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.py



web3J

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/>



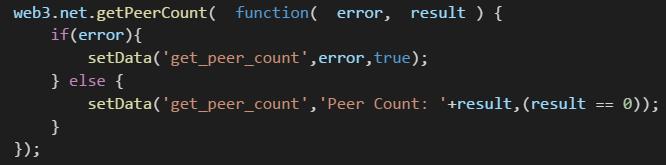
|  |  |  |
| --- | --- | --- |
| *Web3* API Overview | <https://github.com/ethereum/wiki/wiki/JavaScript-API> |  |
|  |  |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  | eth | Ethereum blockchain related methods |  |
|  |  |  |  |  |
|  |  |  |  |  | Node’s network status |  |
|  |  |  |  | net |  |
|  |  |  |  |  |
|  |  |  |  |  | Account functions and sending |  |
|  |  |  |  | |  |
|  |  |  | personal | |  |
|  |  |  |  |
|  |  |  |  |  | Get/put for local *LevelDB* |  |
|  |  |  |  |  |  |
|  |  |  |  | db |  |
|  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  | shh | P2P messaging using *Whisper* |  |
|  |  |  |  |  |
|  |  |  |  |  |  |  |

Web3 JS Asynchronous Calls

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



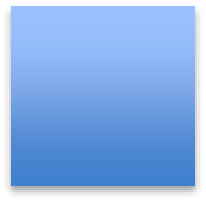
Install NodeJS

Tools/Components



|  |  |  |  |
| --- | --- | --- | --- |
| 1 |  |  |  |
| Install Yeoman | > npm install –g yo |  |
|  |  |  |  |

|  |  |
| --- | --- |
| Install Yeoman webapp template | > npm install –g generator-webapp |
|  |  |

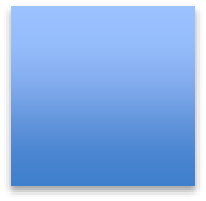


|  |  |  |  |
| --- | --- | --- | --- |
| 3 |  |  |  |
| Install Gulp | > npm install –g gulp |  |
|  |  |  |  |

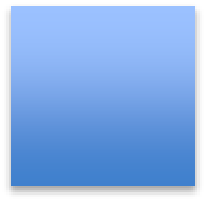


4 Install Bower > npm install –g bower

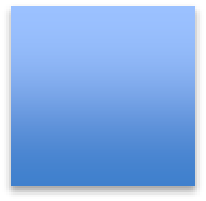
Setup Dapp



|  |  |
| --- | --- |
| 1 | Create a folder for application |
|  |  |



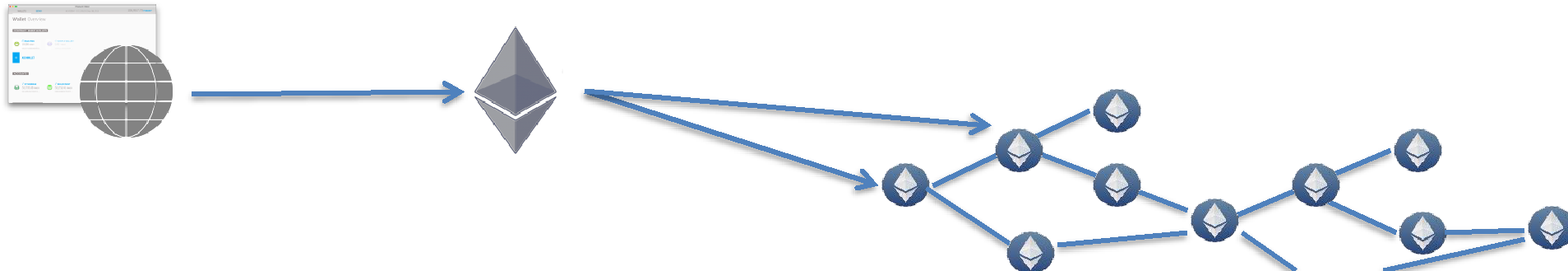
|  |  |  |  |
| --- | --- | --- | --- |
| 2 |  |  |  |
| Create the application | > yo webapp |  |
|  |  |  |  |



|  |  |  |  |
| --- | --- | --- | --- |
| 3 |  |  |  |
| Install web3 library | > bower install web3 --save |  |
|  |  |  |  |

Web3 Sample App: 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*



JSON/RPC

|  |  |  |
| --- | --- | --- |
| geth | Testnet |  |
|  |  |

Web3 Workbench

* Decentralized application

• Will show the use of web3 API

http://**T**he**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

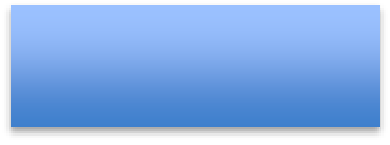
3. Run > npm install

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 |
|  |  |

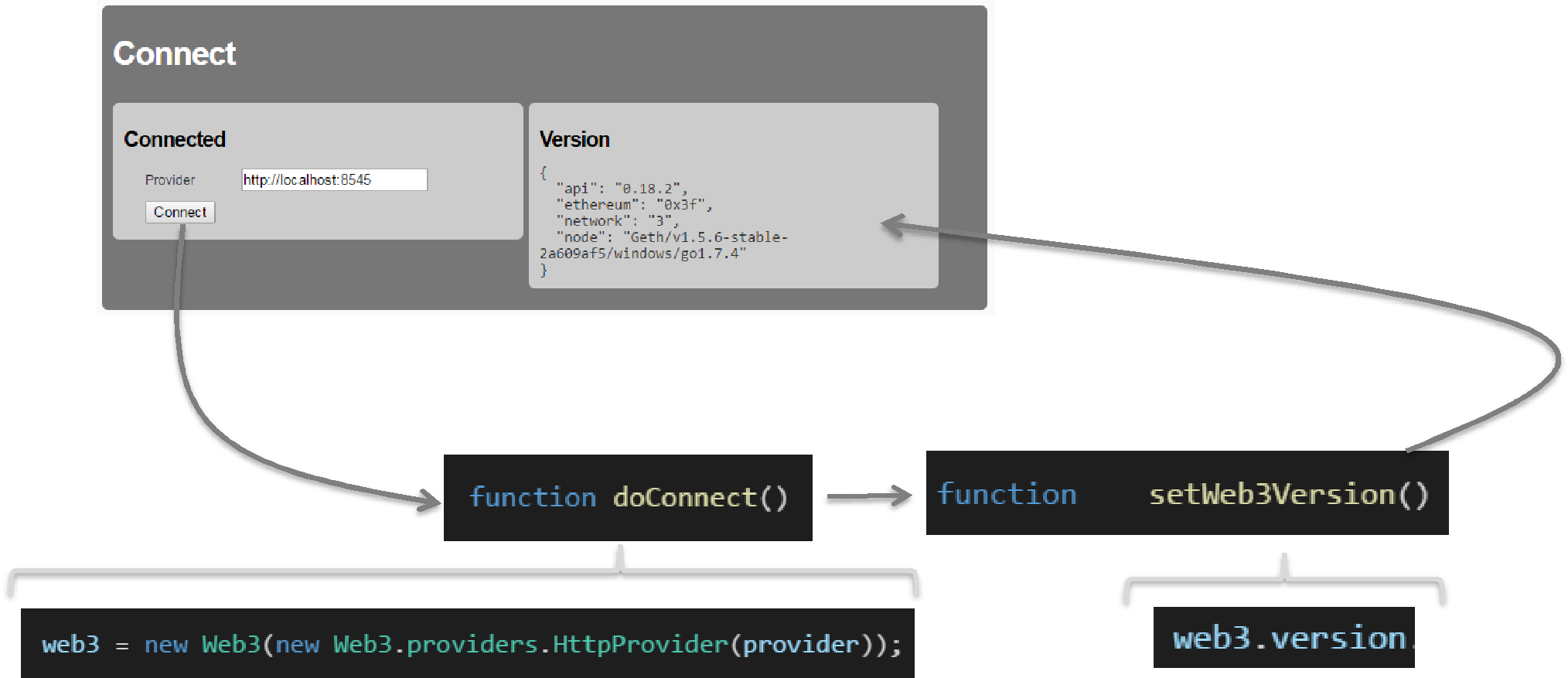


|  |  |
| --- | --- |
| main.js | • All web3 JS API calls + some UI related code |
|  |  |



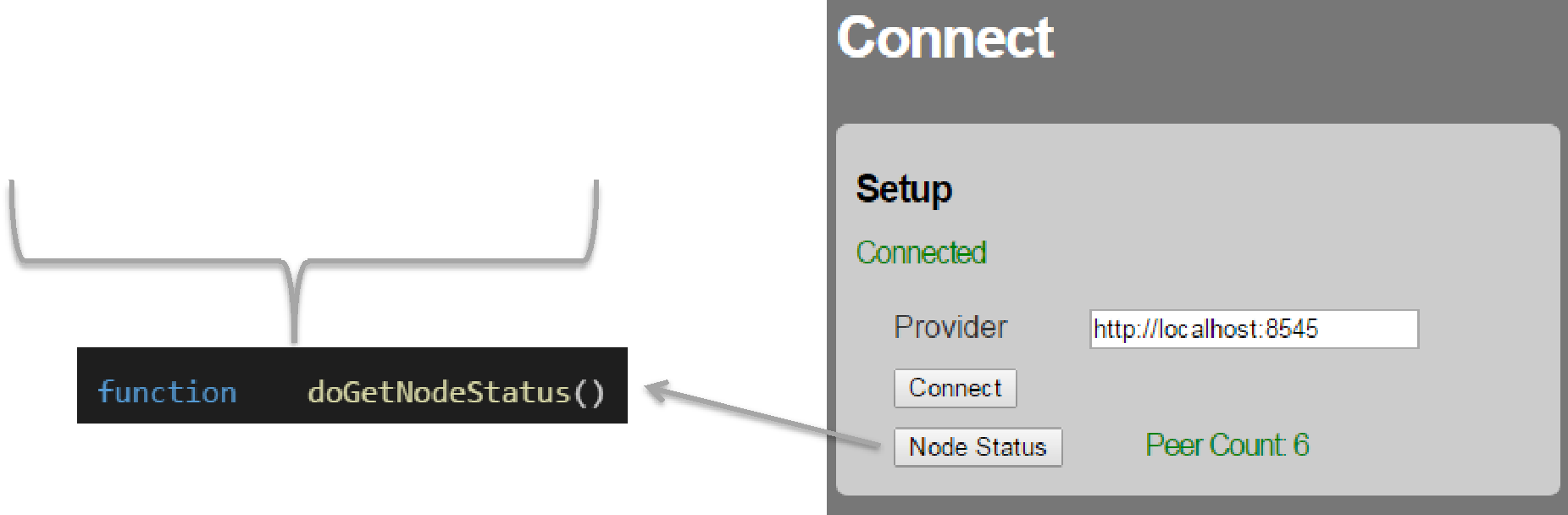
|  |  |
| --- | --- |
| utils.js | • UI related utility functions |
|  |  |

Connect & Version



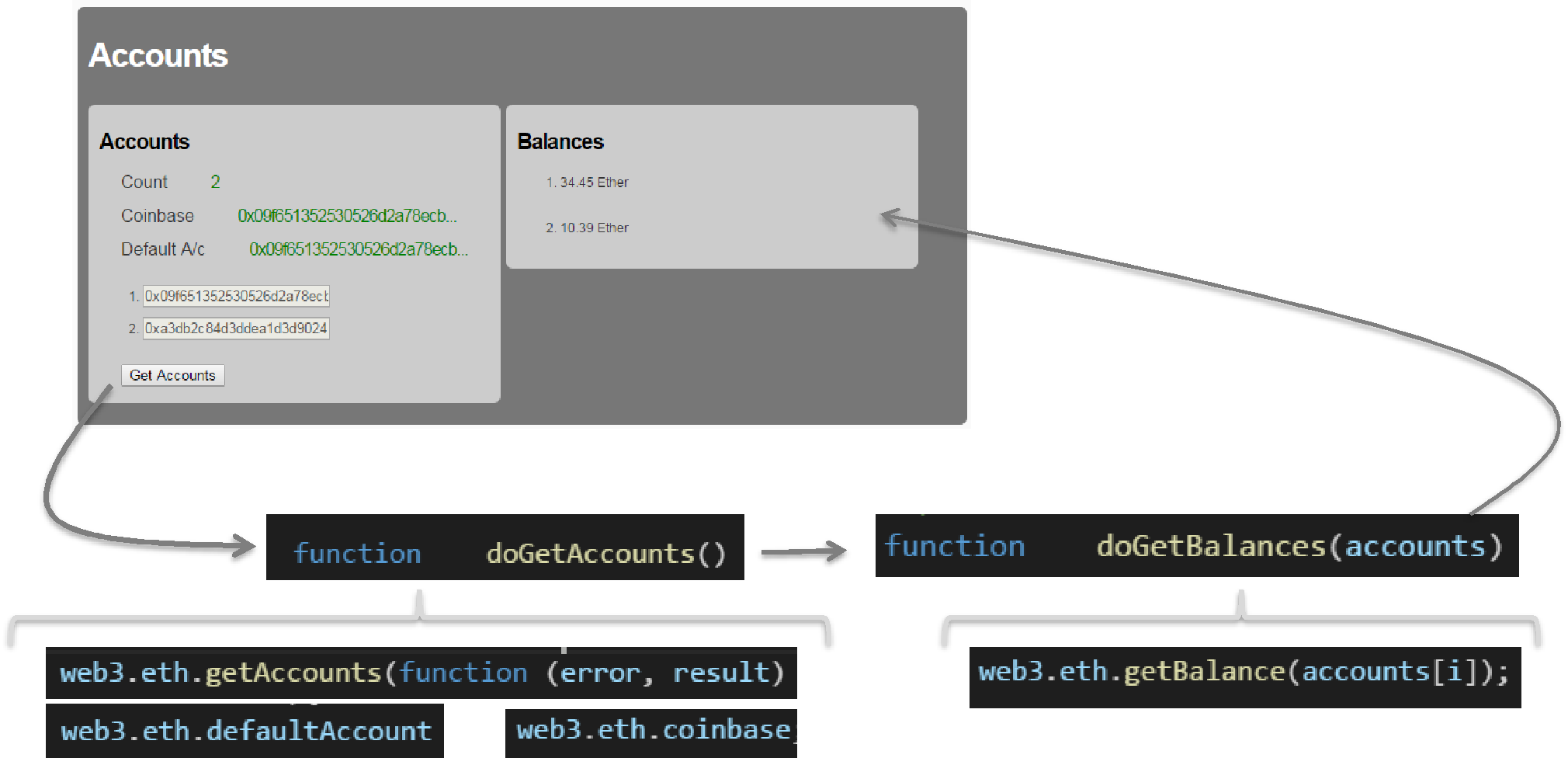
web3.net

* web3.net



listening / getListening peerCount/ getPeerCount

Get Accounts & Balances



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*

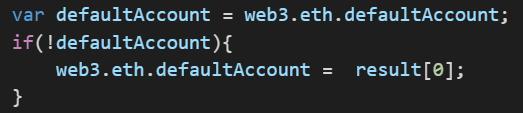
web3.eth.defaultAccount

* + Read/Write
* Used in these methods if *from:* not specified

*web3.eth.sendTransaction()*

*web3.eth.call()*

* May be undefined *(depending on implementation)*



web3.eth.getBalance

* Gets the balance for the account

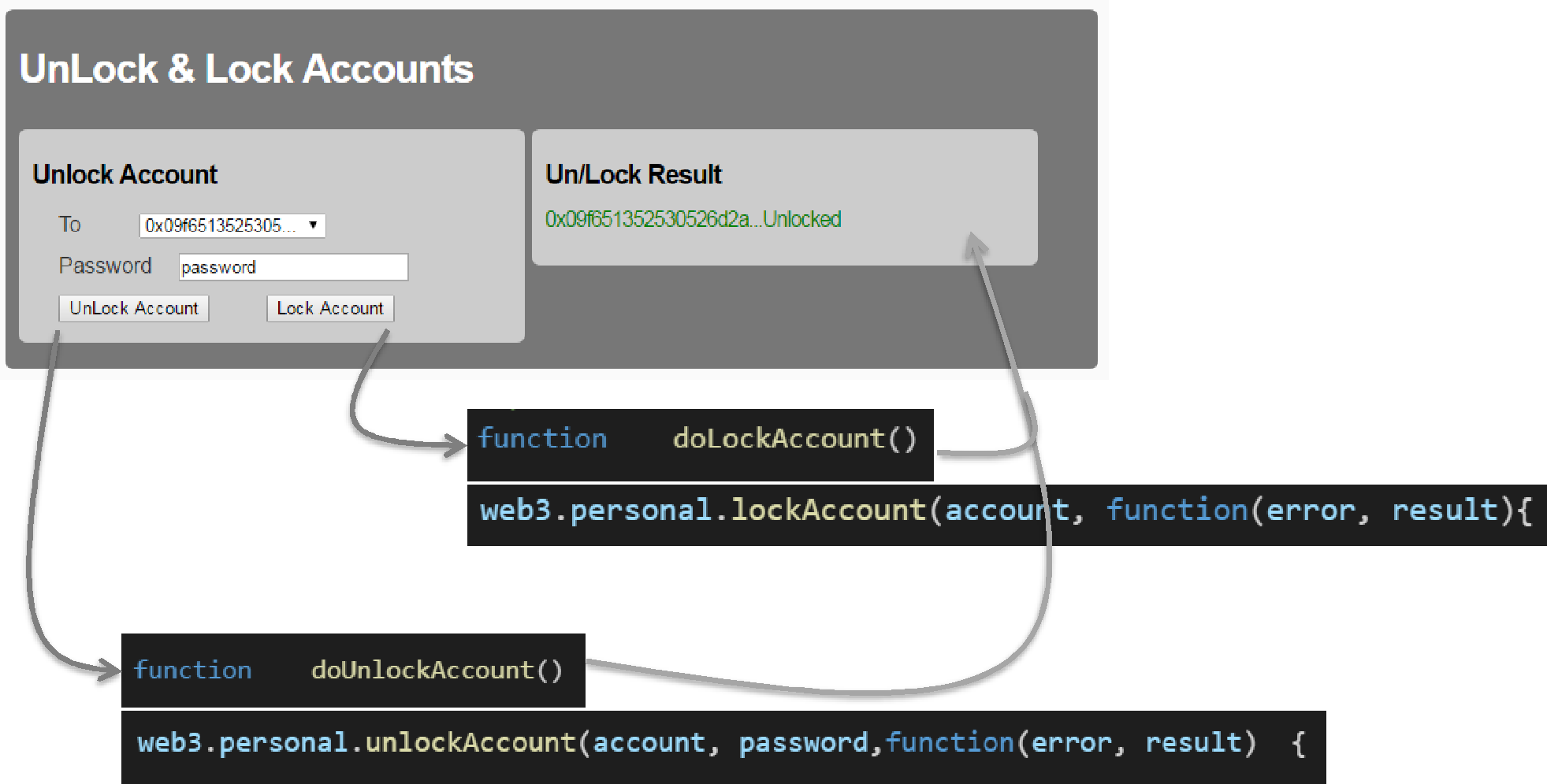
web3.eth.getBalance (address)

Result: Balance in ***wei***

var balance\_in\_ethers = web3.fromWei(balance\_in\_wei, ‘ether’)

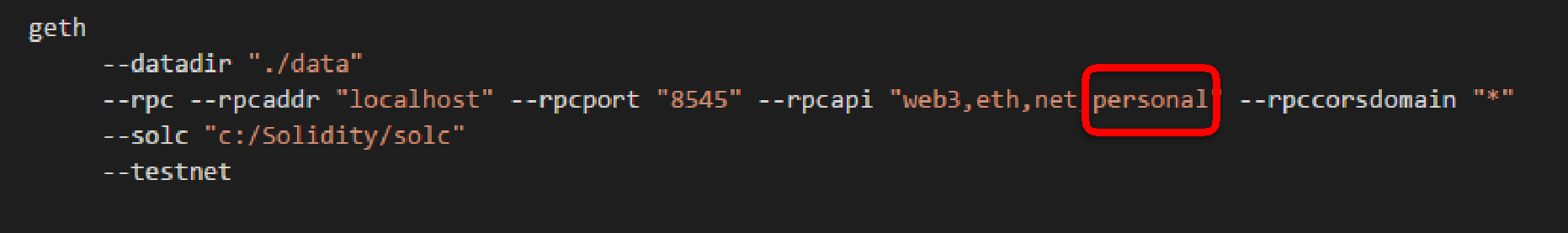
USE THE Asynchronous version as synchronous version NOT supported by MetaMask

Lock/Unlock Accounts



RPC API

* Ensure that “personal” API is enabled for RPC



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”

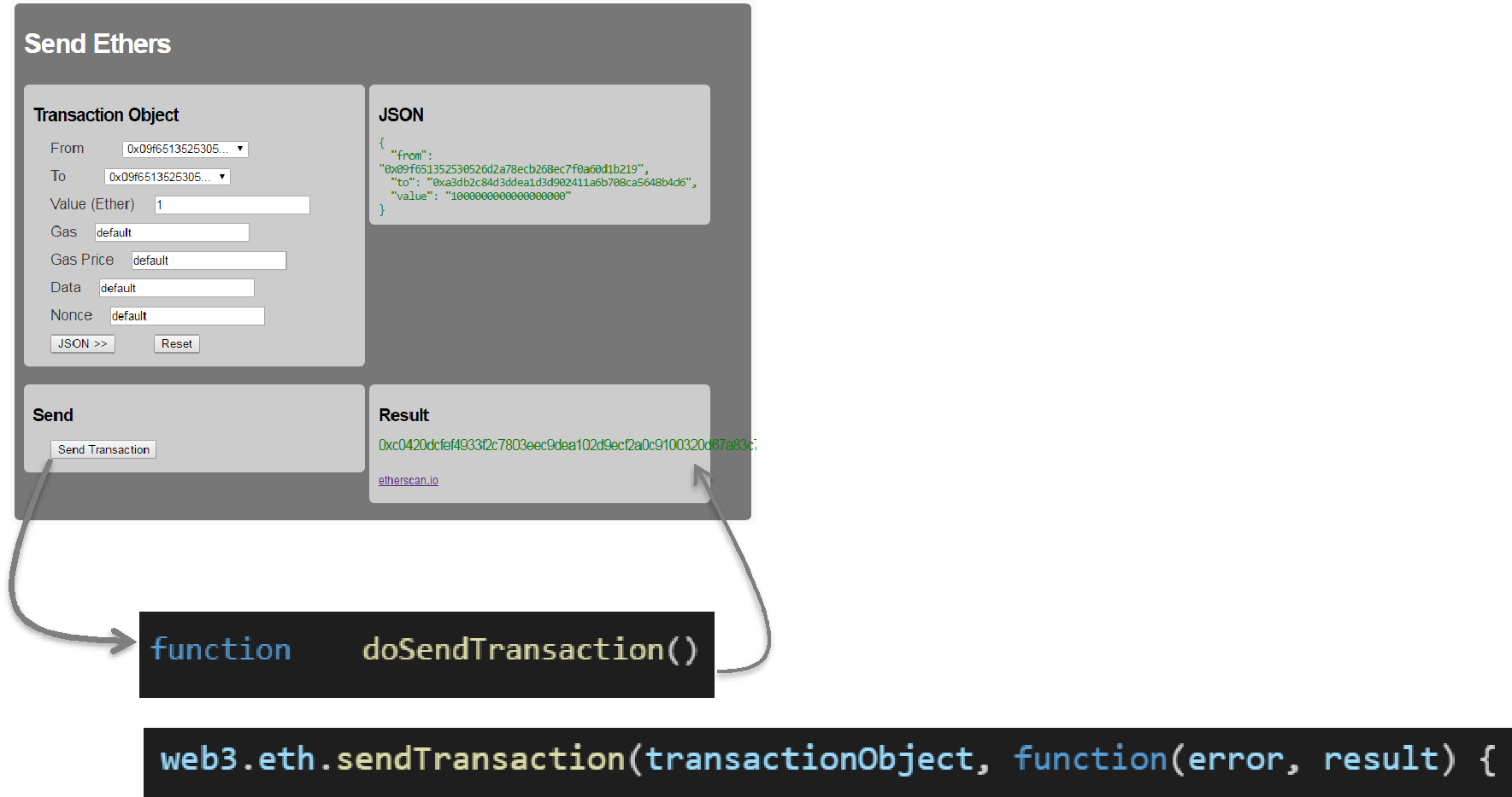
Lock Account

* Lock Account API

web3.personal.lockAccount(account)

web3.personal.lockAccount(account, callback\_func)

Send Ethers



sendTransaction



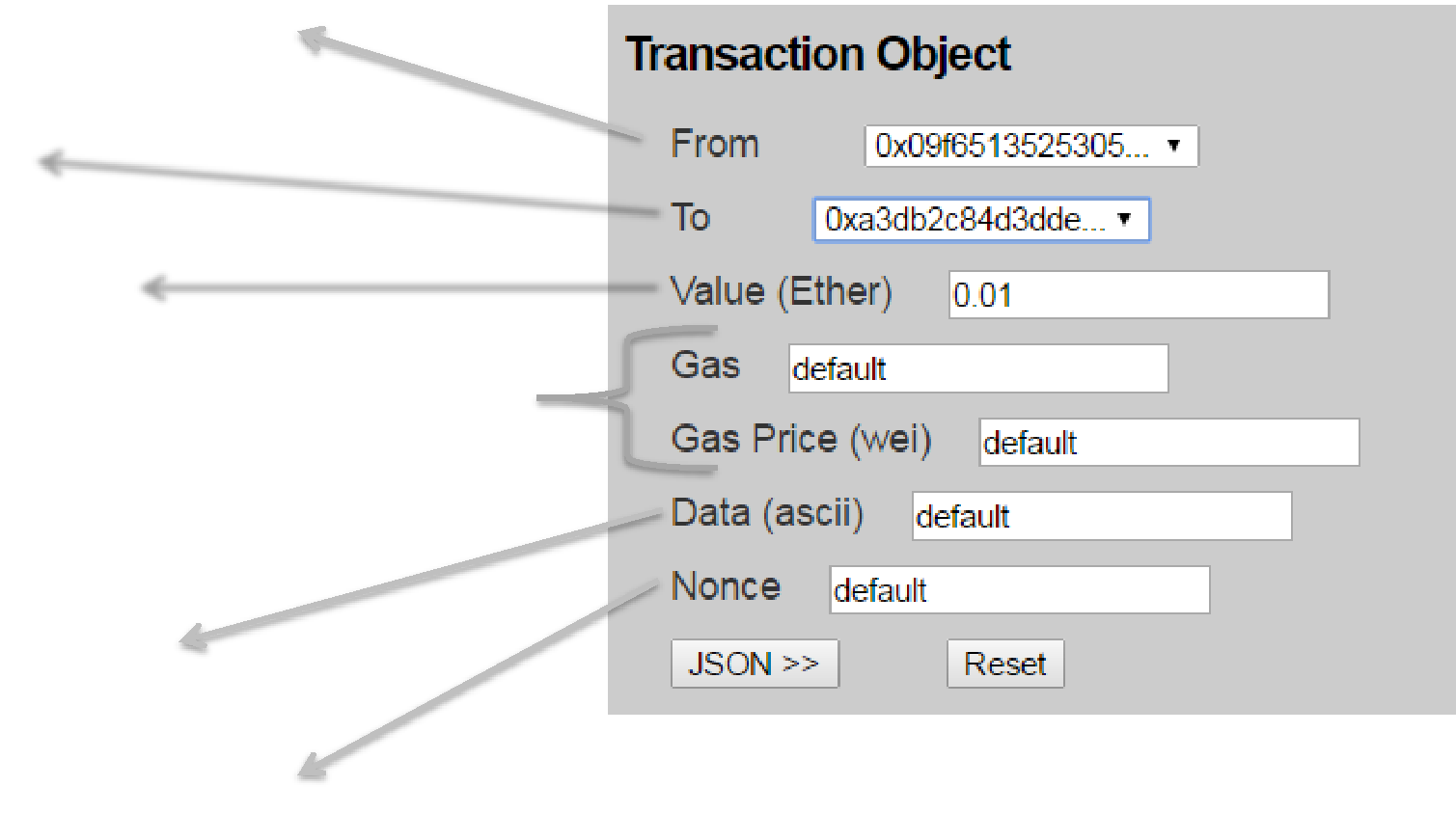
* Sending ethers
* Invoking contracts

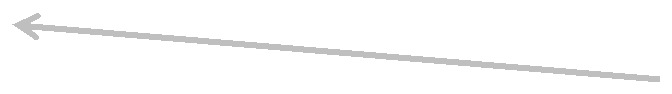
Success: result = Transaction Hash

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

Web3 JS API:

• Deployment

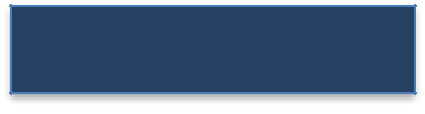
Contract deployment

* 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,

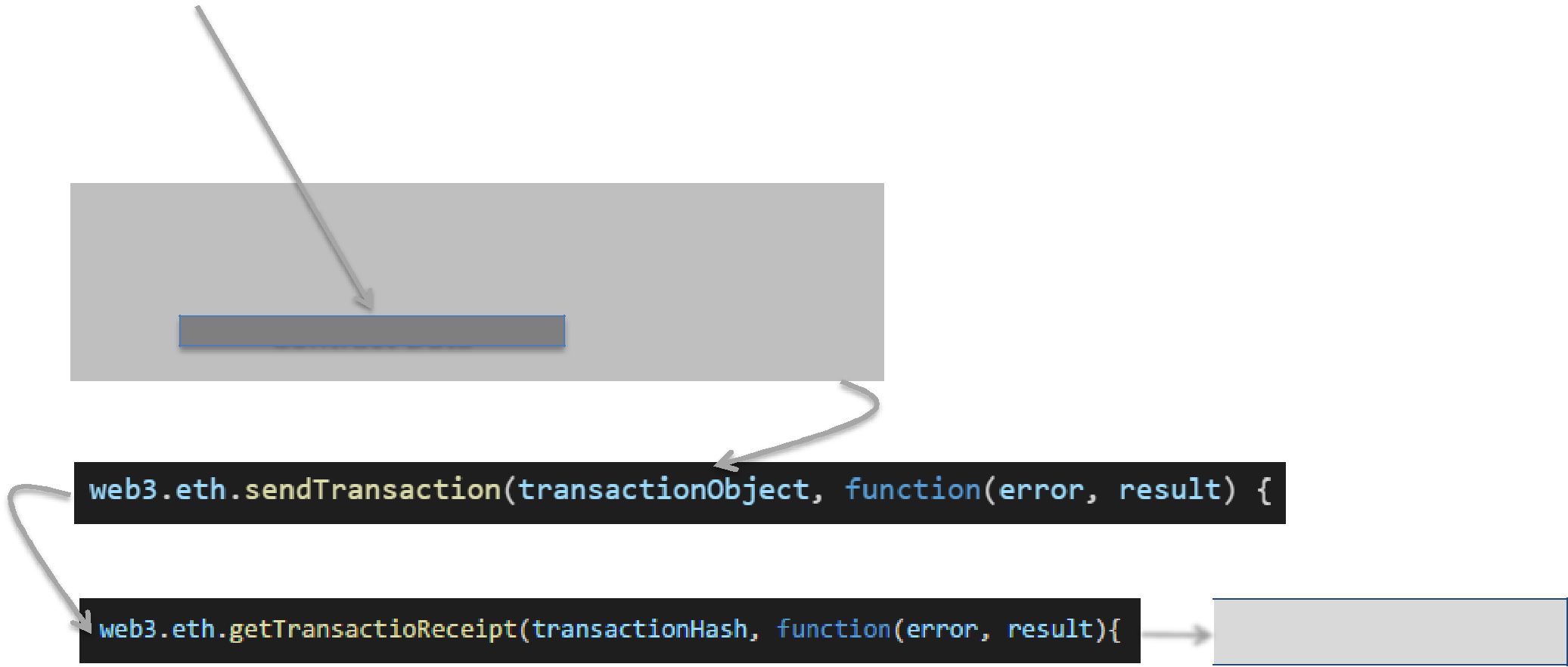
gas: gas } ,

**Callback(error, result){….}** )

* Callback function gets called 2 times in case of success
  + 1. Result = Transaction Hash
  1. Result = Contract Instance Address

Deployment using **sendTransaction**

var conData = contract.**new**.**getData**( Constructor\_Param1, Constructor\_Param2 ….,



|  |  |
| --- | --- |
| { data: bytecode } | ) |

**Transaction Object**

{

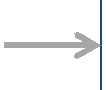
"from": "0x09f651352530526d2a78ecb268ec7f0a60d1b219",

…..,

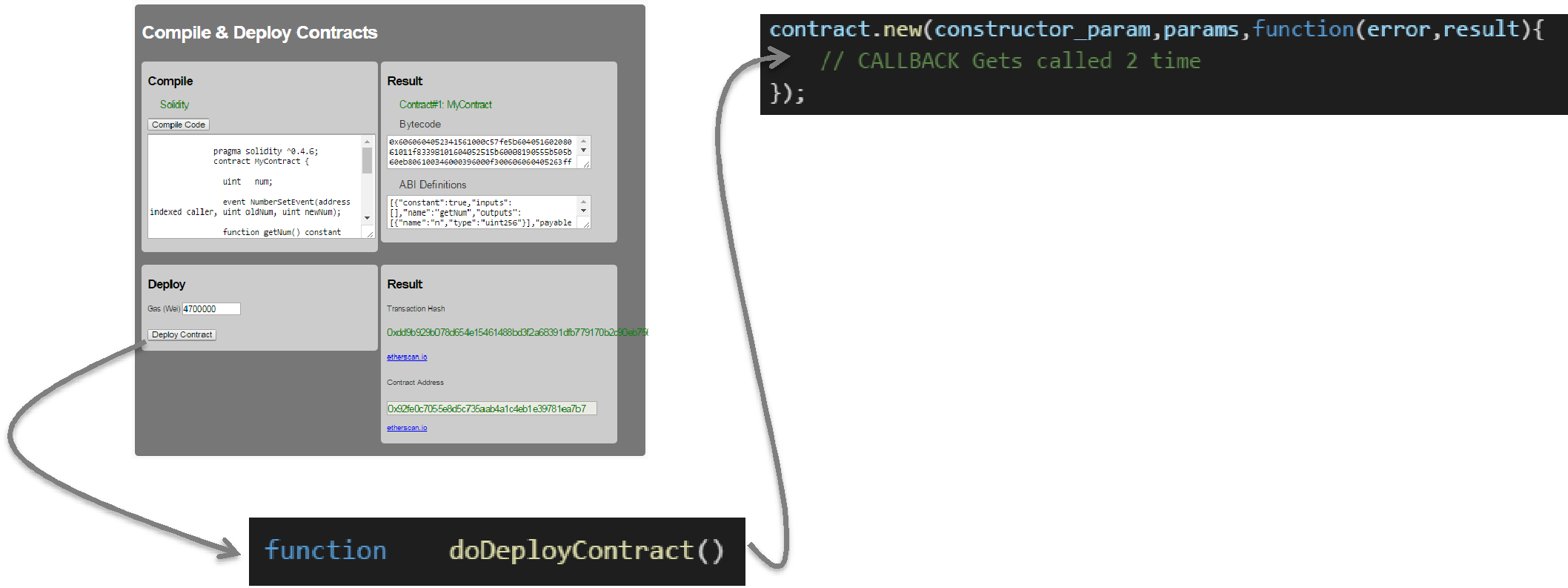
"data":

}

Contract Data

 **{Contract Address}**

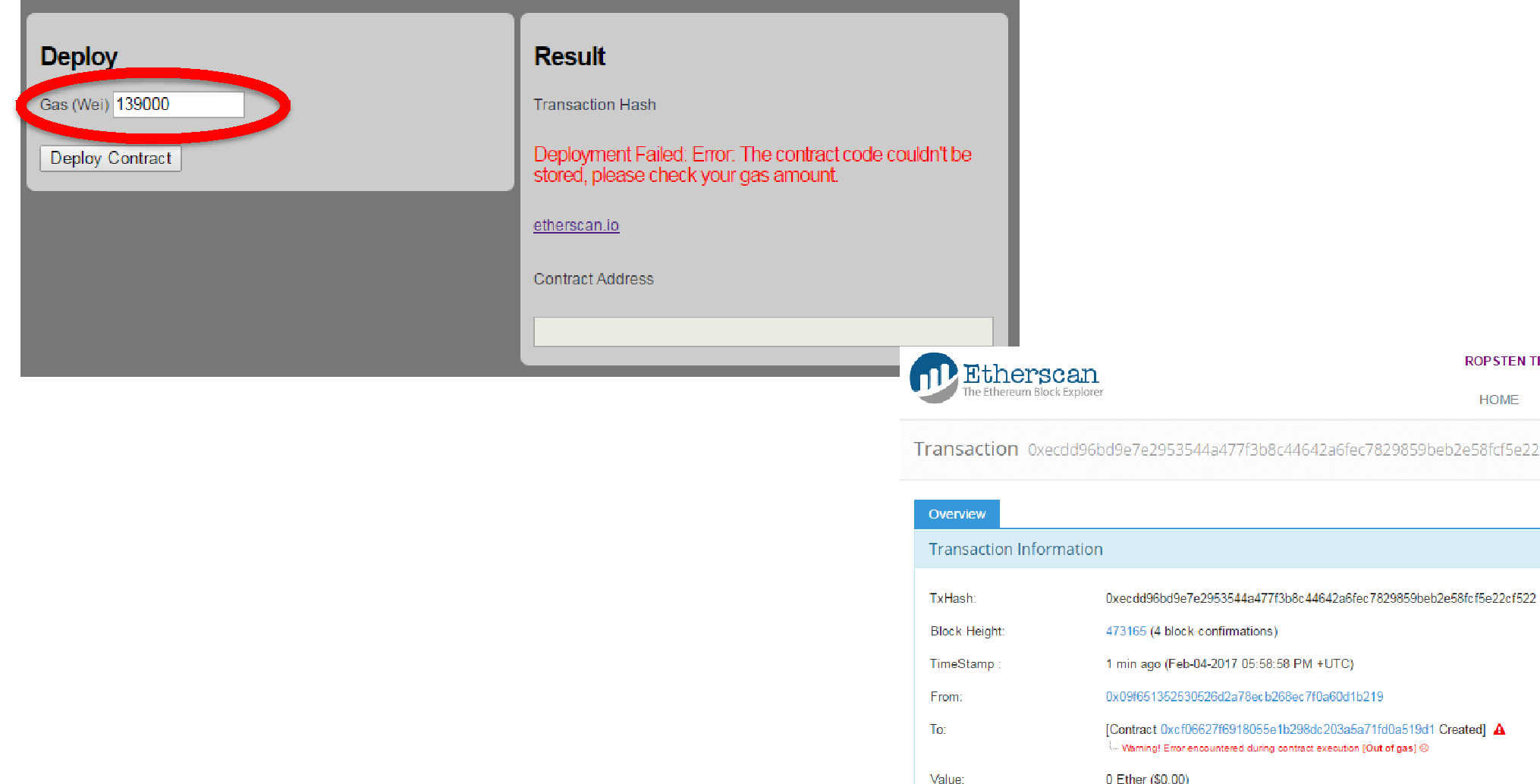
Deploy Contract



#1 result >> Transaction Hash

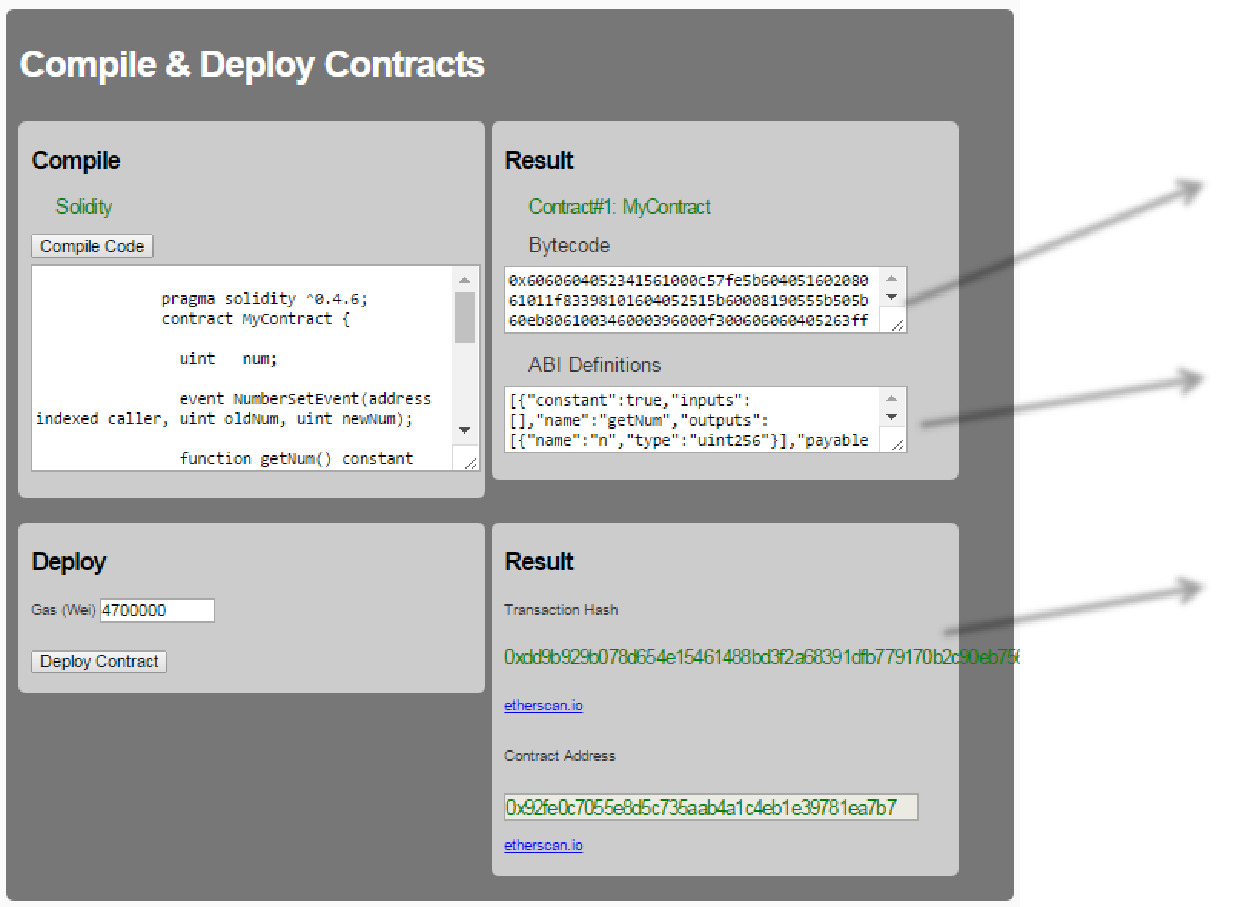
#2 result >> Contract Address

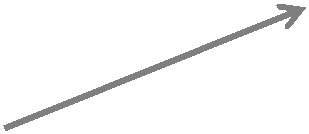
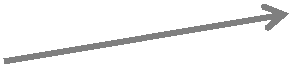
Deployment Cost

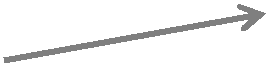


#1 result >> Transaction Hash #2 **Error**

Deploy Contract



 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(…)

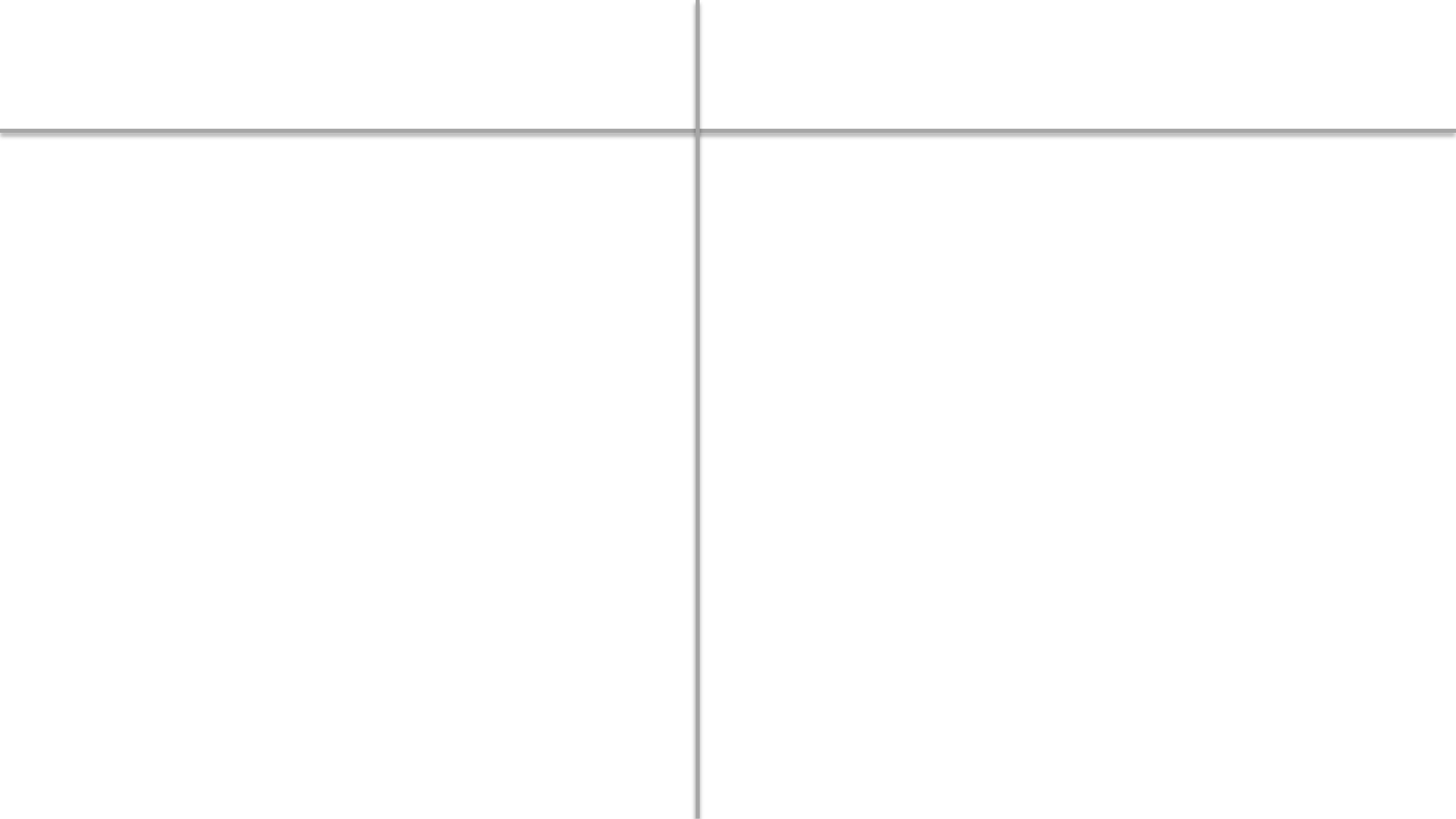
*contractInstance. Method.call(…)*

2. sendTransaction(…)

Cost of Call = 0 ETH

Cost of Send = Gas paid by caller

*contractInstance. Method.sendTransaction(…)*

*Method. call(…)* *Method. sendTransaction(…)*

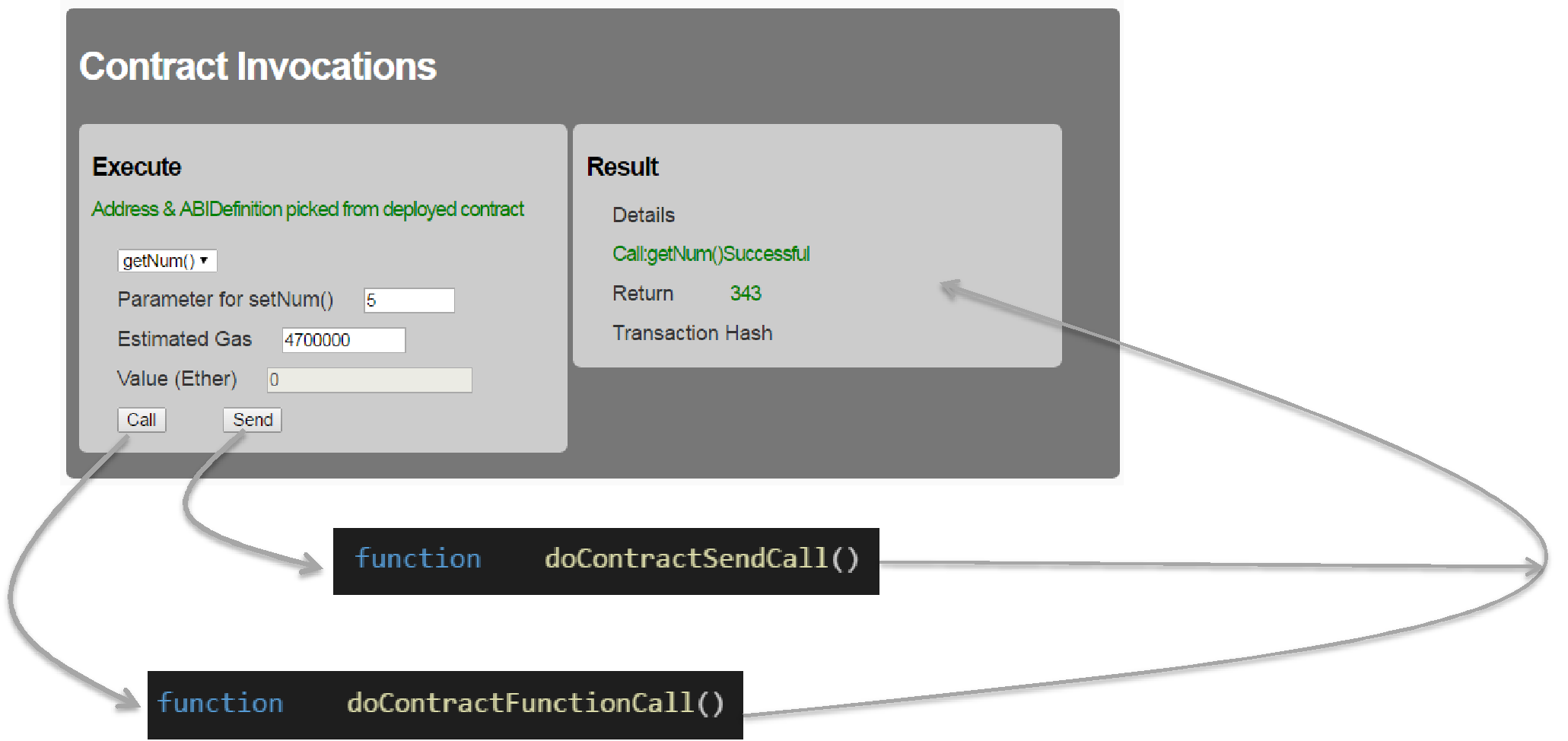
|  |  |
| --- | --- |
| • Executed locally on the node | • Executed on miner nodes |

|  |  |
| --- | --- |
| • Value= Return value from function | • Value= Transaction hash |

|  |  |
| --- | --- |
| • No state changes in contract | • State changes in contracts |

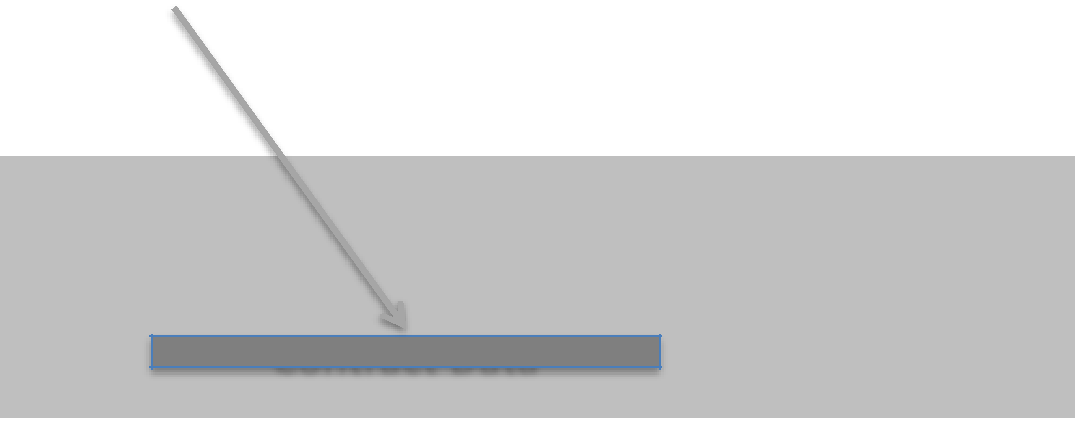
|  |  |
| --- | --- |
| • 0 execution fee | • Gas paid by caller |

Call() & sendTransaction()



web3.eth.call web3.eth.sendTransaction

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



**Transaction Object**

{

"from": "0x09f651352530526d2a78ecb268ec7f0a60d1b219", …..,

"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])*