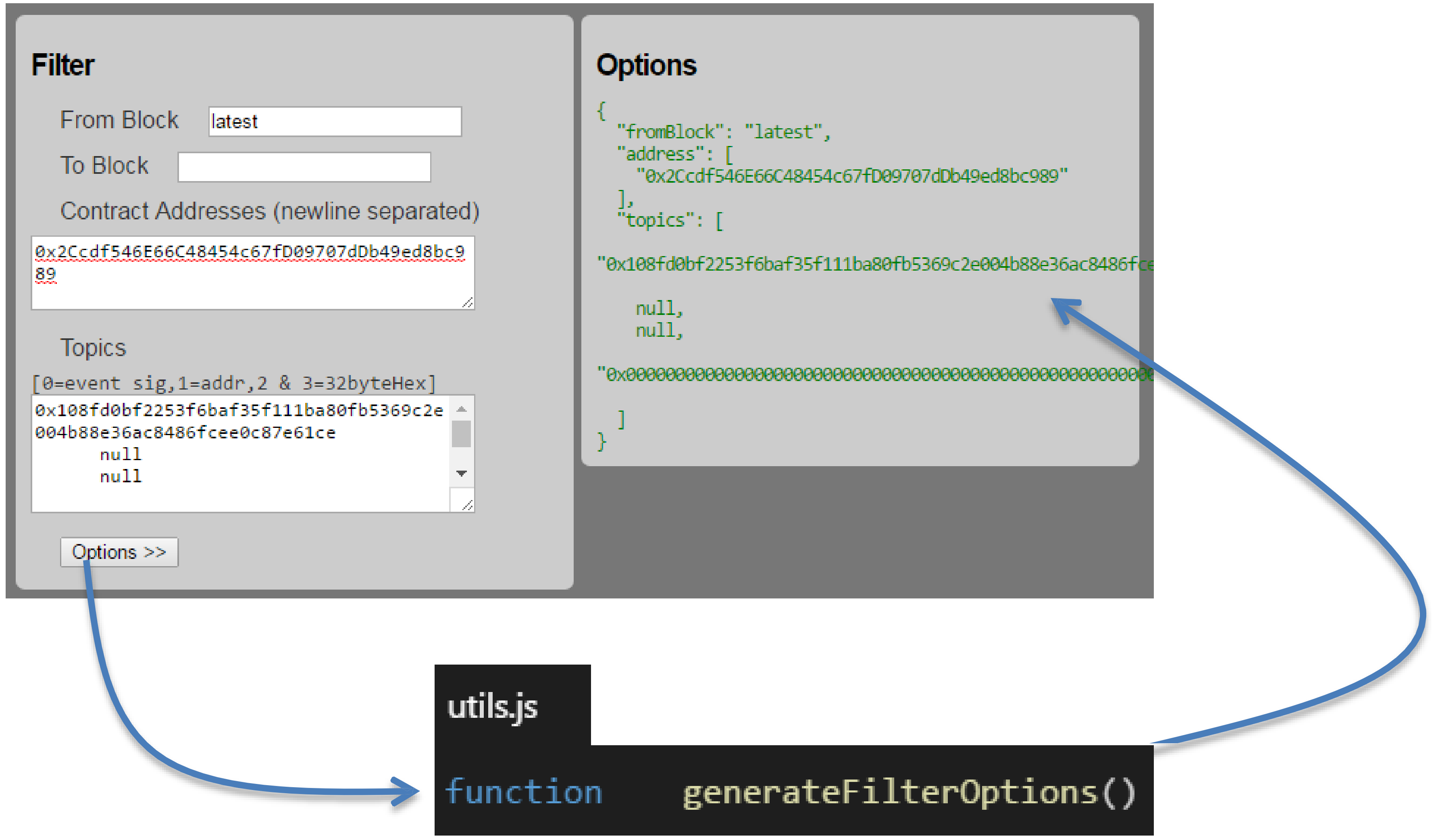
setNum(**6**) **{NO Event Received}**

watch() & get()

- filter.**get**(callback_func)
  - Result : Array of events

- filter.**watch**(callback_func)
  - Result : event data

Walkthrough



Filter

From Block: latest

To Block:

Contract Addresses (newline separated)

```
0x2Ccdf546E66C48454c67fD09707dDb49ed8bc9
89
```

Topics

```
[0=event sig,1=addr,2 & 3=32byteHex]
0x108fd0bf2253f6baf35f111ba80fb5369c2e
004b88e36ac8486fcee0c87e61ce
        null
        null
```

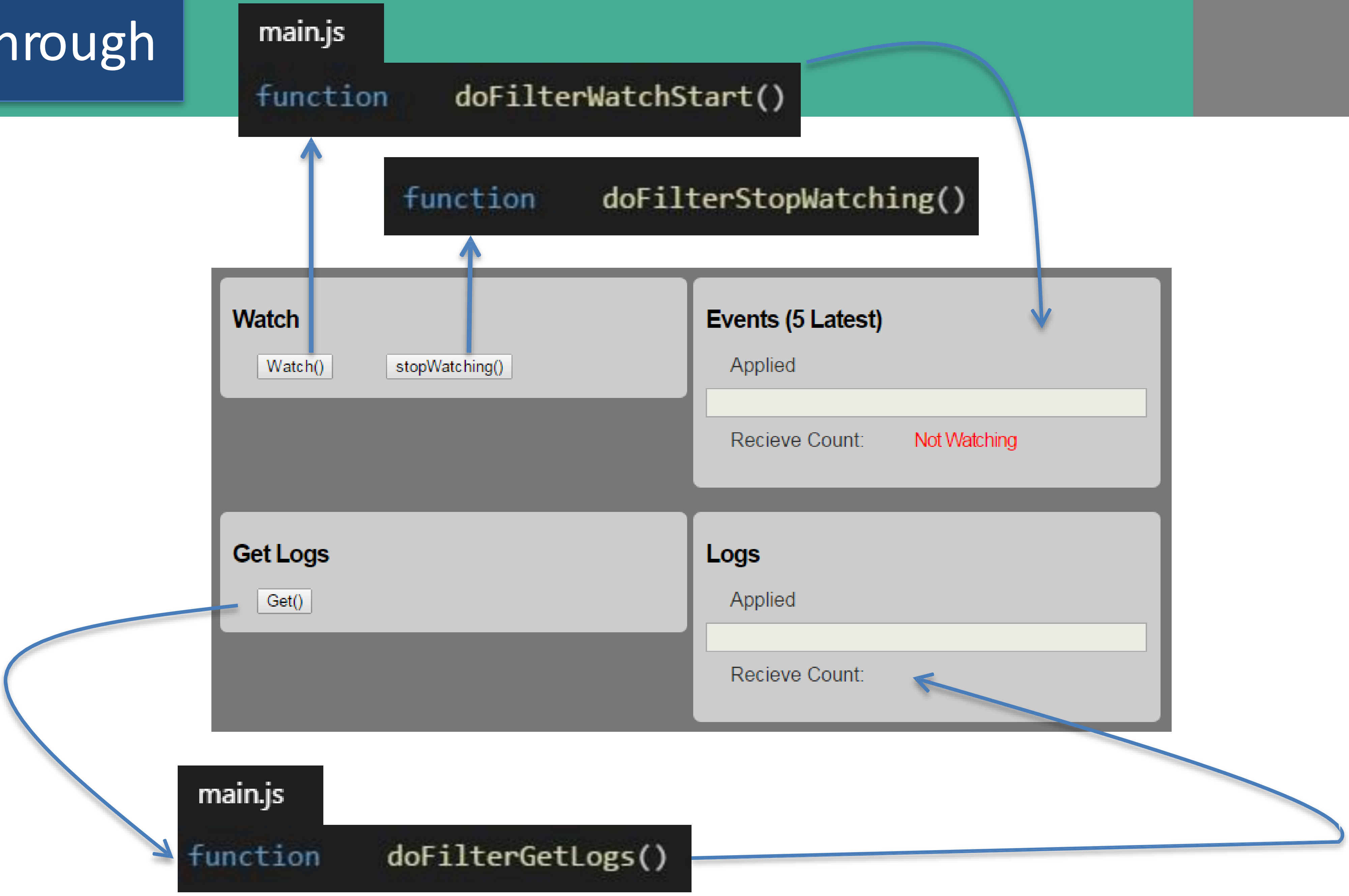
Options

```
{
  "fromBlock": "latest",
  "address": [
    "0x2Ccdf546E66C48454c67fD09707dDb49ed8bc989"
  ],
  "topics": [

"0x108fd0bf2253f6baf35f111ba80fb5369c2e004b88e36ac8486fc

    null,
    null,

"0x000000000000000000000000000000000000000000000000000000

  ]
}
```

utils.js

```
function     generateFilterOptions()
```

Walkthrough



main.js

```
function doFilterWatchStart()
```

```
function doFilterStopWatching()
```

Watch

Watch() stopWatching()

Events (5 Latest)

Applied

Recieve Count: Not Watching

Get Logs

Get()

Logs

Applied

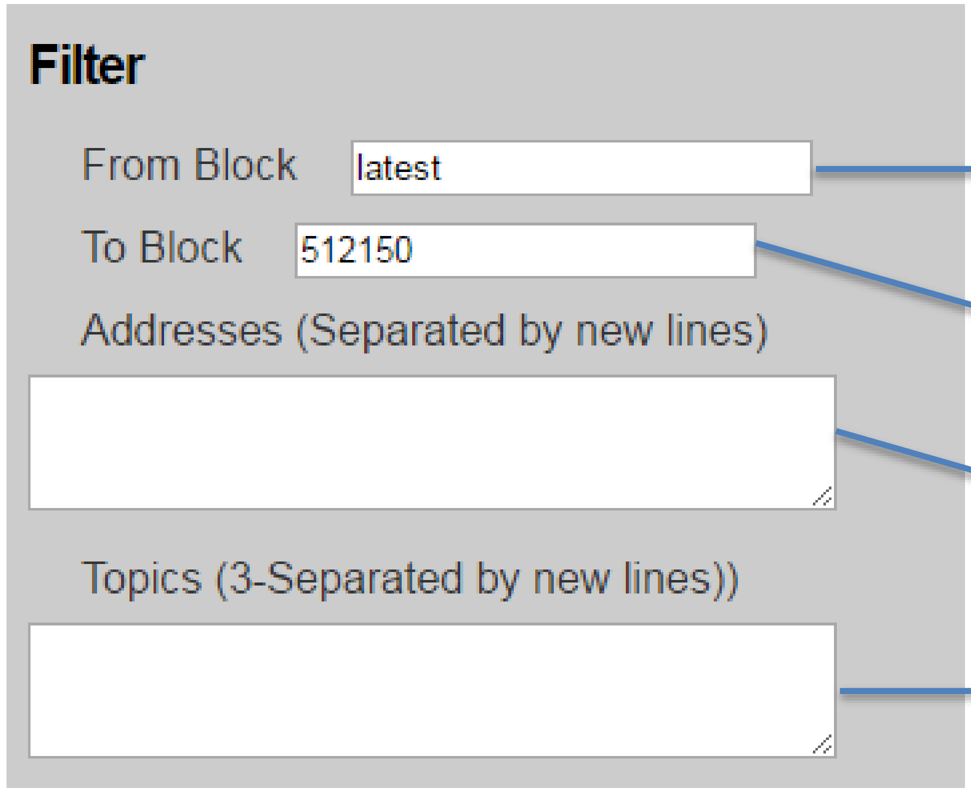
Recieve Count:

main.js

```
function doFilterGetLogs()
```

2. web3.eth.filter(options)

web3.eth.filter(options) for events



Filter

From Block: latest

To Block: 512150

Addresses (Separated by new lines)

Topics (3-Separated by new lines))

fromBlock: by default 'latest' ; number or hash

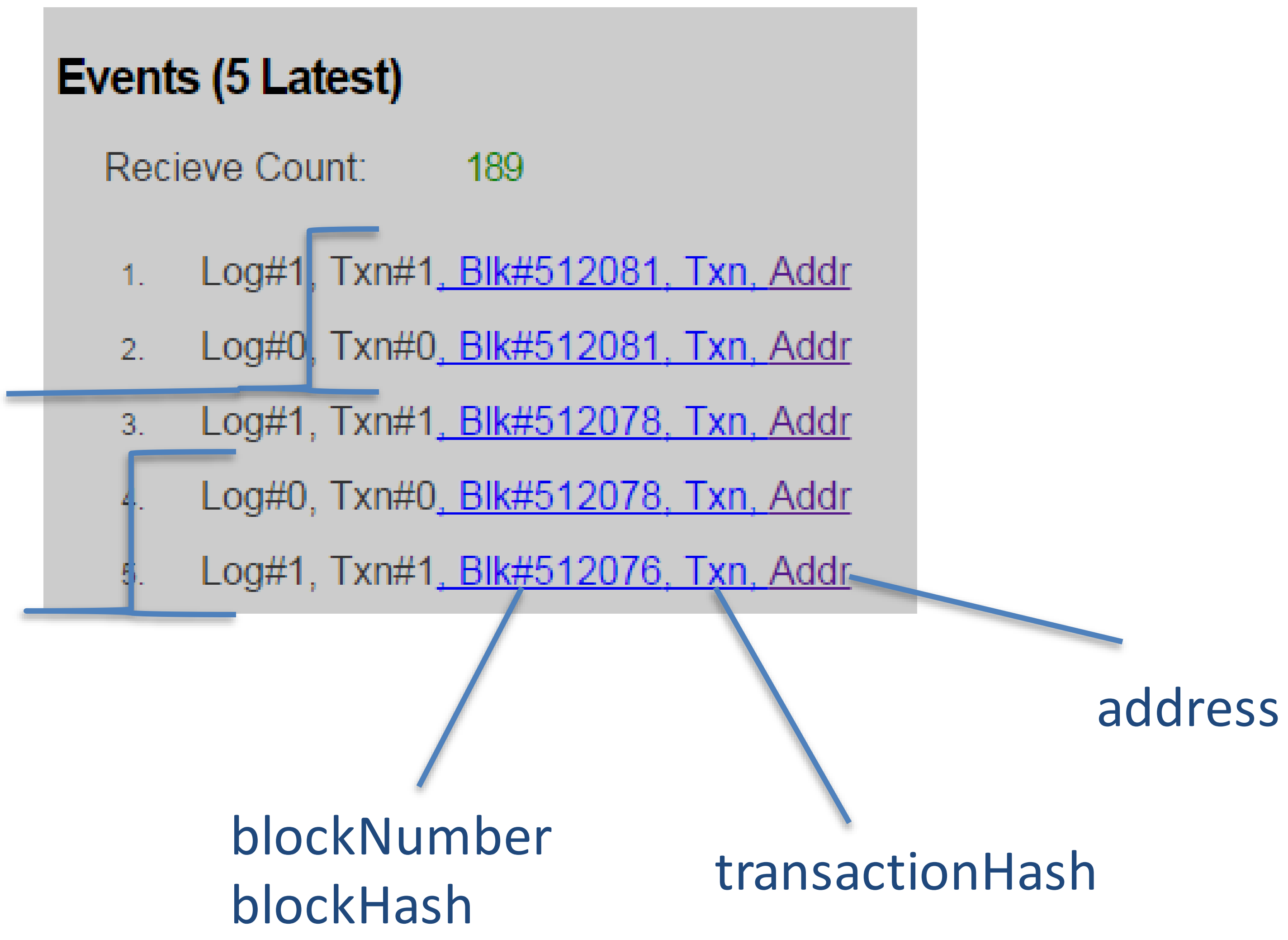
toBlock: leave blank for continuous watching

address: Contract Address(es)

topic: Indexed topic data

Event Data

Events received in real time



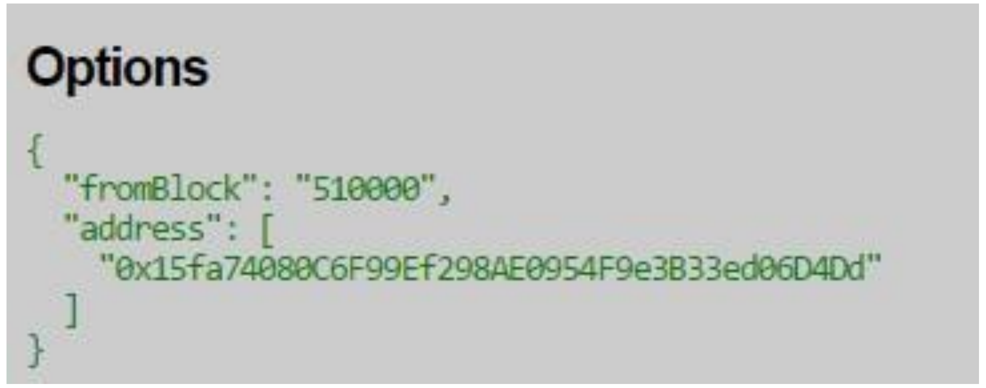
Events (5 Latest)

Recieve Count: 189

1. Log#1, Txn#1, Blk#512081, Txn, Addr
2. Log#0, Txn#0, Blk#512081, Txn, Addr
3. Log#1, Txn#1, Blk#512078, Txn, Addr
4. Log#0, Txn#0, Blk#512078, Txn, Addr
5. Log#1, Txn#1, Blk#512076, Txn, Addr

Get Logs

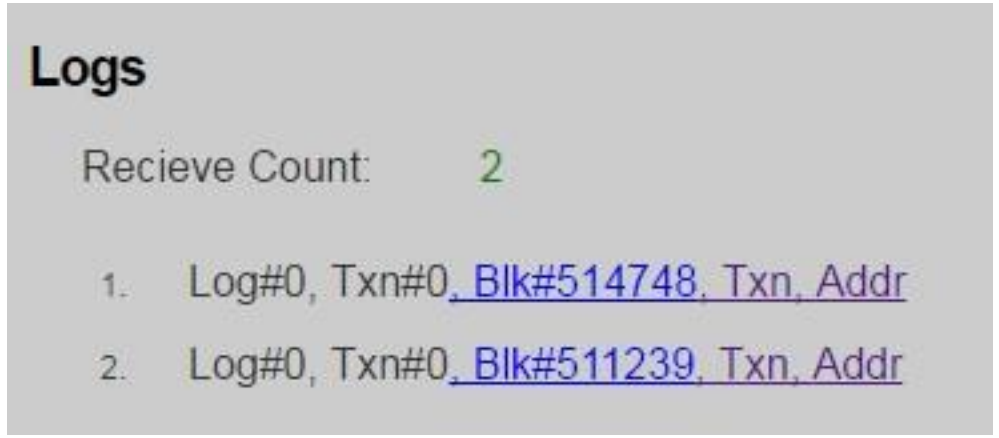
web3.eth.filter(options) for events



Options

```
{
  "fromBlock": "510000",
  "address": [
    "0x15fa74080C6F99Ef298AE0954F9e3B33ed06D4Dd"
  ]
}
```

Array of logs



Logs

```
Recieve Count:      2

1.  Log#0, Txn#0, Blk#514748, Txn, Addr
2.  Log#0, Txn#0, Blk#511239, Txn, Addr
```

filter.watch

- Watch for events => installs the filter on node
  - *watch*() callback receives events based on the filter
  - *stopWatching*() for events; removes the filter on node

- Read the past logs
  - *get*()

Contract Object

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

1. Deploying the contract code to EVM

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

2. Invoking a contract function

3. Watch for events & Get events data from *Log*

Contract Events

Like methods, events are part of *abiDefinition*

9[]JSON
•Cl•
•Cl•
a(),
9()3

- IIIOl\yfflOUS false

{3 [ ]'np.,ts

s{}o

- iw:leJ<ed true
- name "clller"
- type "address•

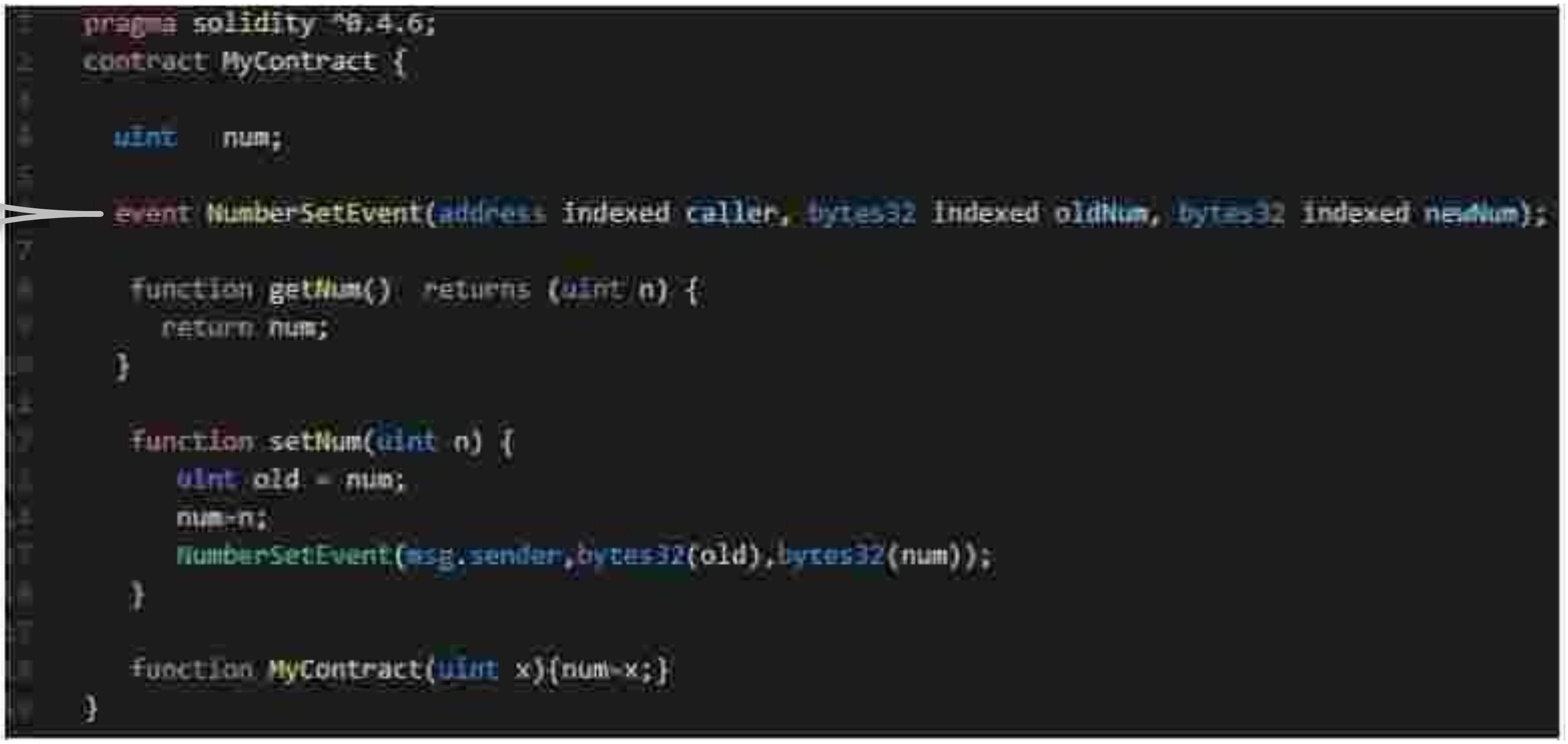
a{}•

- rdexed lalse
- name "ot!Nurn"
- type 'wlt256"

a{},

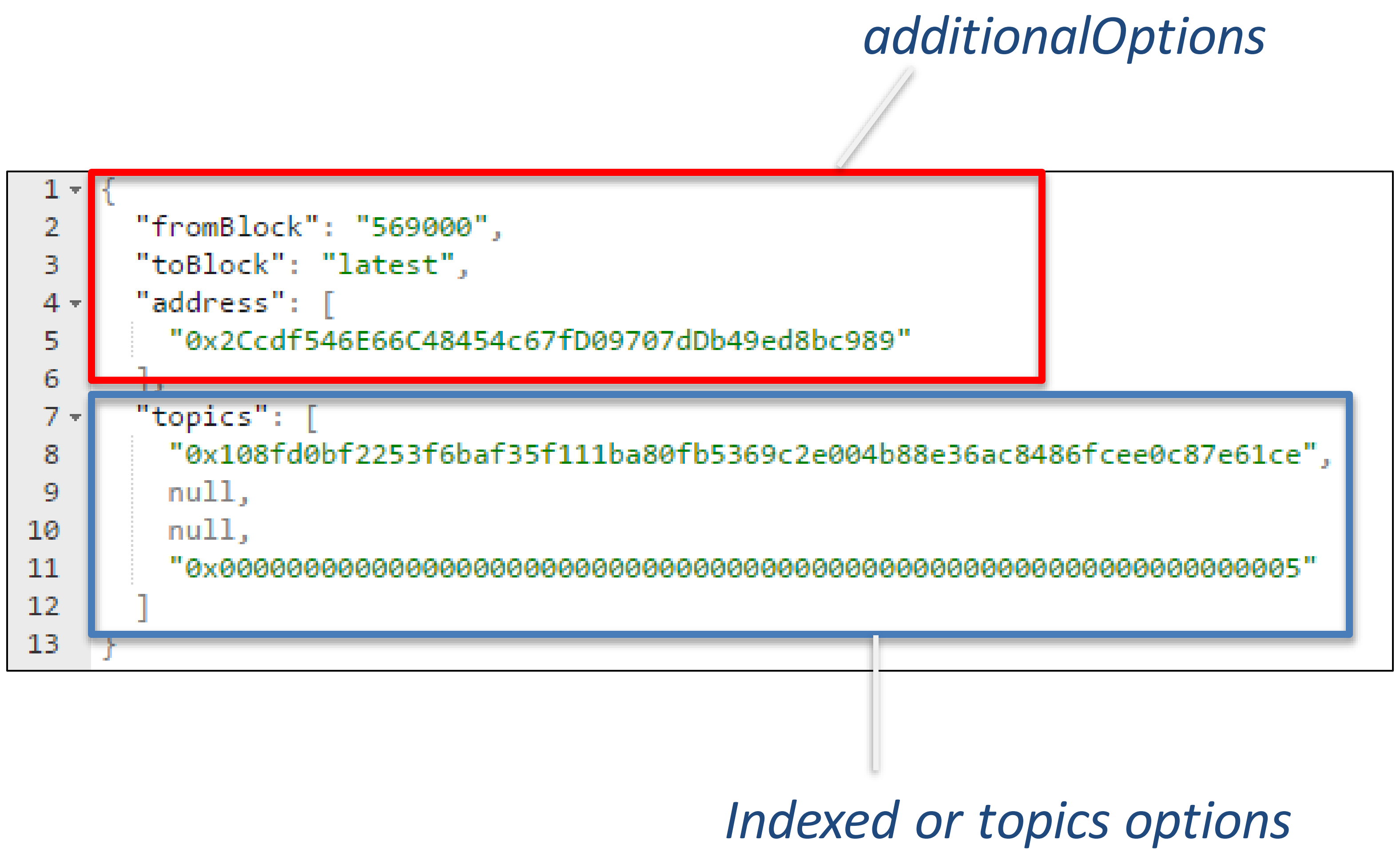
- rdexed raise
- name "newl'lurn"
- type-"uS't2S6"

- name "NumberSetf.�en !"
- IY1)e "eoc.� ·



```
pragma solidity ^0.4.6;
contract MyContract {

  uint   num;

  event NumberSetEvent(address indexed caller, bytes32 indexed oldNum, bytes32 indexed newNum);

   function getNum()  returns (uint n) {
     return num;
   }

   function setNum(uint n) {
      uint old = num;
      num=n;
      NumberSetEvent(msg.sender,bytes32(old),bytes32(num));
   }

   function MyContract(uint x){num=x;}
}
```

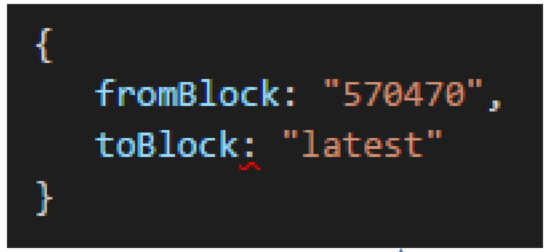
Event Filtering



```
{
  "fromBlock": "569000",
  "toBlock": "latest",
  "address": [
    "0x2Ccdf546E66C48454c67fD09707dDb49ed8bc989"
  ],
  "topics": [
    "0x108fd0bf2253f6baf35f111ba80fb5369c2e004b88e36ac8486fcee0c87e61ce",
    null,
    null,
    "0x0000000000000000000000000000000000000000000000000000000000000005"
  ]
}
```

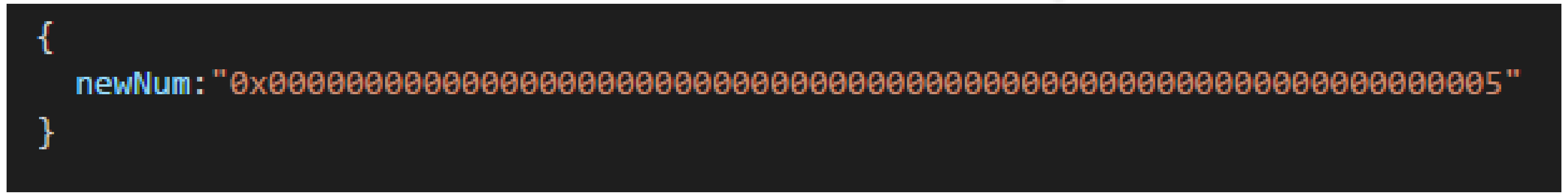
Contract Event

var contractEvent =
    contractInstance.**allEvent**(*additionalOptions*)



```
{
    fromBlock: "570470",
    toBlock: "latest"
}
```

var contractEvent =
    contractInstance.**NumberSetEvent**(*indexedOptions, additionalOptions*)



```
{
  newNum:"0x0000000000000000000000000000000000000000000000000000000000000005"
}
```

get(), watch(), stopWatching()

1. contractEvent .**get**(callback_function)
   - Result : Array of events

2. contractEvent .**watch**(callback_function)
   - Result : Event data

3. contractEvent .**stopWatching**()

Filter : get/watch

- All events from any source
- May be used for writing tools etc
- Indexed data in options/topics array

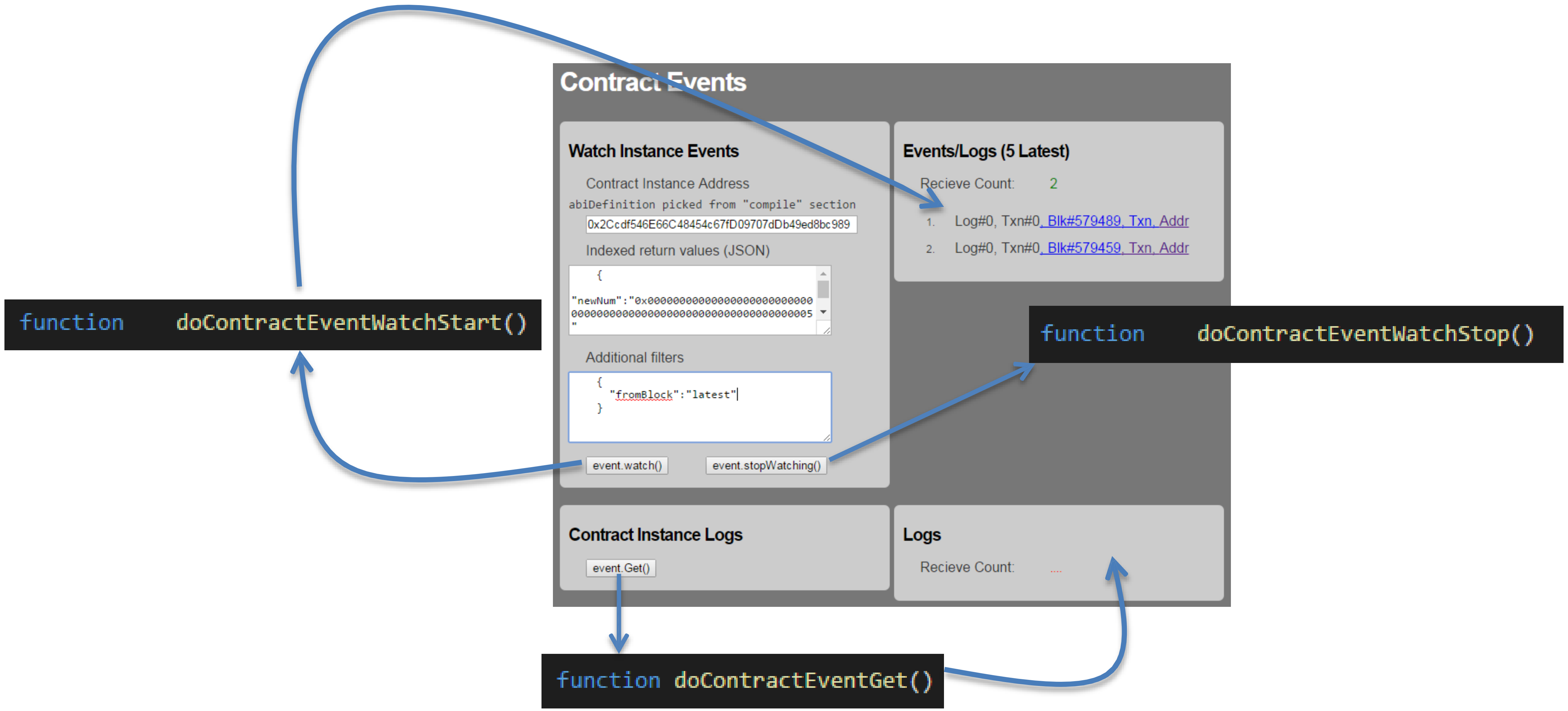
Event : get/watch

- Events from specific contract instance
- For Dapp only
- Indexed/Topic data is a JSON object

Web3 JS API:

- DAPP Infrastructure

get(), watch(), stopWatching()



Contract Events

Watch Instance Events

Contract Instance Address

abiDefinition picked from "compile" section

0x2Ccdf546E66C48454c67fD09707dDb49ed8bc989

Indexed return values (JSON)

```
{
"newNum":"0x0000000000000000000000000000000000000000000000000000000000000005
"
```

Additional filters

```
{
  "fromBlock":"latest"
}
```

event.watch() event.stopWatching()

Events/Logs (5 Latest)

Recieve Count: 2

1. Log#0, Txn#0, Blk#579489, Txn, Addr
2. Log#0, Txn#0, Blk#579459, Txn, Addr

Contract Instance Logs

event.Get()

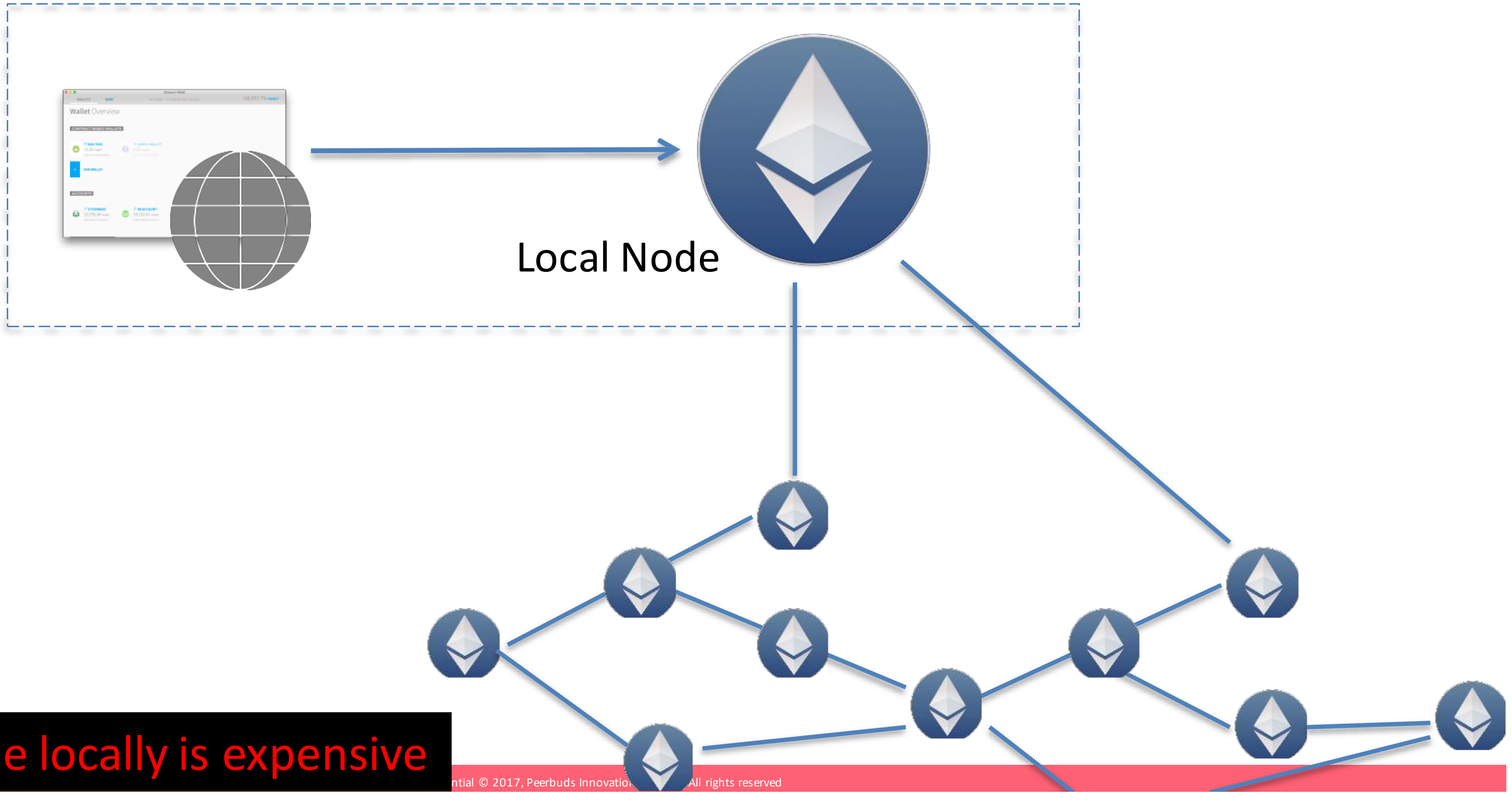
Logs

Recieve Count: ....

```
function doContractEventWatchStart()
function doContractEventWatchStop()
function doContractEventGet()
```

DAPP Infrastructure

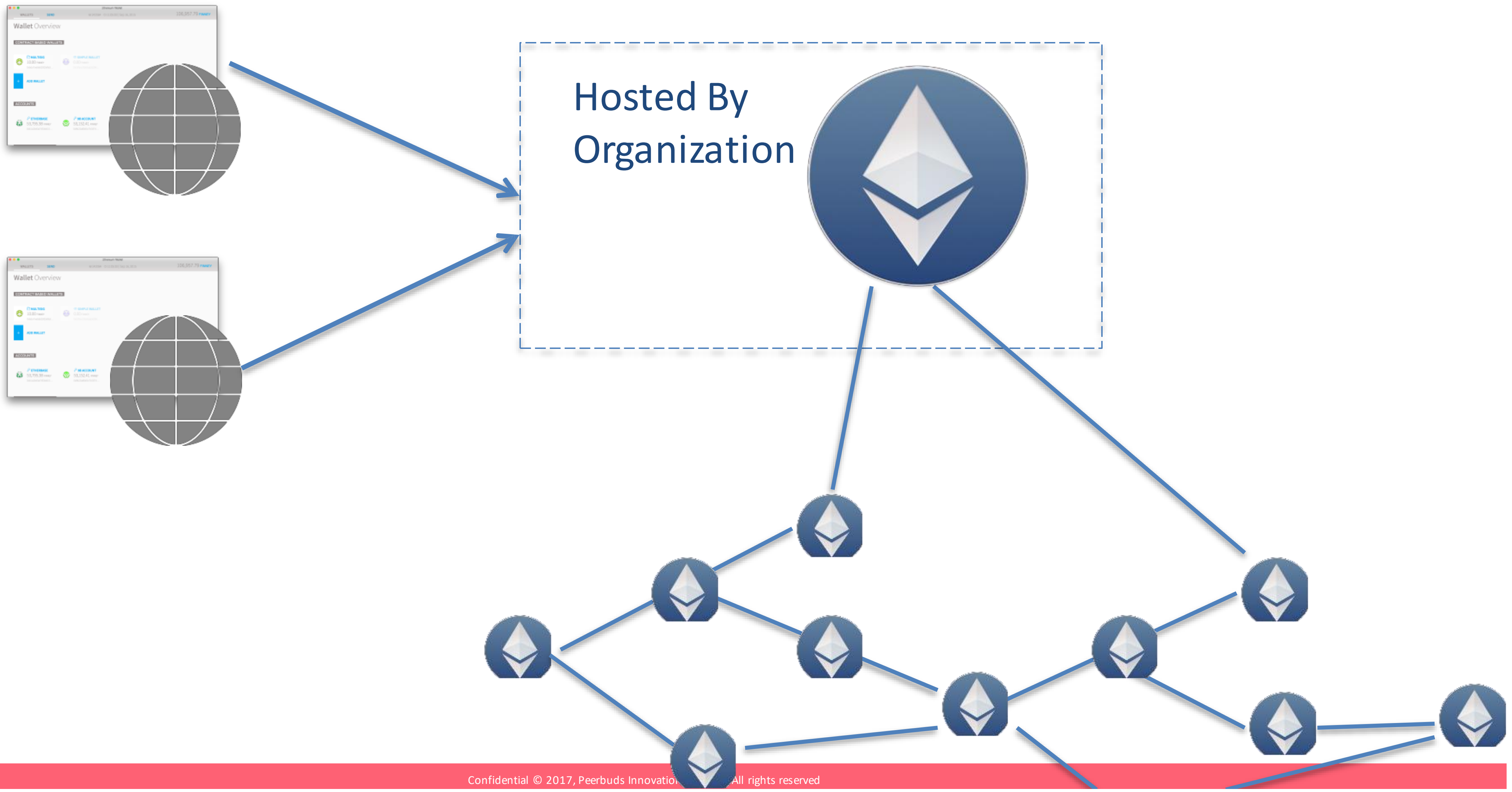
Option#1 DAPP co-located with the client



Deploying node locally is expensive

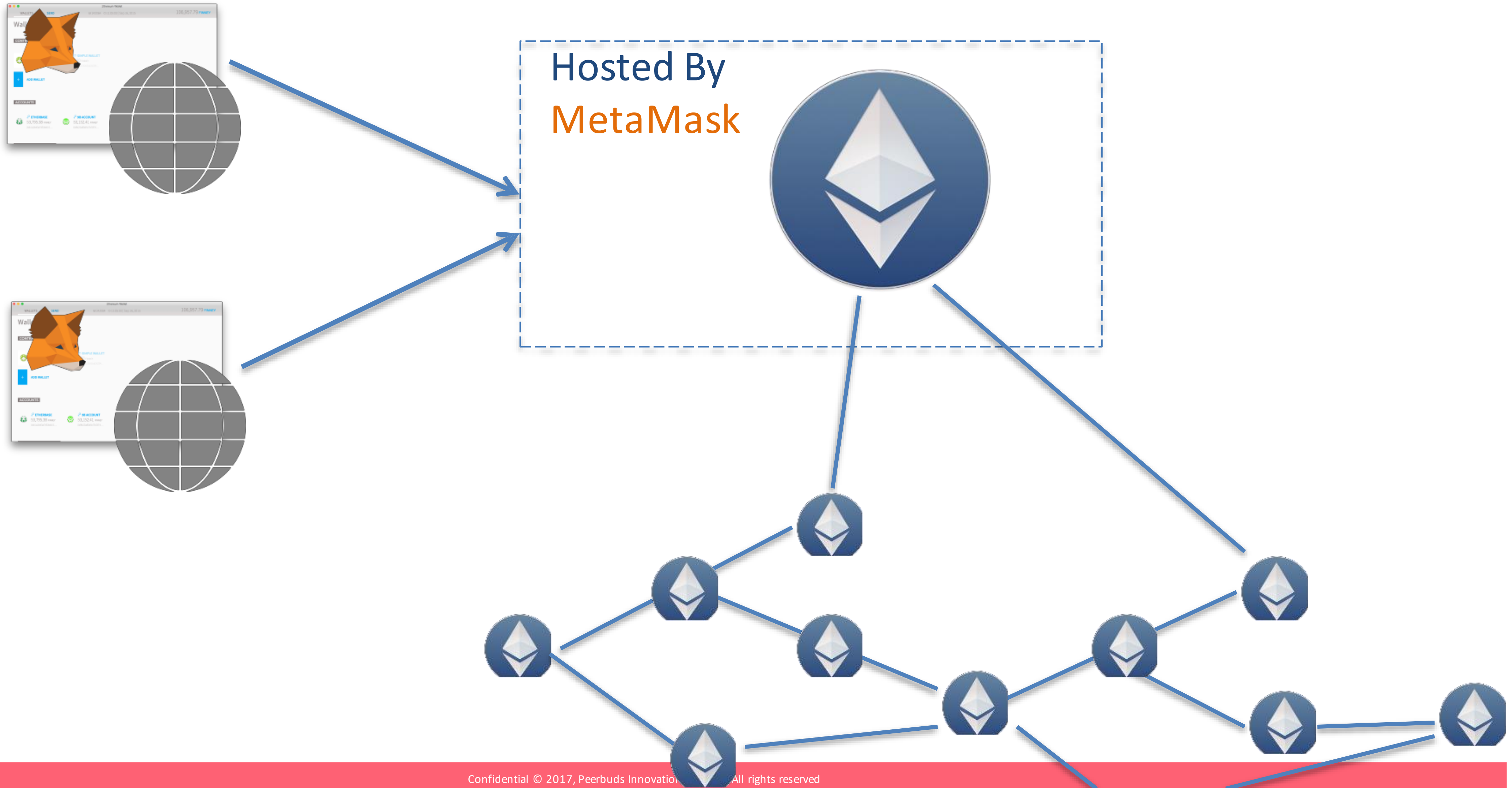
DAPP Infrastructure

Option#2 Client in midtier



DAPP Infrastructure

Option#3 Meta Mask Chrome Plugin



METAMASK

- Manage accounts in a browser vault
  - Export/Import accounts
  - Send Funds
- Exposes web3 object to browser app
  - Single Page Applications
- Supports multiple endpoints
- Does not support mining