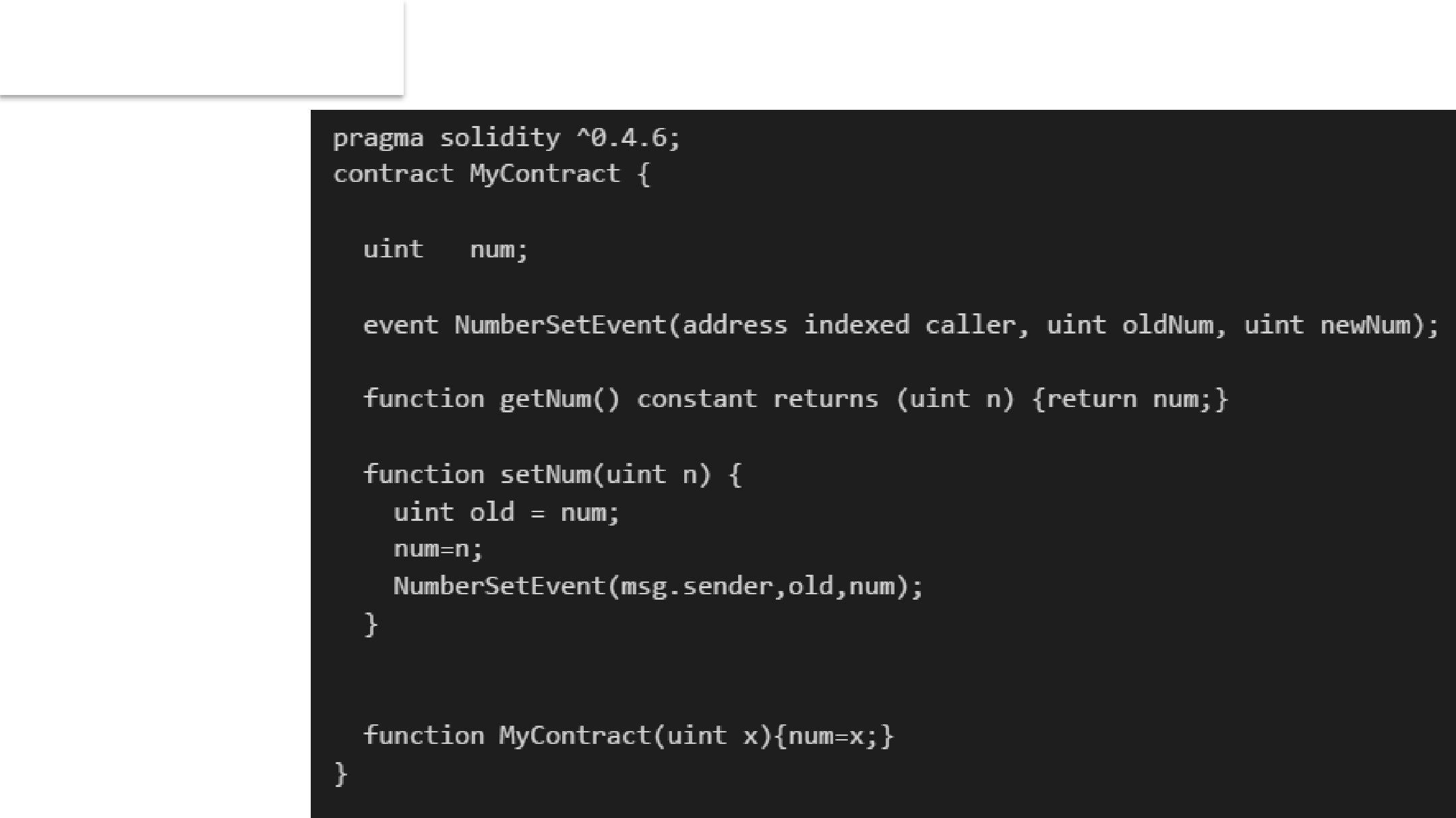
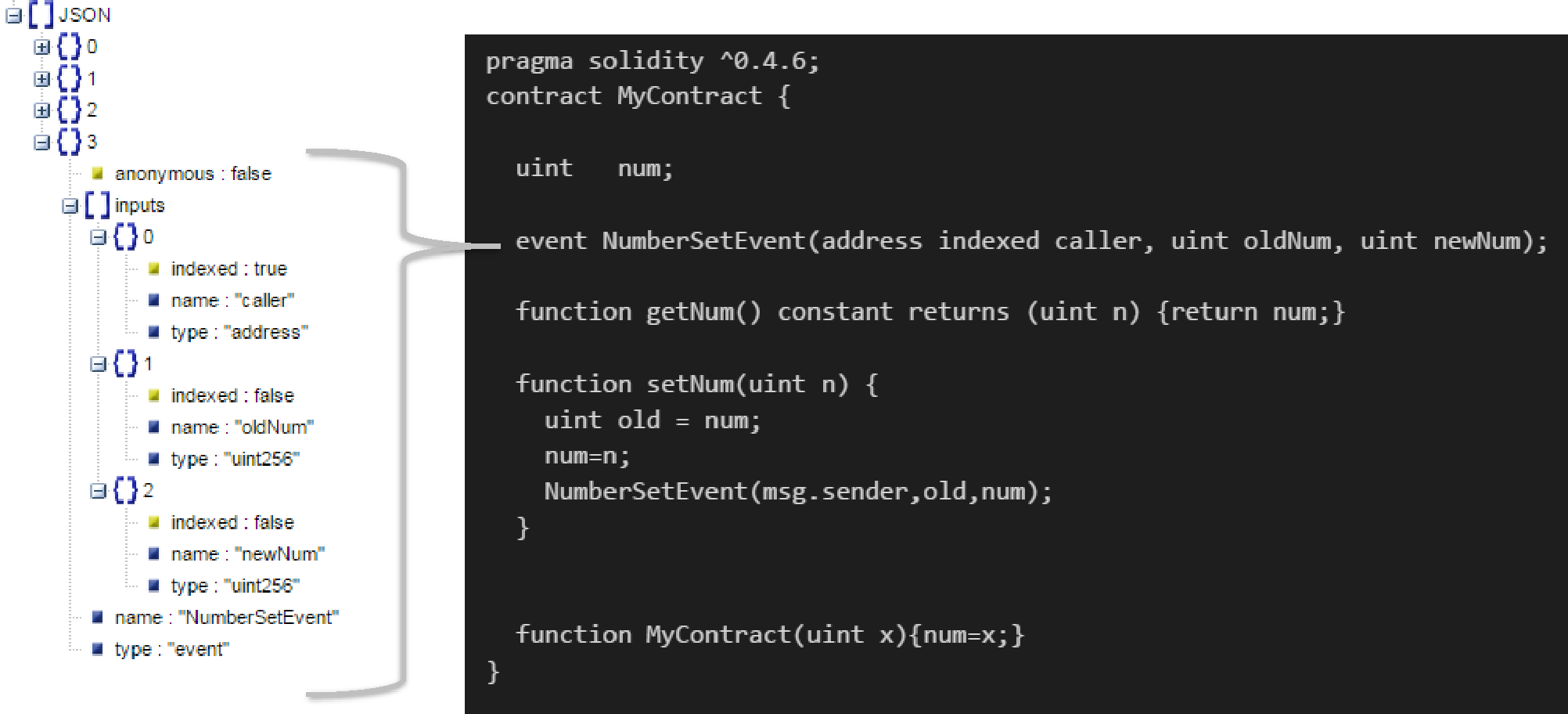
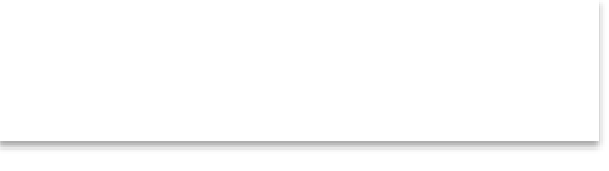
Web3 JS API:

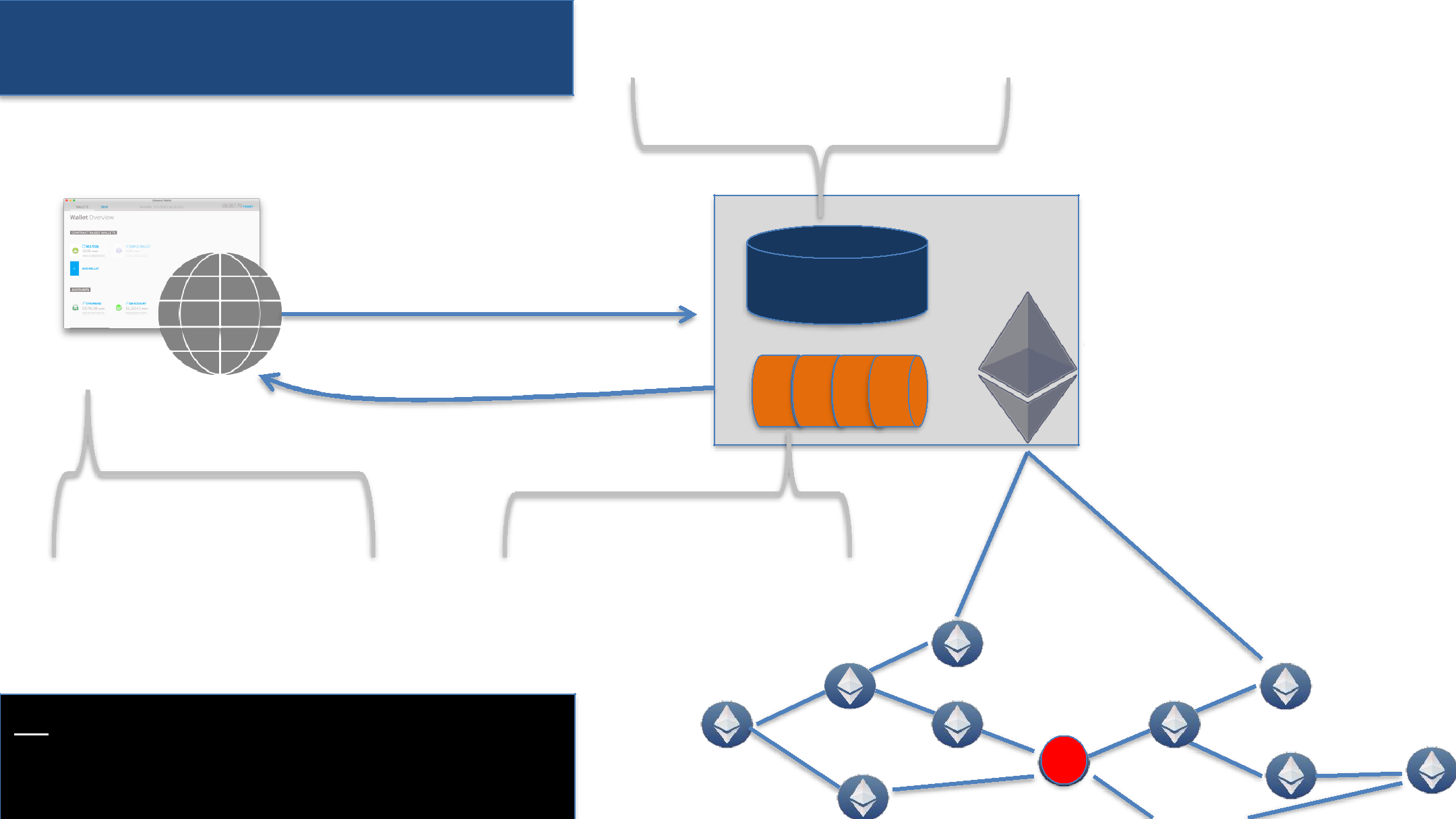
* Logs
* Events

|  |  |
| --- | --- |
| Contract Events | Contracts may emit events |
|  |  |



|  |  |
| --- | --- |
| Contract Events | Contracts may emit events |
|  |  |





Ethereum Logs & Events

Contract State Changes

|  |  |  |
| --- | --- | --- |
|  | sendTransaction | BlockChain |
| Dapp | {Event} |  |

|  |  |  |
| --- | --- | --- |
|  | Logs |  |
| Watching for events | Logs from execution |  |
|  |  |

Network

|  |  |
| --- | --- |
| **PS** |  |
| • All event Logs available on all nodes | M |

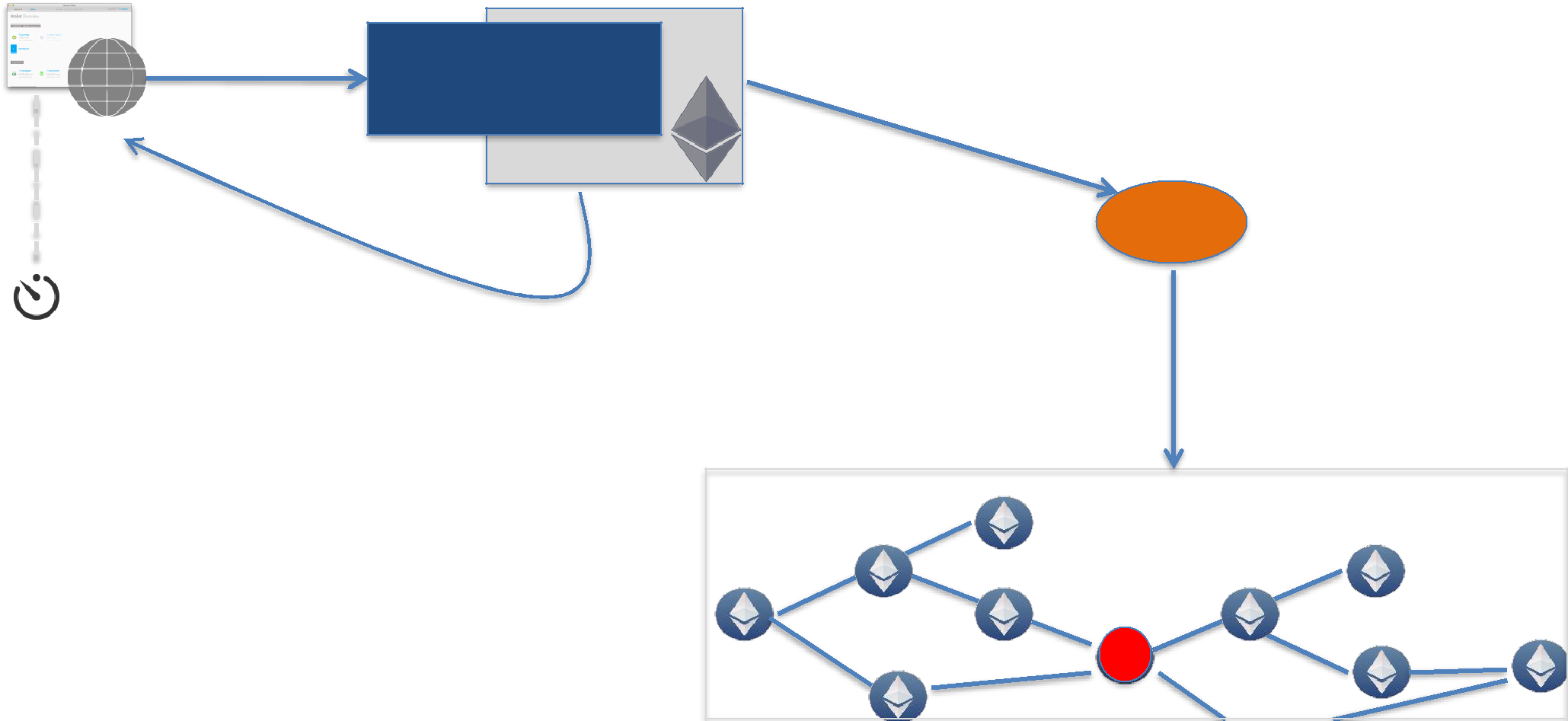
* If watching the Dapp gets notified

Event/Log usage patterns

1. Receive data for transaction
2. Asynchronous trigger
3. Cheap data storage
4. Receives data for transaction
   * 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



Calls

Send Transaction

Creates

Tx #

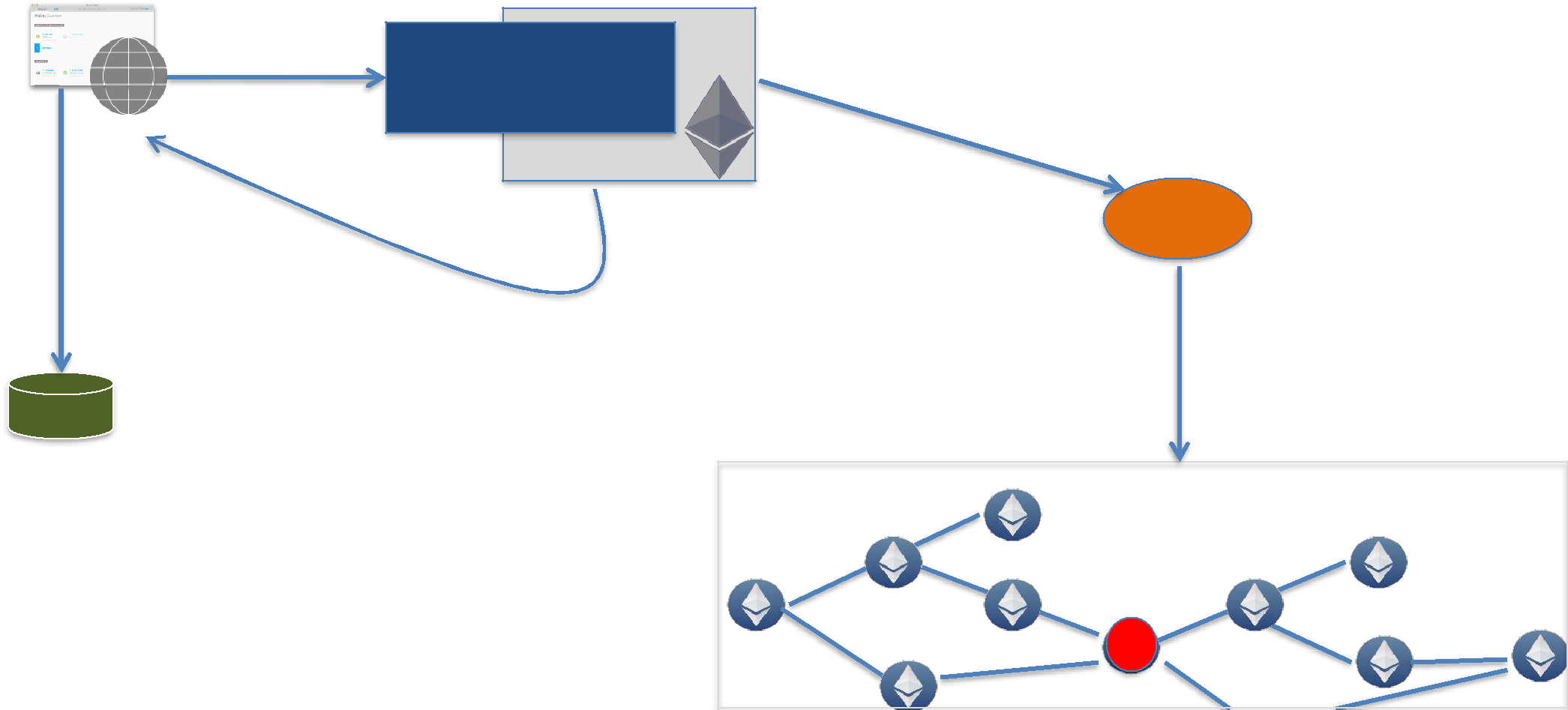
**Event:** Call Return Data

Wait for mining

Network

|  |  |
| --- | --- |
| • *Example:* Sample contract | M |

2. Asynchronous Notifications



Calls

Send Transaction

Creates

Tx #

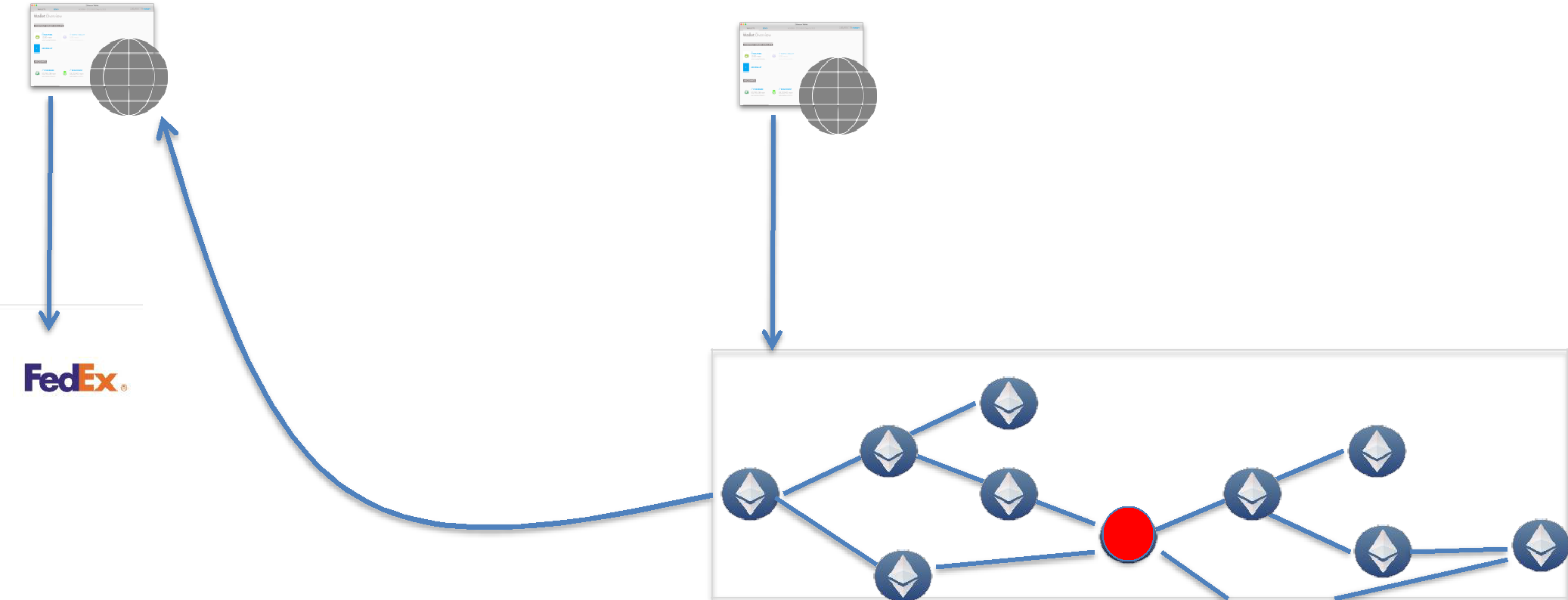
Update

**Event:** Notification

Network

|  |  |
| --- | --- |
| • *Example:* Wallet app | M |

2. Asynchronous Notifications



|  |  |  |
| --- | --- | --- |
| <Merchant> | <Customer> |  |
|  |  |

Invoke Contract for payment

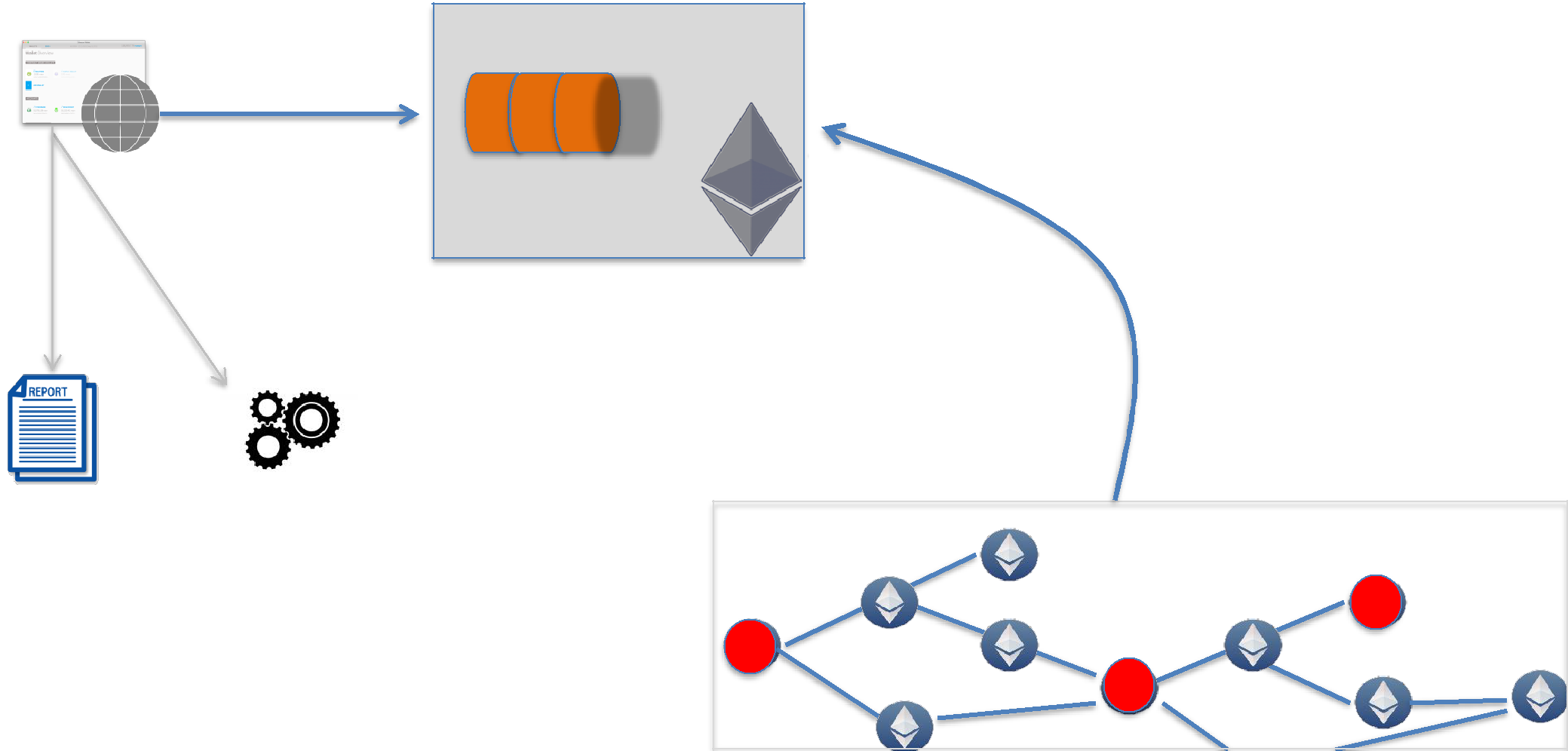
{Event}

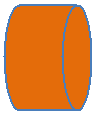
Network

M

* 1. 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



Logs 

**Events** stream

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Reports | Processing | ….. |  |  |
|  |  |  | Network |  |
| • *Example:* | |  | M |  |
|  | M |  |
| Encrypted customer identity | | | M |  |

* 1. Data storage
* Cheaper than contract storage

•

Log data storage cost

8 Gas/byte

•

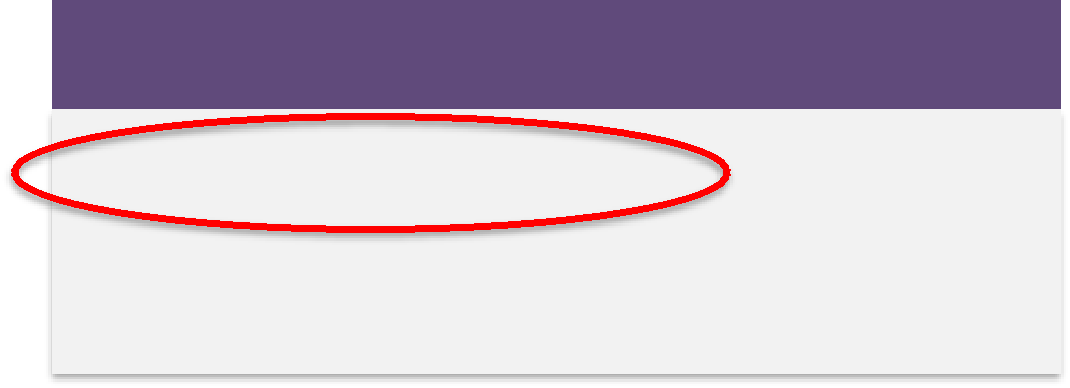
Contract data storage cost

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. Using the contract instance

Using Filter



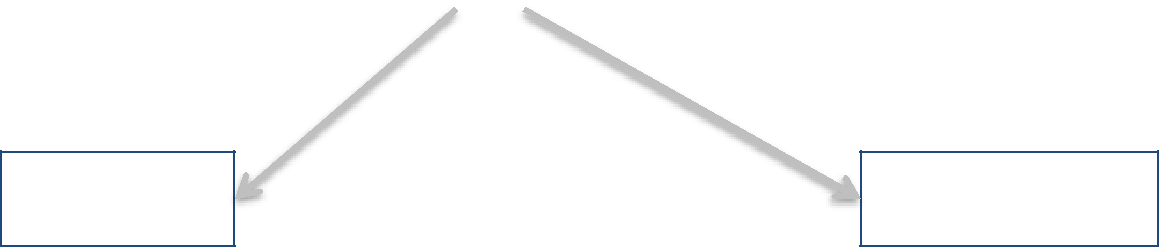
|  |  |
| --- | --- |
| var filter = web3.eth.filter(…) | • Argument = events selection criteria |
|  |  |

filter.watch(…) *filter.stopWatching()*

filter.get(…)

web3.eth.filter(…)

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



“latest” “pending”

*Result=*Block Hash of latest Block *Result=*Transaction Hash of latest txn

1. *web3.eth.filter*(options\_object)

Options\_object

•

Block range

*fromBlock, toBlock*

•

Specific contract instance

*[address]*

•

Event data

•

Data in the log fields

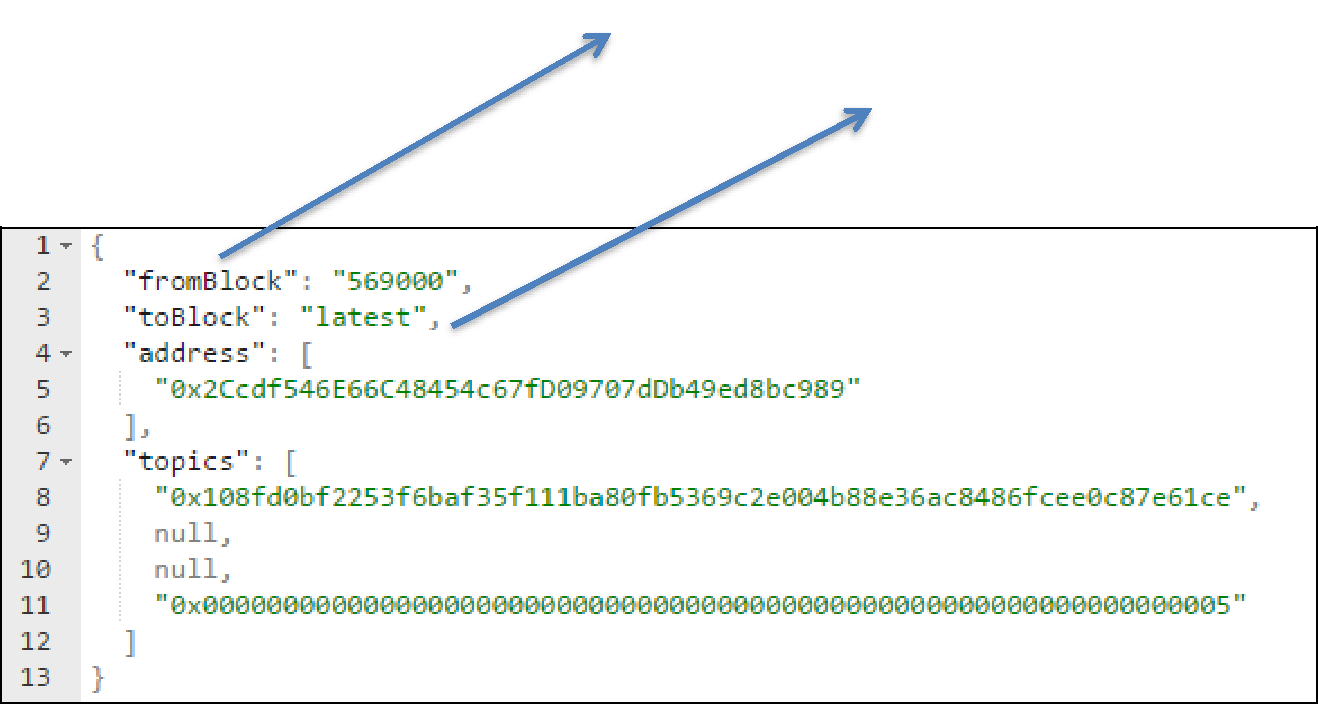
topic: *[‘event-signature’, ‘data1’, ‘data2’, ‘data3’]*

* Fields marked *indexed* used in topics
* Maximum of 3 indexed fields & order is important

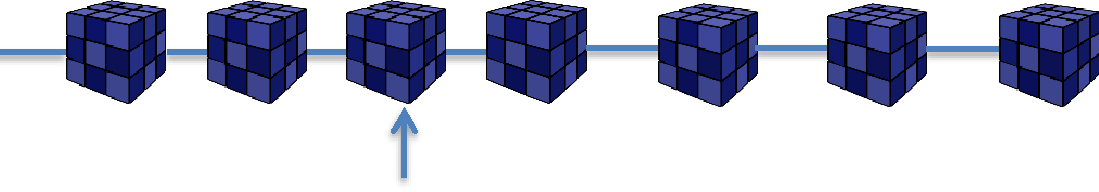


Options JSON

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

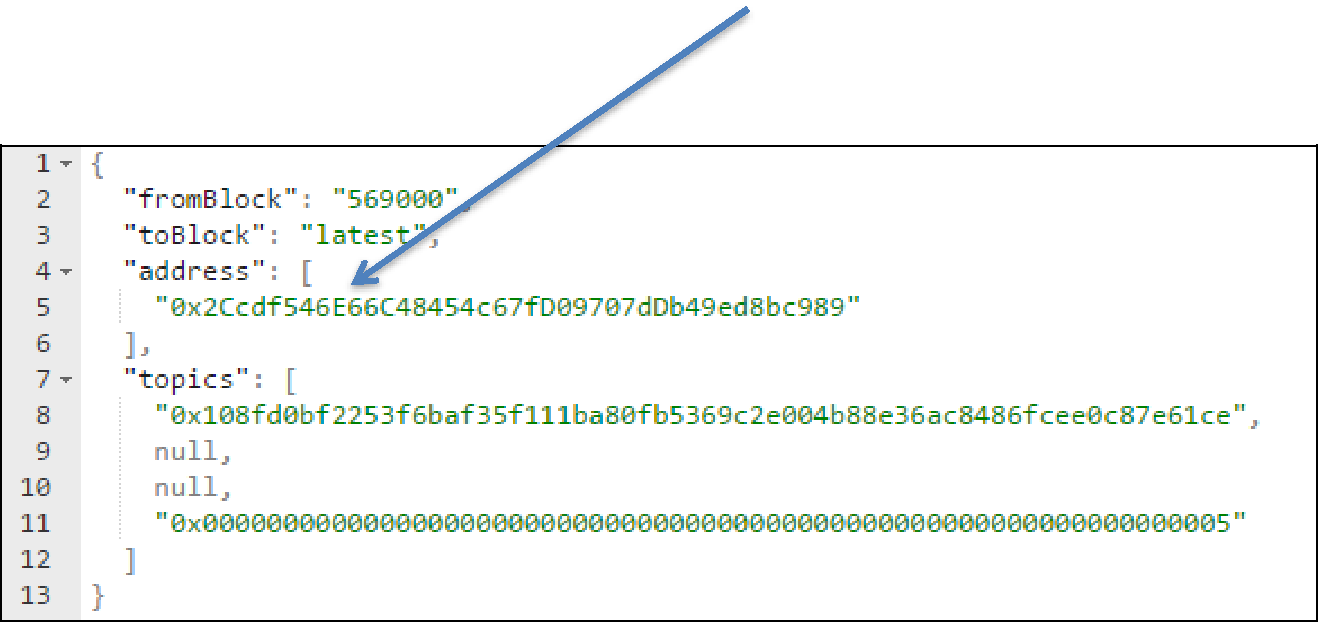


toBlock = latest

#569000

Options JSON

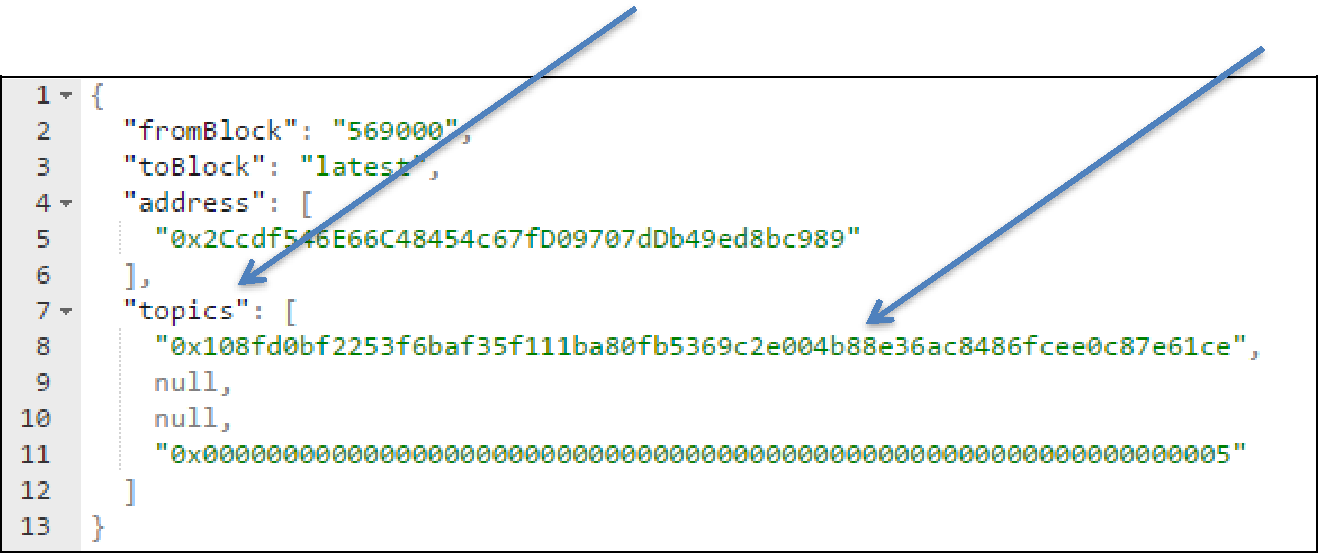
Array of contract addresses



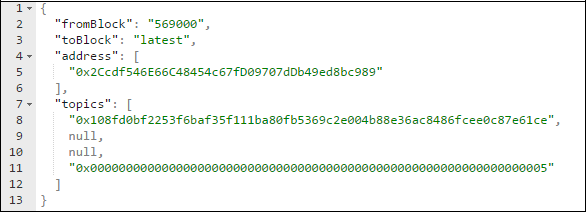
Options JSON

topics = event data criteria

topics[0] = Event Signature



Options JSON



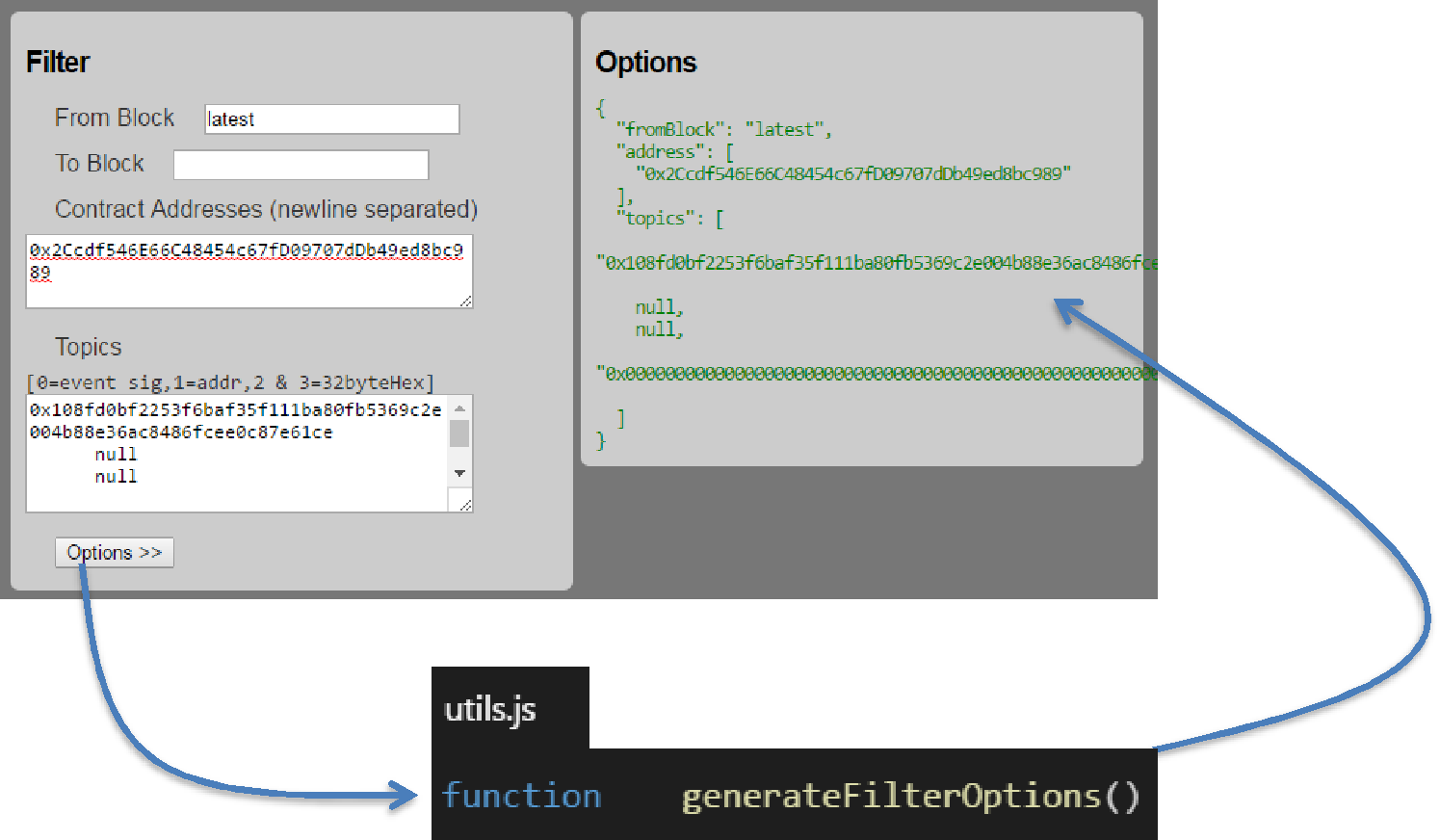
event NumberSetEvent(address indexed **caller**, bytes32 indexed **oldNum**, bytes32 indexed **newNum**);

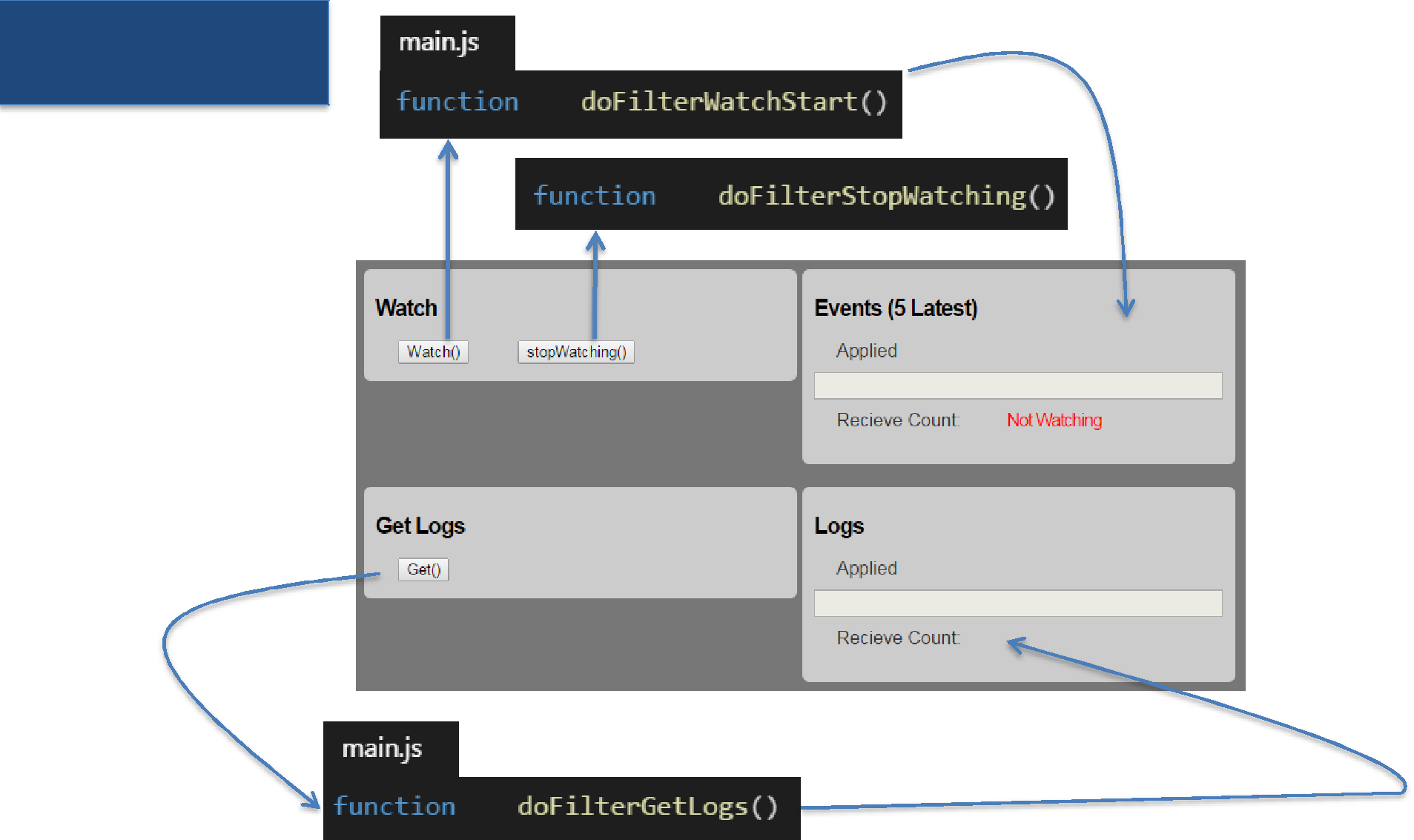
setNum( **5** ) **{Event Received}** setNum( **6** ) **{NO Event Received}**

watch() & get()

* filter.**get**(callback\_func)
  + - Result : Array of events
  + filter.**watch**(callback\_func)
    - * Result : event data

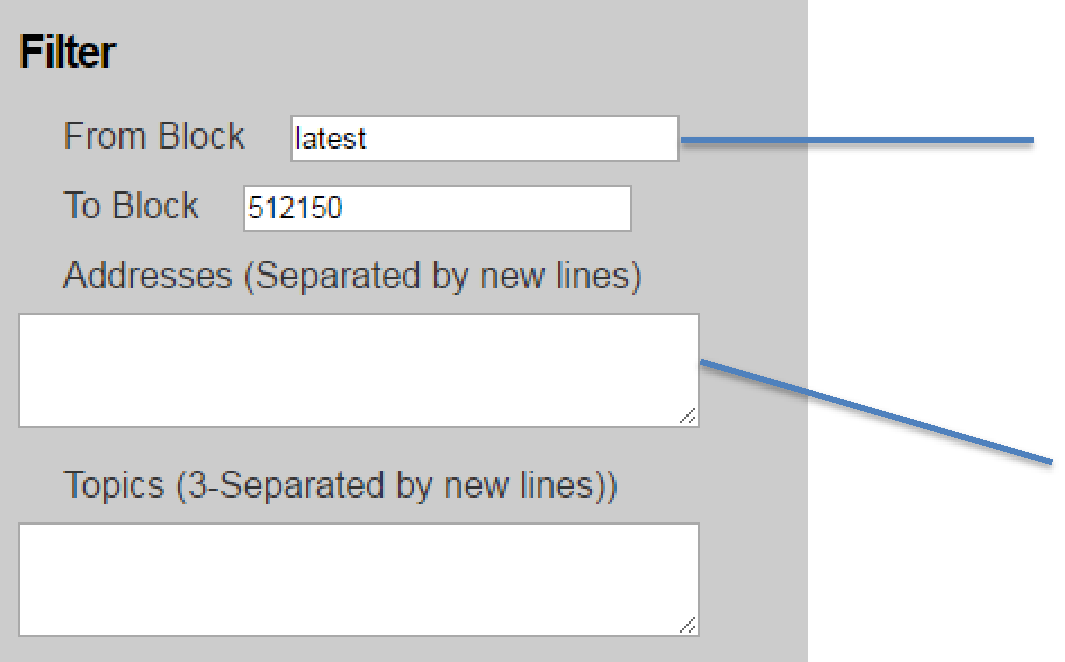
Walkthrough



Walkthrough

1. web3.eth.filter(options)

*web3.eth.filter*(options) for events



*fromBlock*:by default ‘latest’ ; number or hash

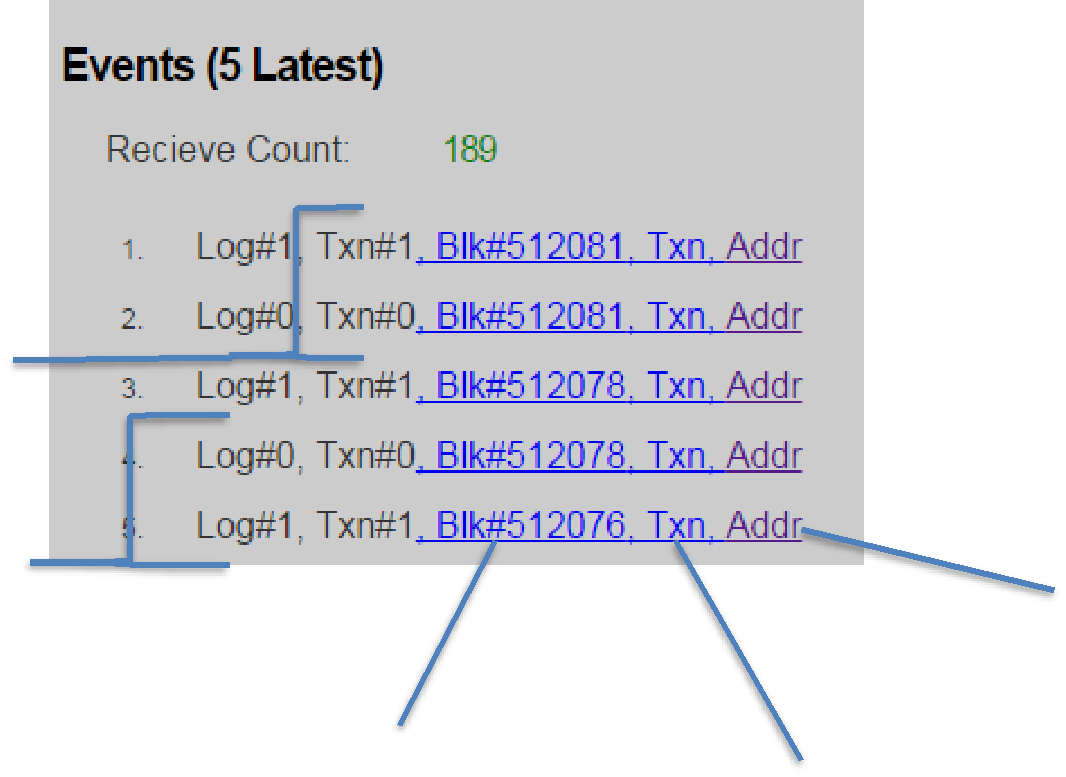
 *toBlock*:leave blank for continuous watching

*address:* Contract Address(es)

 *topic*:Indexed topic data

Event Data

*Events received in real time*



transactionIndex

logIndex

address

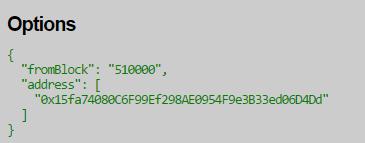
blockNumber

transactionHash

blockHash

Get Logs

*web3.eth.filter*(options) for events



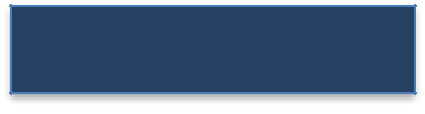
*Array of logs*



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* )

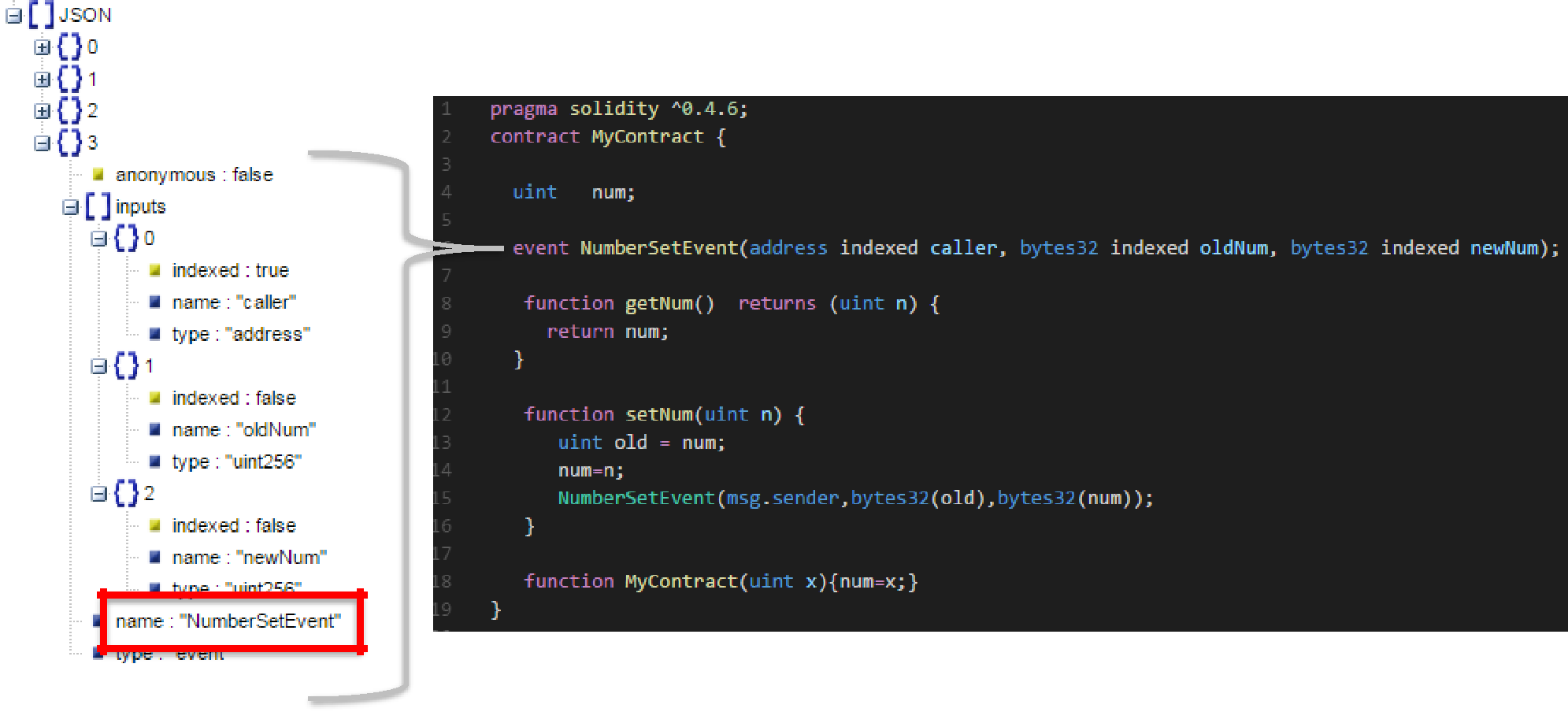
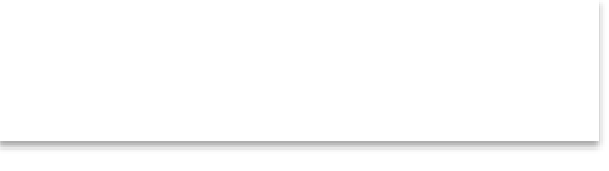
1. Deploying the contract code to EVM

var contractInstance = contract.**at**(*contract\_address*)

1. Invoking a contract function
2. Watch for events & Get events data from *Log*



|  |  |
| --- | --- |
| Contract Events | Like methods, events are part of *abiDefinition* |
|  |  |



Event Filtering

*additionalOptions*



*Indexed or topics options*

Contract Event

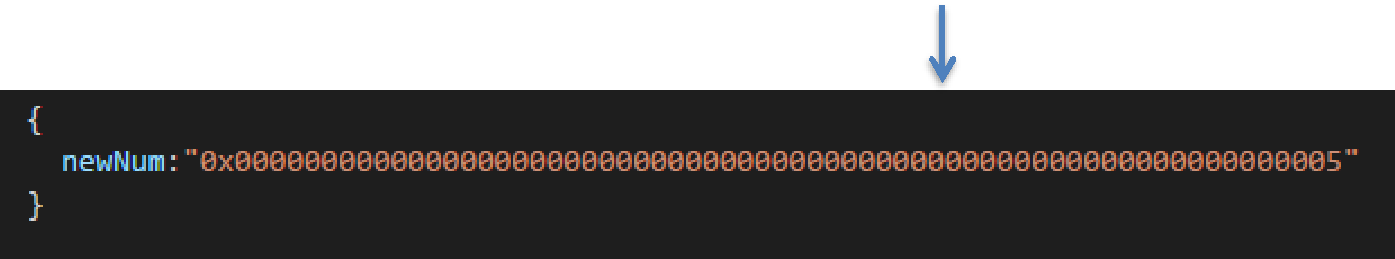
var contractEvent =

contractInstance.**allEvent**(*additionalOptions*)



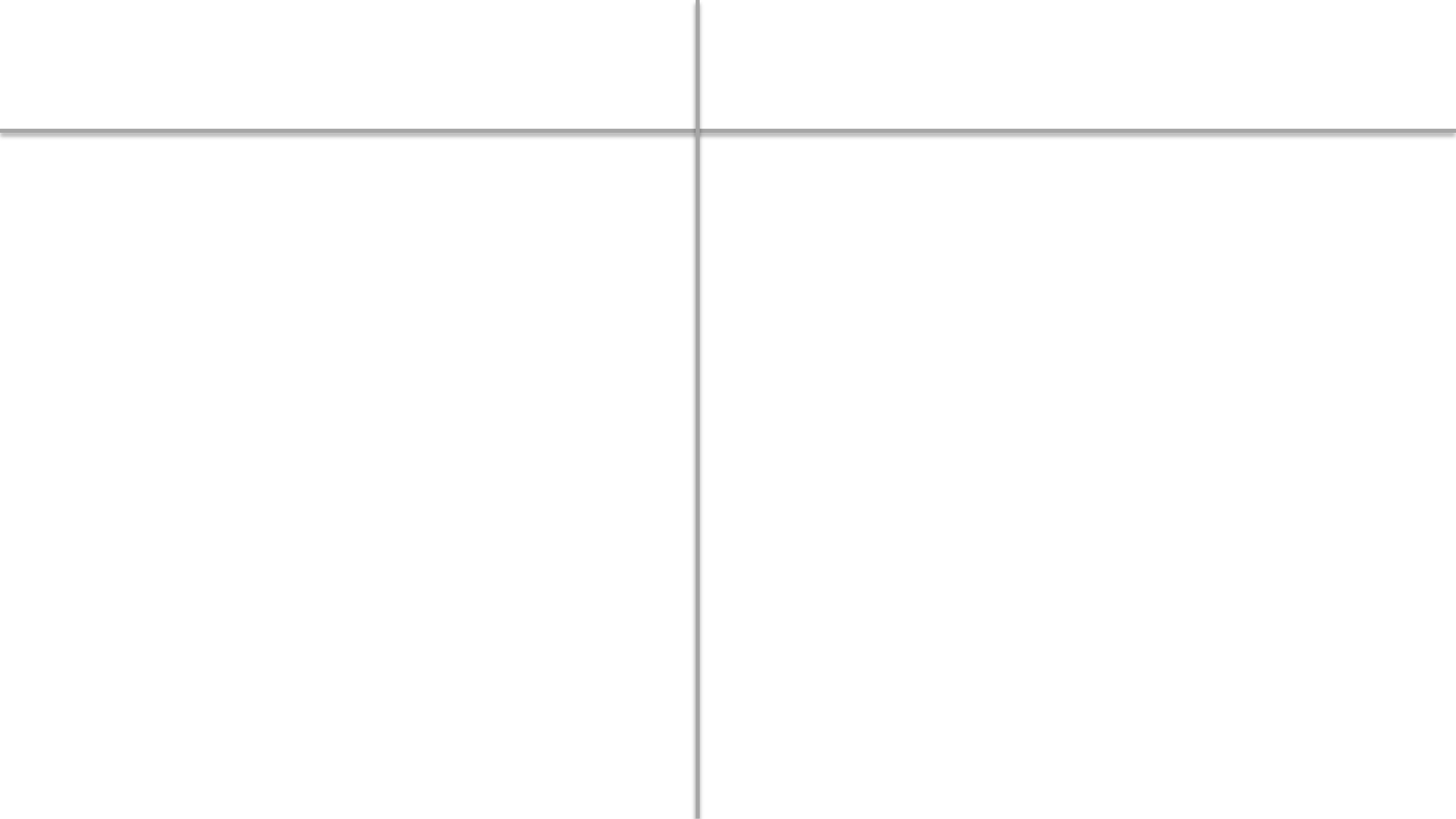
var contractEvent =

contractInstance.**NumberSetEvent**( *indexedOptions, additionalOptions*)



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* *Event : get/watch*

|  |  |
| --- | --- |
| • All events from any source | • Events from specific contract instance |

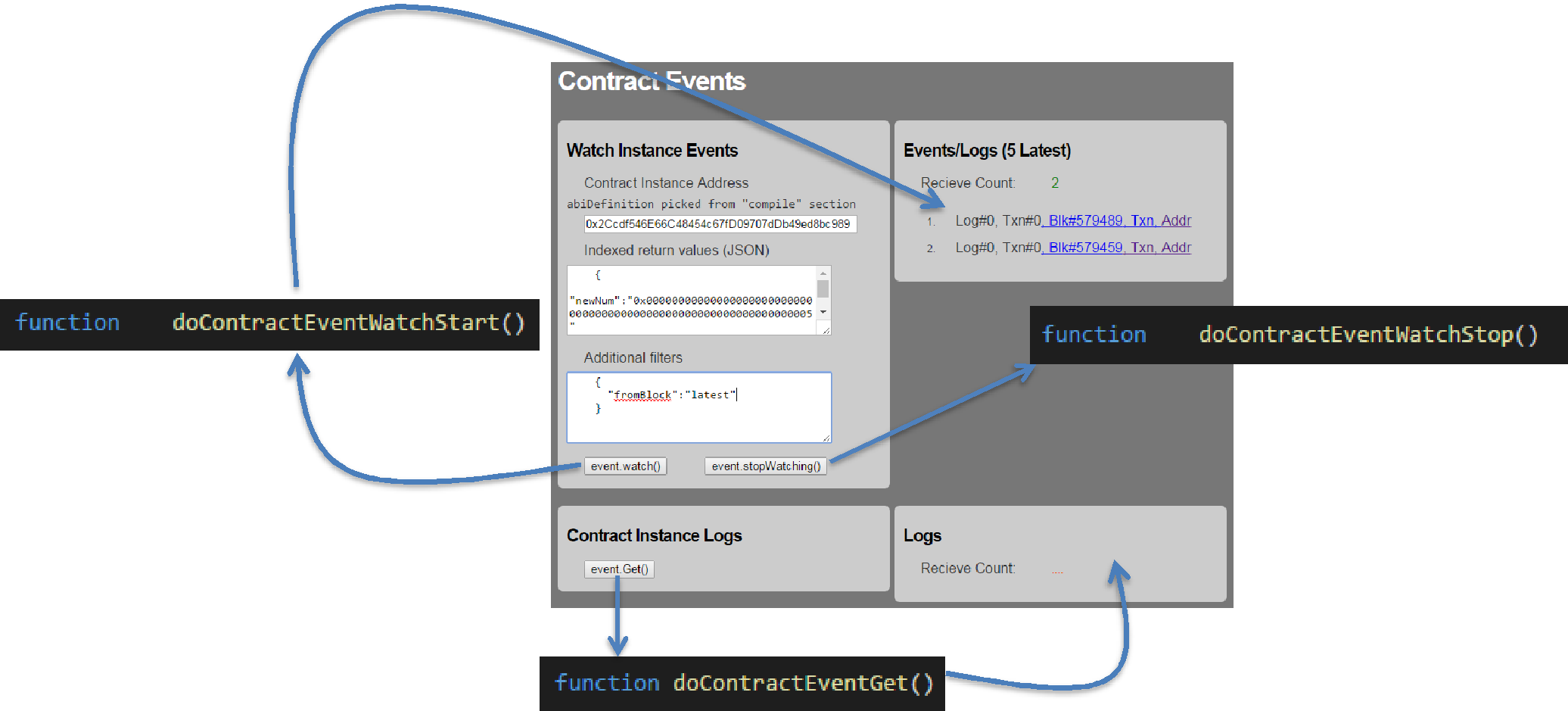
|  |  |
| --- | --- |
| • May be used for writing tools etc | • For Dapp only |

|  |  |
| --- | --- |
| • Indexed data in options/topics array | • Indexed/Topic data is a JSON object |

Web3 JS API:

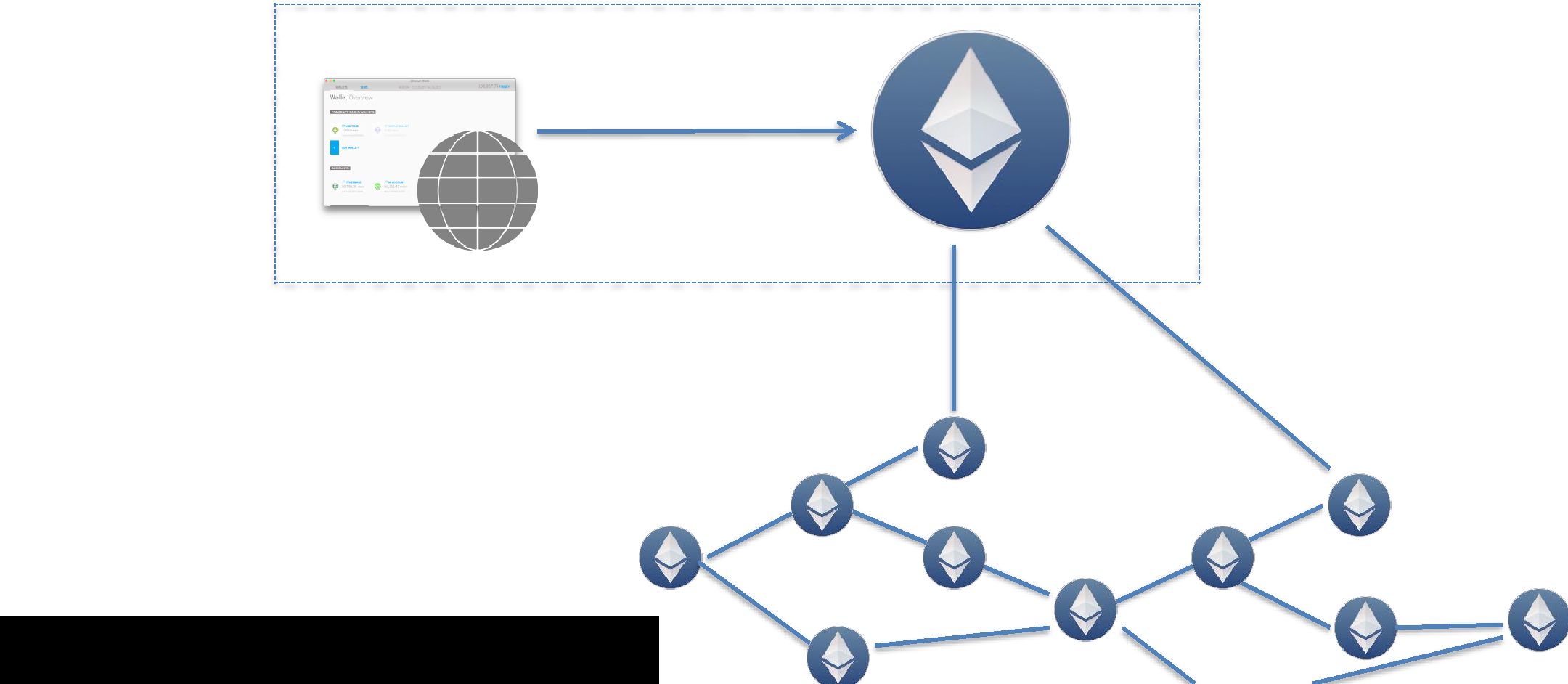
* DAPP Infrastructure

get(), watch(), stopWatching()





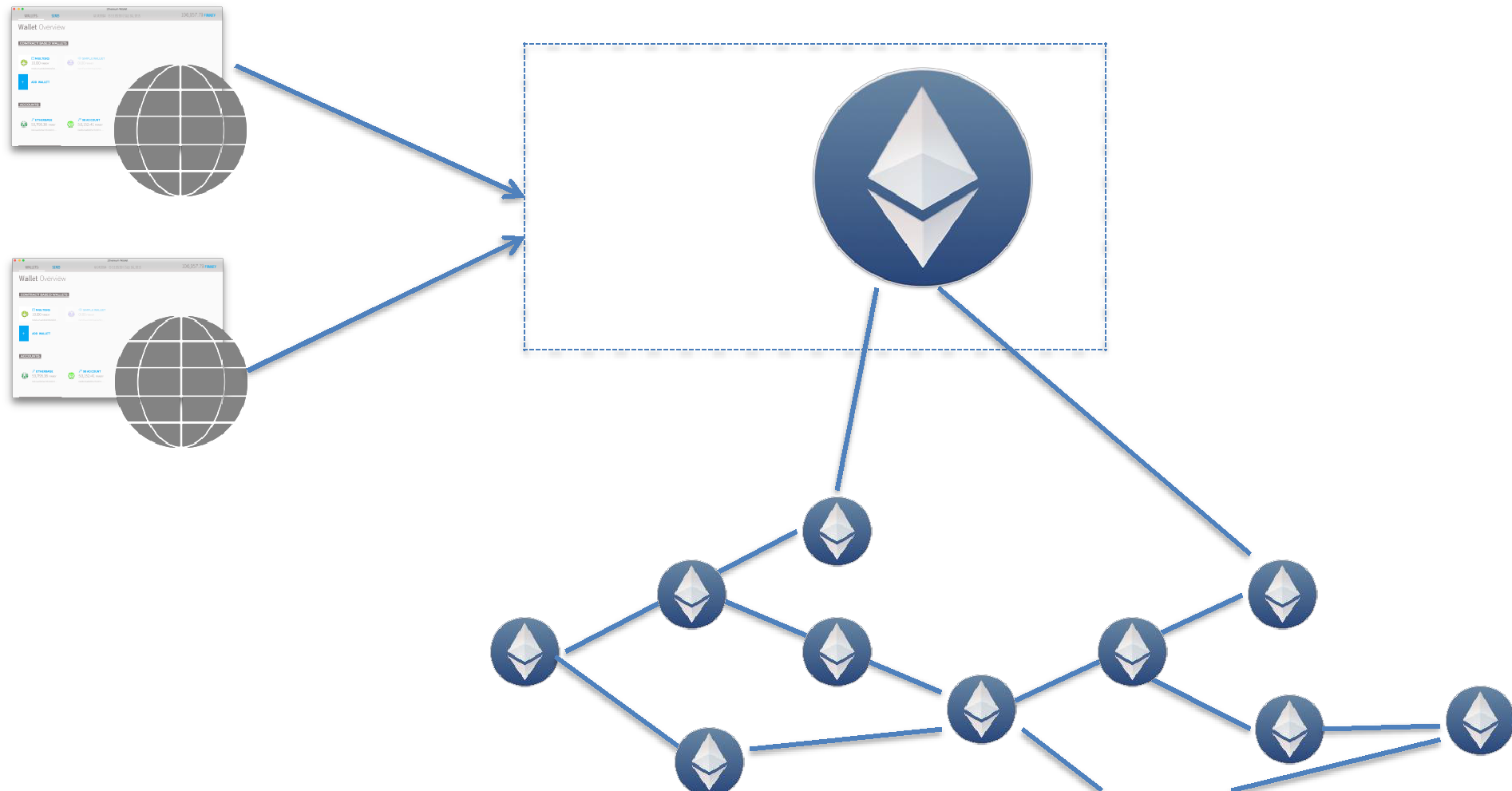
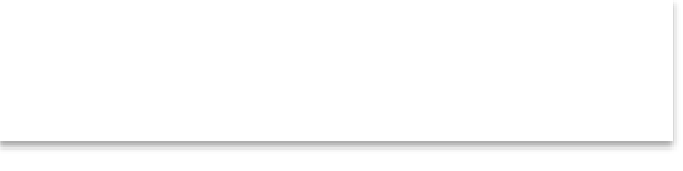
DAPP Infrastructure Option#1 DAPP co-located with the client



Local Node

Deploying node locally is expensive

|  |  |
| --- | --- |
| DAPP Infrastructure | Option#2 Client in midtier |
|  |  |

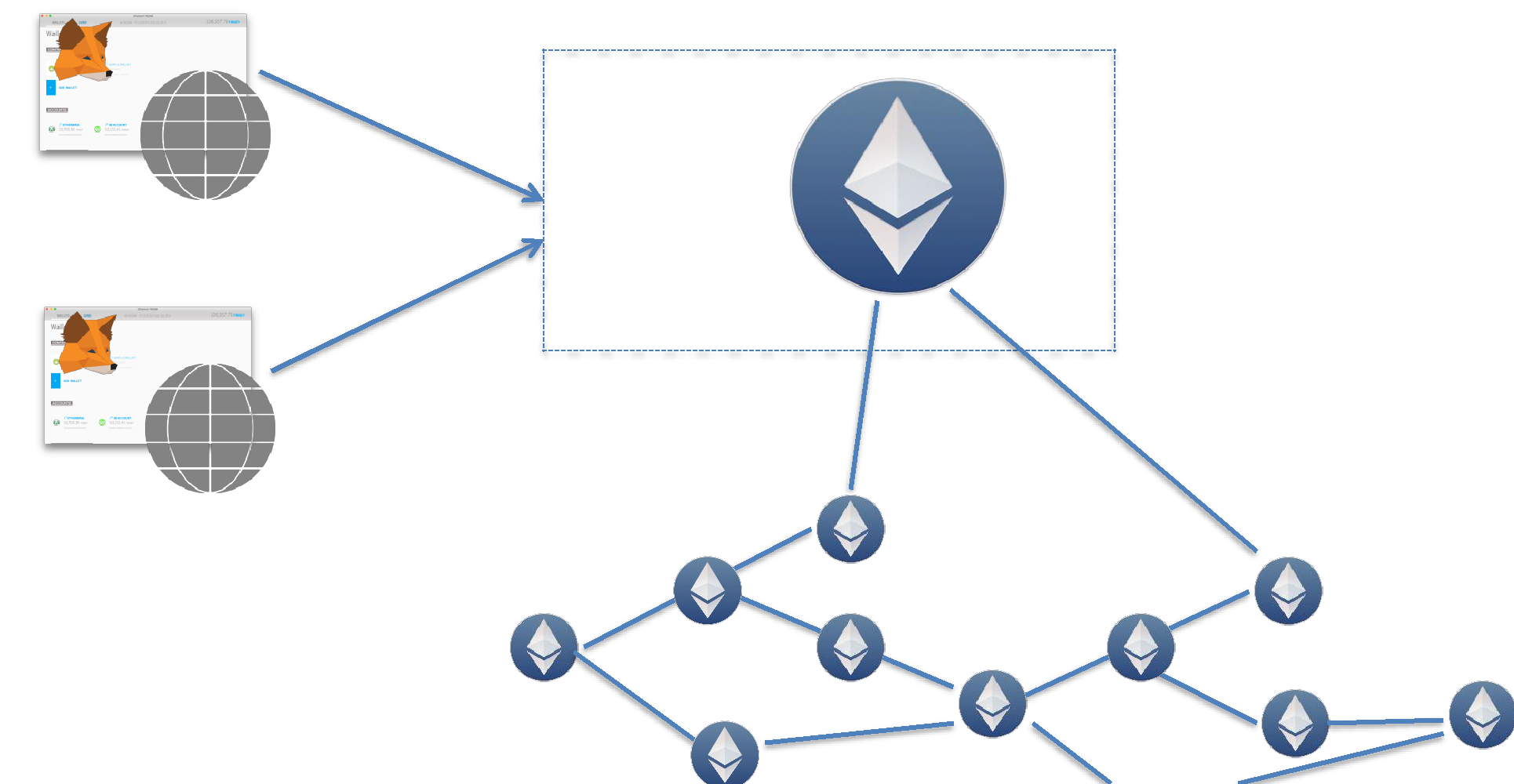


Hosted

By

Organization

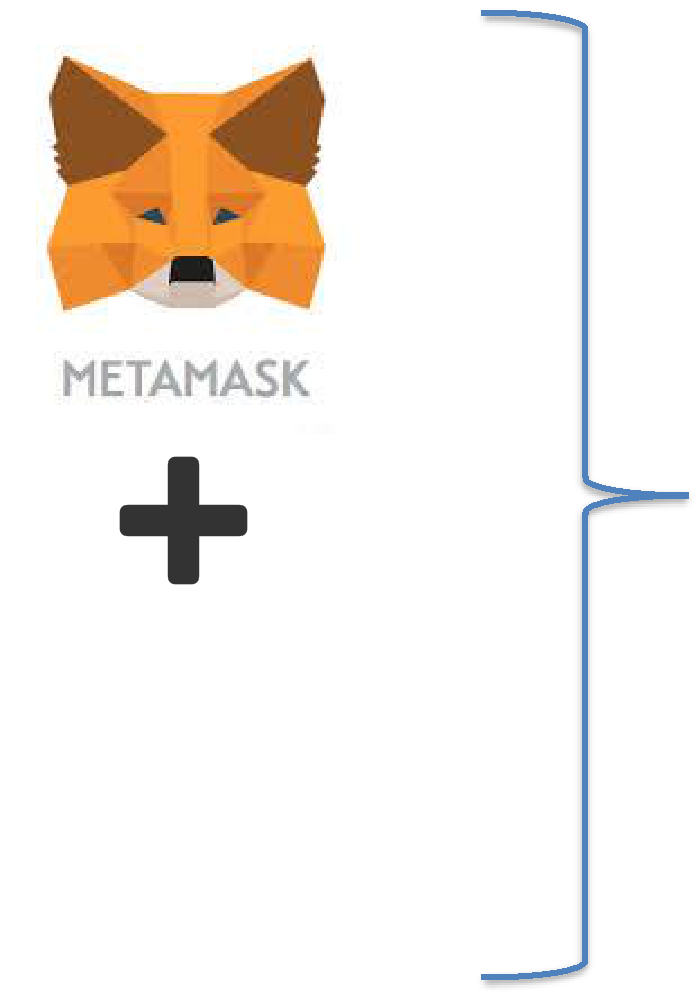
DAPP Infrastructure Option#3 Meta Mask Chrome Plugin



Hosted

By

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

Web3 JS API:

* Compilation

Compilation Output



Bytecode / EVM

*code*



* Deployed to the blockchain

Application Binary Interface

*abiDefinition*

* Interface definition
* Needed for contract deployment
* Needed for invoking contracts

Compiler options



OLC



Web3 Compilation

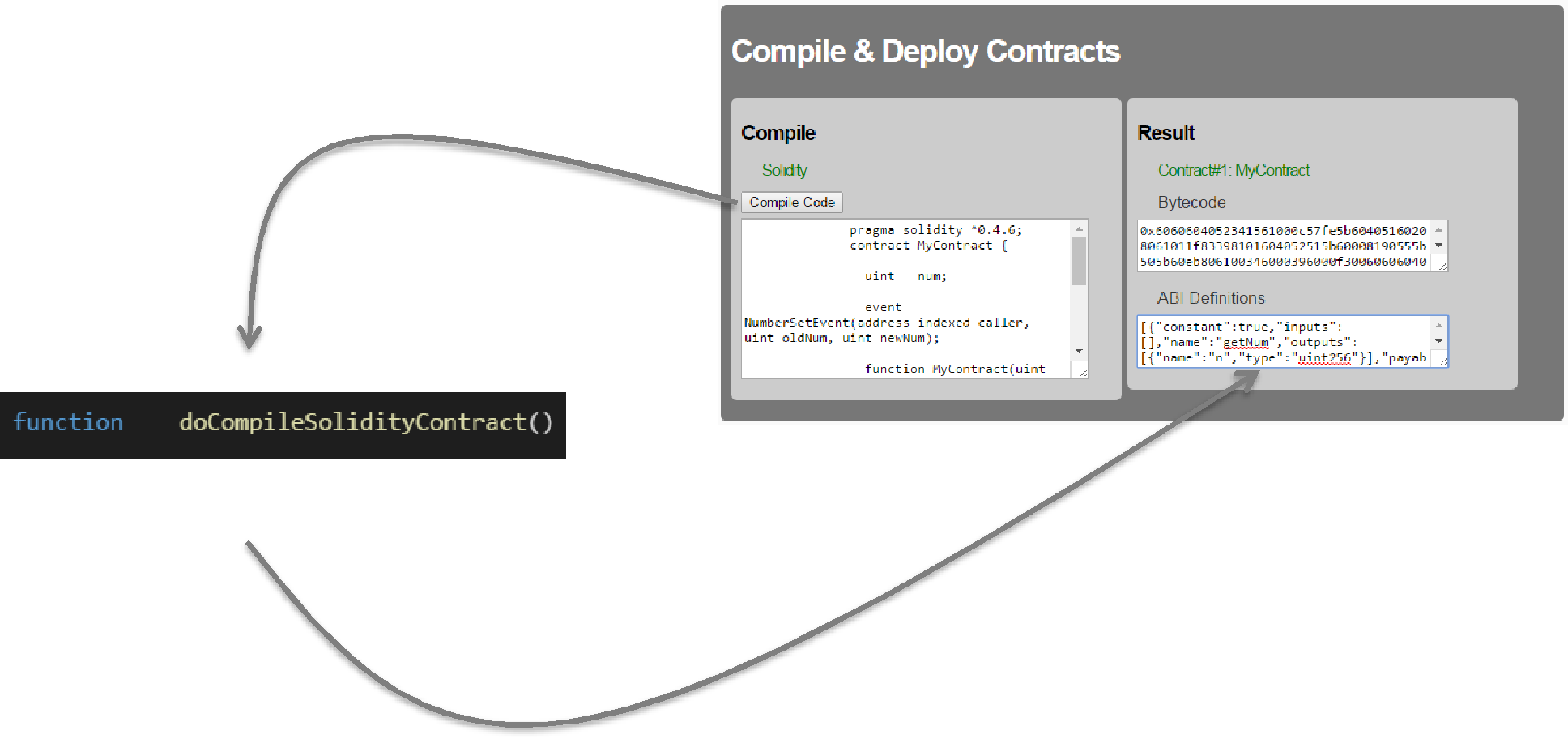


* Supported till geth version 1.5.9

web3.eth.**compile.solidity**(source\_string, callback\_func)

* TestRPC supports this API; but may not support it in future
* MetaMask does not support it

Solidity Compiler



web3.eth.compile.solidity(source\_string, callback\_func)