

Web3 JS API:

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

Discount Coupon Link to UDEMY course:

<https://www.udemy.com/ethereum-dapp/?couponCode=ETHDAPP101>

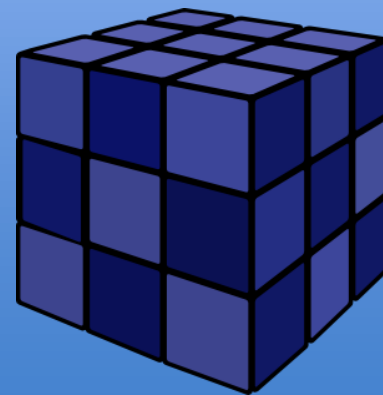
This deck is part of a online course on
[“Ethereum: Design and Development of
Decentralized Apps.”](#)

raj@acloudfan.com



@acloudfan

<http://ACloudFan.com>



- Multiple libraries available for connecting to Ethereum



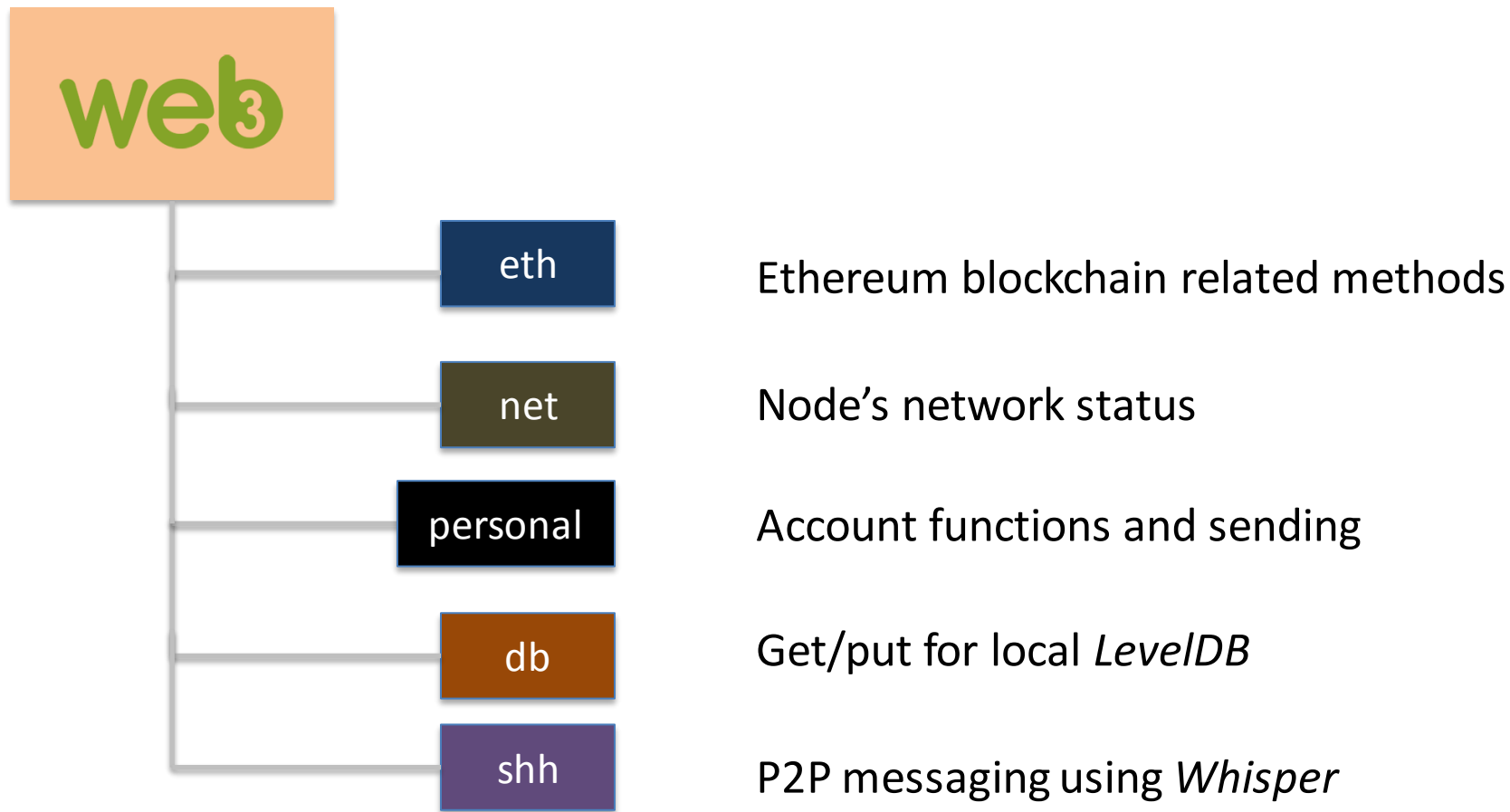
Big Numbers

- 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

Web3 API Overview

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



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
```

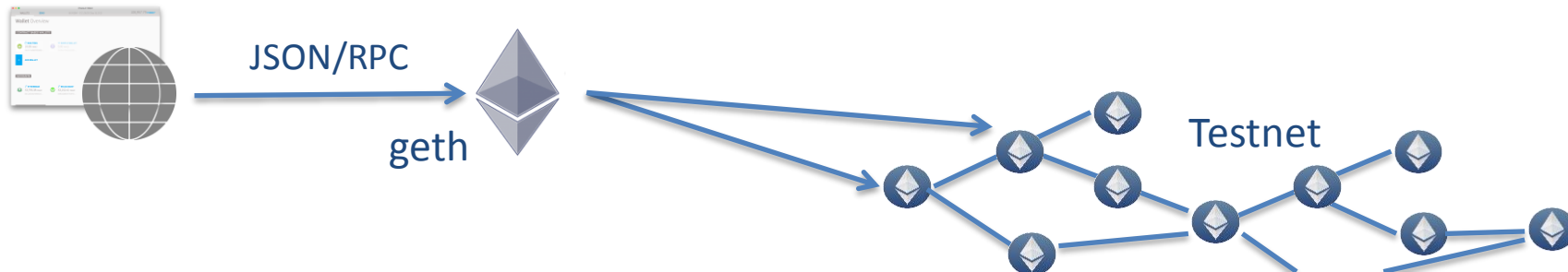
3

Install web3 library

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

Web3 Sample App: 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 Ethereum client : *geth*



Web3 Workbench

- Decentralized application
 - Will show the use of web3 API

<http://TheDapps.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

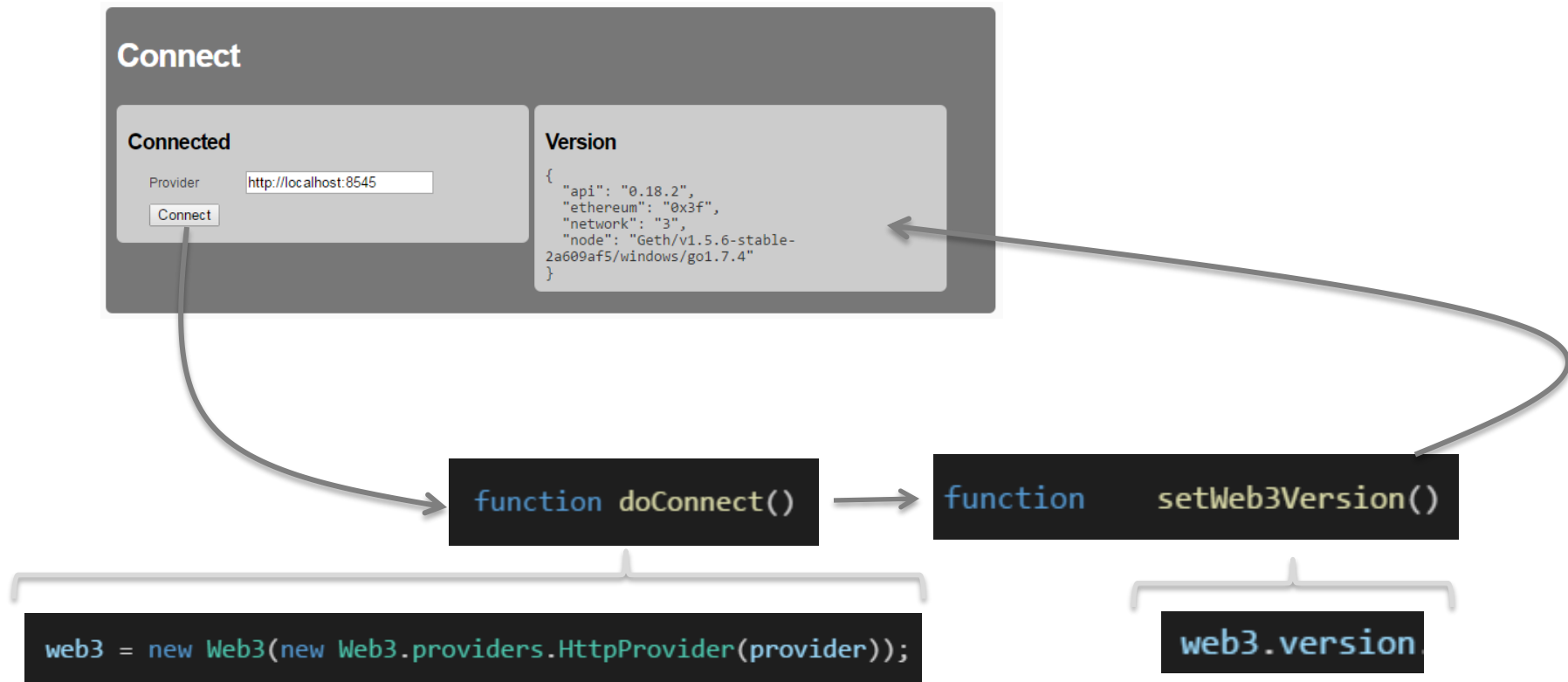
main.js

- All web3 JS API calls + some UI related code

utils.js

- UI related utility functions

Connect & Version



- web3.net

listening / getListening

peerCount / getPeerCount

```
function doGetNodeStatus()
```

Connect

Setup

Connected

Provider

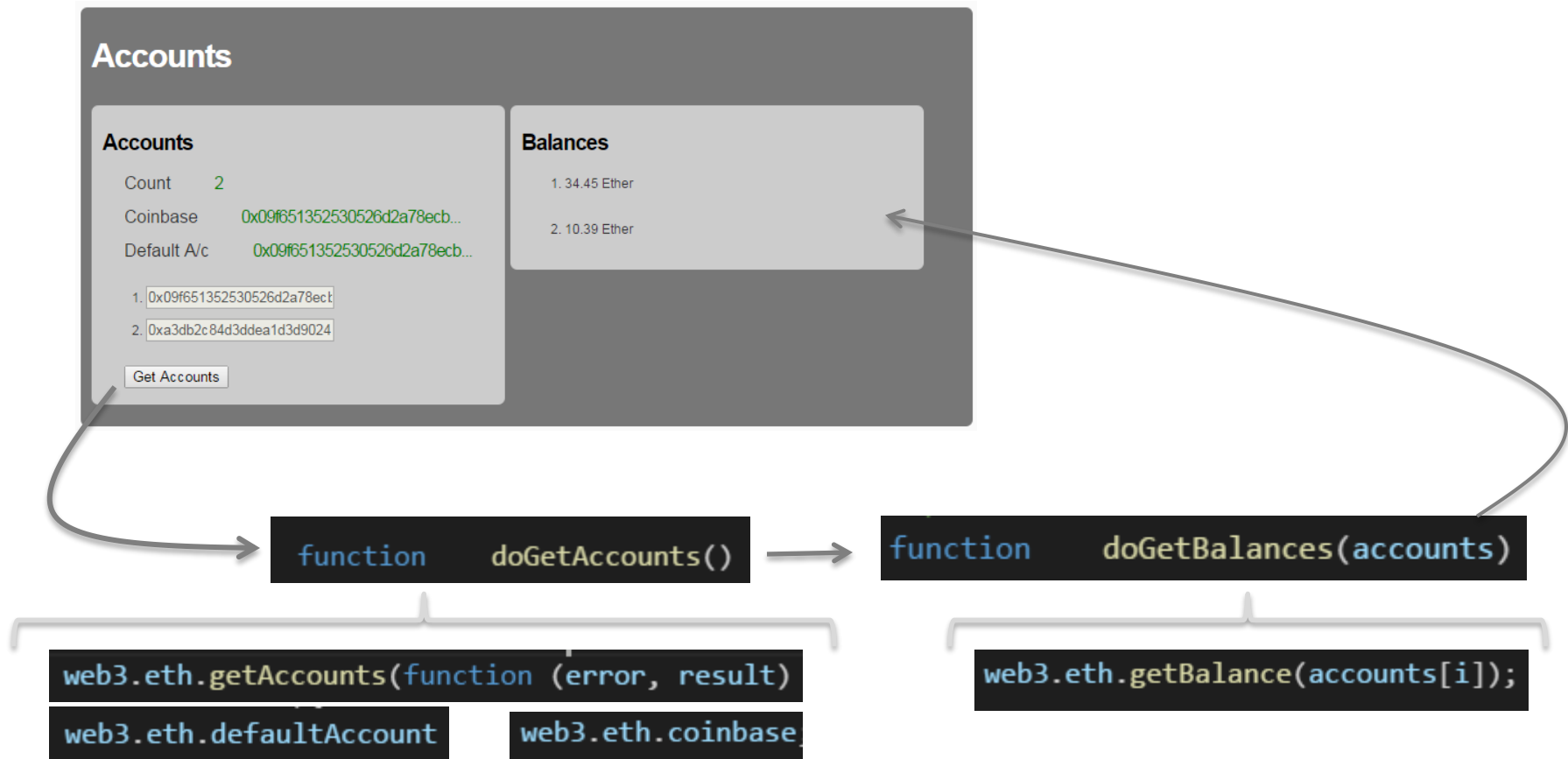
http://localhost:8545

Connect

Node Status

Peer Count: 6

Get Accounts & Balances



- 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

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

The screenshot displays the Etherscan.io interface for a transaction. The top navigation bar includes links for Home, Blocks, Transactions, and Tokens. The main content area is divided into two columns. The left column, titled "Transaction Object", contains fields for From, To, Value (Ether), Gas, Gas Price, Data, and Nonce, each with a default value or a dropdown menu. Below these fields are buttons for "JSON >>" and "Reset". The right column, titled "JSON", displays the transaction data in a JSON format. The "Send" section at the bottom left contains a "Send Transaction" button. The "Result" section at the bottom right displays the transaction hash and a link to the transaction details on Etherscan.io.

Send Ethers

Transaction Object

From

To

Value (Ether)

Gas

Gas Price

Data

Nonce

JSON

```
{
  "from":
    "0x09f651352530526d2a78ecb268ec7f0a60d1b219",
  "to": "0xa3db2c84d3ddda1d3d902411a6b708ca5648b4d6",
  "value": "1000000000000000000"
}
```

Send

Result

[0xc0420dcfe4933f2c7803eec9dea102d9ec12a0c9100320d67a83c...](#)

[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

$\text{Fee} = \text{gas} * \text{gasPrice}$

Data | Contract call
In Hex

Overwrite pending

Transaction Object

From

To

Value (Ether)

Gas

Gas Price (wei)

Data (ascii)

Nonce

JSON >>

Reset

Web3 JS API:

- Deployment

Discount Coupon Link to UDEMY course:

<https://www.udemy.com/ethereum-dapp/?couponCode=ETHDAPP101>

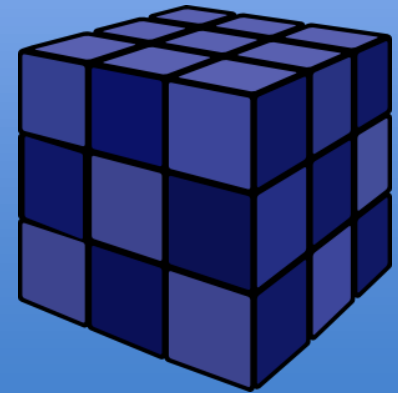
This deck is part of a online course on
“Ethereum: Design and Development of
Decentralized Apps.

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.getTransactionReceipt(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#: MyContract
Bytecode
0x6060604052341561000c57fe5b60405160200061011f8339810160405215b600008190555b505b60eb806100346000396000f300606060405263ff

ABI Definitions

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

Deploy
Gas (Wei) 4700000
Deploy Contract

Result
Transaction Hash
0xd19b929b078d654e15461488bd3f2a68391dfb779170b2c90eb754
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)

Result

Transaction Hash

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

etherscan.io

Contract Address

#1 result >> Transaction Hash

#2 **Error**






ROPSTEN TI

[HOME](#)

Transaction [0xecdd96bd9e7e2953544a477f3b8c44642a6fec7829859beb2e58fcf5e22](#)

Overview

Transaction Information

TxHash:	0xecdd96bd9e7e2953544a477f3b8c44642a6fec7829859beb2e58fcf5e22c1f522
Block Height:	473165 (4 block confirmations)
TimeStamp :	1 min ago (Feb-04-2017 05:58 PM +UTC)
From:	0x09f651352530526d2a78ecb268ec7f0a60d1b219
To:	[Contract 0xc06627f6918055e1b298dc203a5a71fd0a519d1 Created] 
	 Warning! Error encountered during contract execution [Out of gas] 
Value:	0 Ether (\$0.00)

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

```
0x606060604052341561000c57fe5b604051602080
61011f83398101604052515b60008190555b505b
60eb806100346000396000f300606060405263ff
```

ABI Definitions

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

Deploy

Gas (Wei) 4700000

Deploy Contract

Result

Transaction Hash

0xdd9b929b078d654e15461488bd3f2a68391dfb779170b2c80eb754

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 Link to UDEMY course:

<https://www.udemy.com/ethereum-dapp/?couponCode=ETHDAPP101>

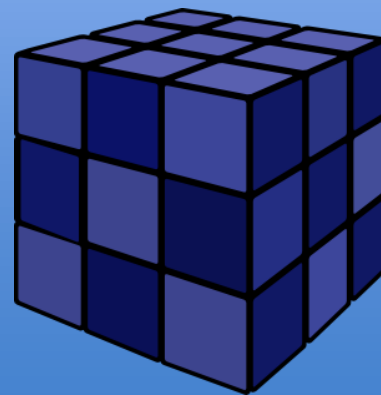
This deck is part of a online course on
“Ethereum: Design and Development of
Decentralized Apps.

raj@acloudfan.com



@acloudfan

<http://ACloudFan.com>



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(...)

- Executed **locally** on the node
- Value= Return value from function
- No **state changes** in contract
- **0** execution fee

Method.sendTransaction(...)

- Executed on **miner** nodes
- Value= **Transaction hash**
- **State changes** in contracts
- Gas **paid** by caller

Call() & sendTransaction()

Contract Invocations

Execute

Address & ABI Definition picked from deployed contract

getNum() ▾

Parameter for setNum()

Estimated Gas

Value (Ether)

Call

Send

Result

Details

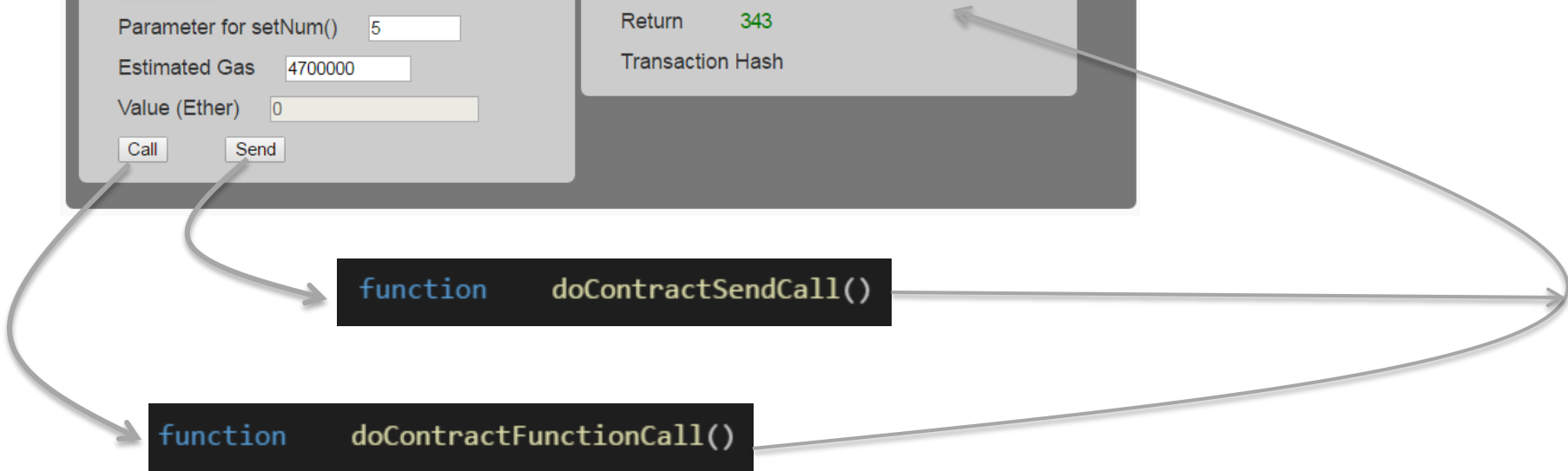
Call: getNum() Successful

Return 343

Transaction Hash

`function doContractSendCall()`

`function doContractFunctionCall()`




web3.eth.call web3.eth.sendTransaction

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

Transaction Object

```
{  
  "from": "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])*