

Ethereum Basics

CS61065 Theory and Applications of Blockchain

Ethereum

What is Ethereum?

Ethereum is a technology that's home to digital money, global payments, and applications. The community has built a booming digital economy, bold new ways for creators to earn online, and so much more. It's open to everyone, wherever you are in the world – all you need is the internet.

Source: <https://ethereum.org/>

What is Ethereum?

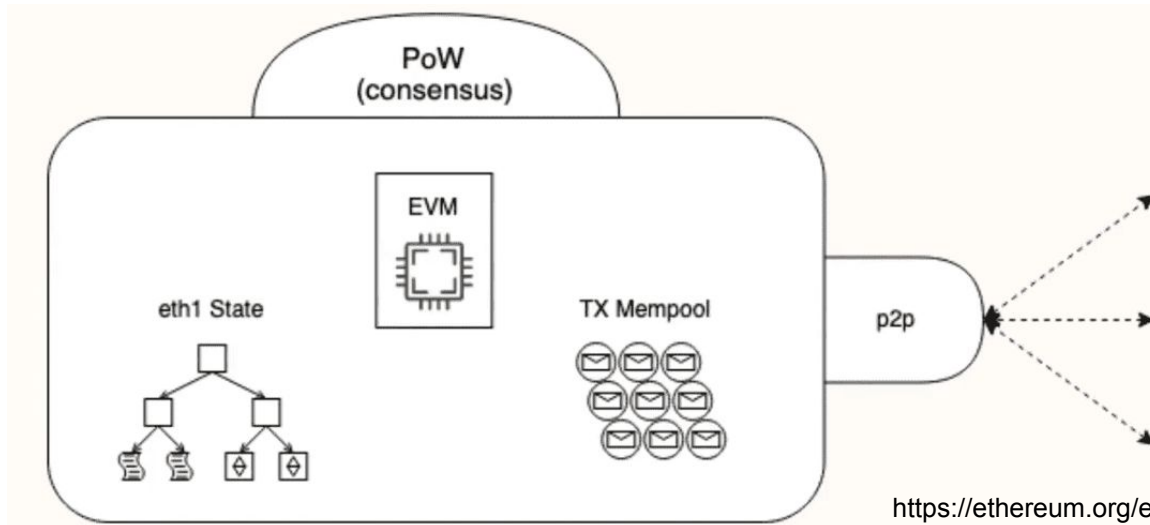
Ethereum is a **decentralized platform that runs smart contracts**, applications that run exactly as programmed without possibility of downtime, censorship, fraud or third party interference.

Source: <https://github.com/ethereum/go-ethereum>

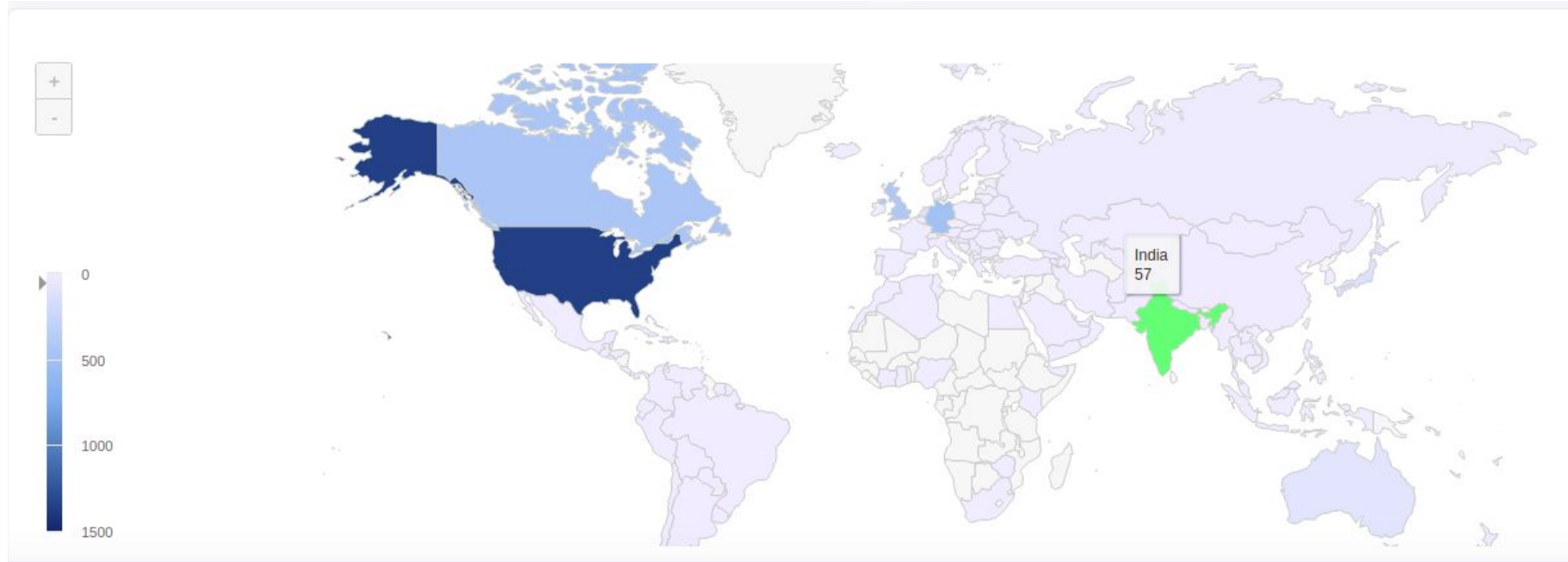
Ethereum Network

Distributed network of computers running software (known as nodes) that can verify blocks and transaction data.

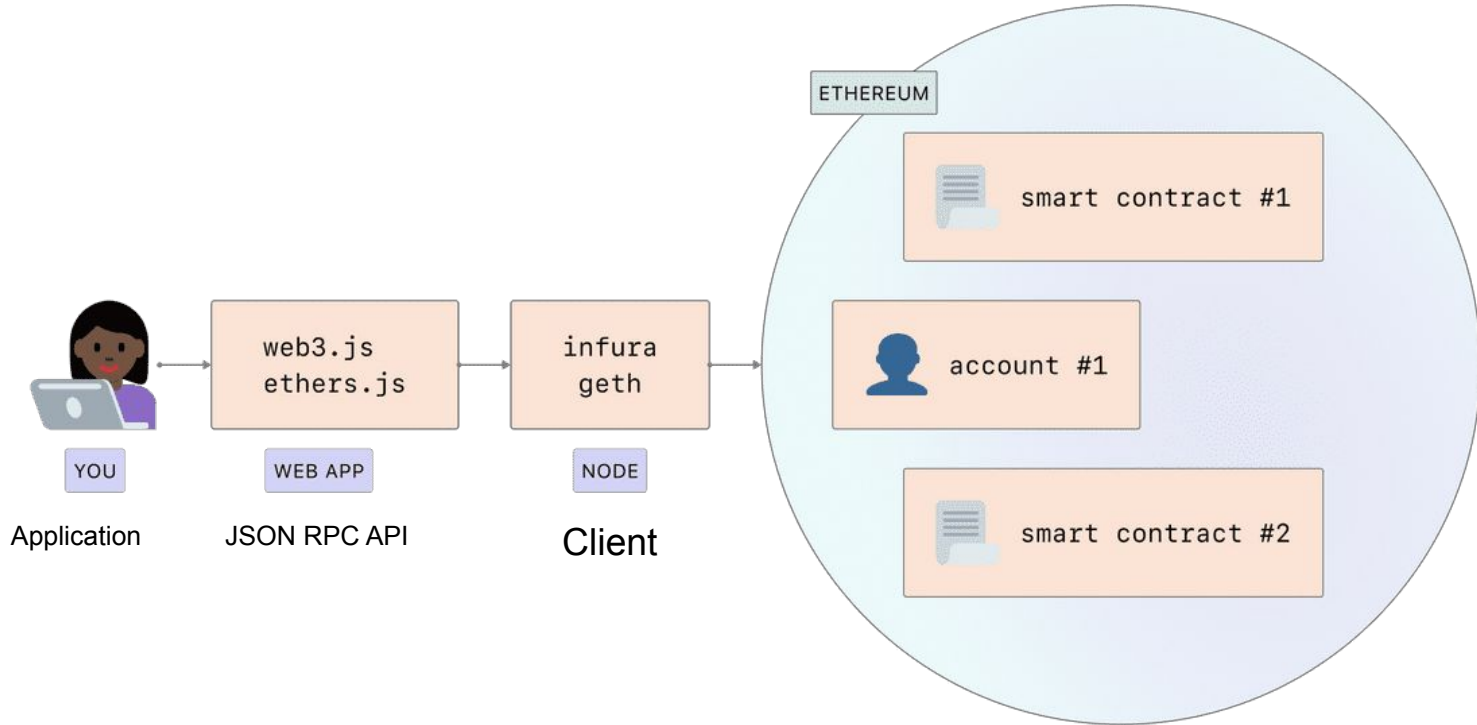
An application, known as a **client**, running on your computer is a node.



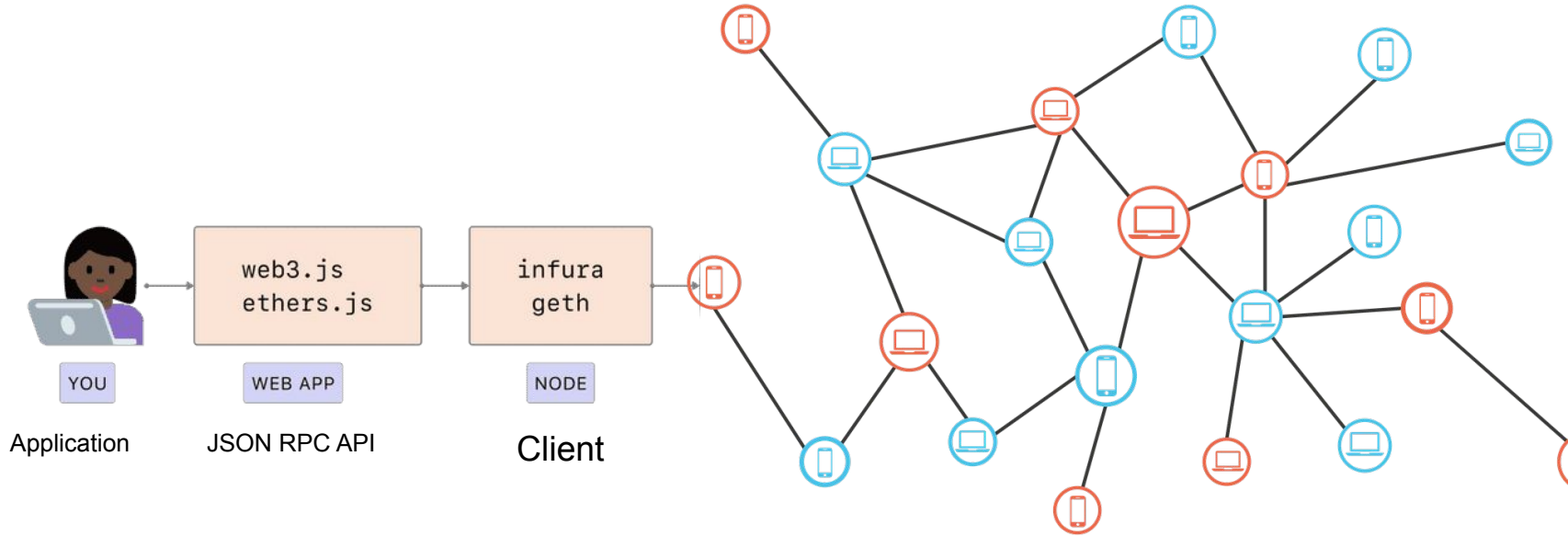
No servers, nodes connected in P2P network.



How to interact with the Ethereum Network?



How to interact with the Ethereum Network?



Geth

Official Go implementation of the Ethereum protocol

- Main Ethereum CLI client
- Entry point into the Ethereum network
 - main, test, or private net
- Capable of running as:
 - Full node (default)
 - Archive node (retaining all historical state)
 - Light node (retrieving data live).
- Provides JSON RPC endpoints exposed on top of HTTP, WebSocket and/or IPC transports.

Installing Geth

For Ubuntu

To enable our launchpad repository run:

```
sudo add-apt-repository -y ppa:ethereum/ethereum
```

Then install the stable version of go-ethereum:

```
sudo apt-get update
```

```
sudo apt-get install ethereum
```

For other platforms follow the instructions :

<https://geth.ethereum.org/docs/install-and-build/installing-geth>

Managing Accounts

USAGE:

```
geth account command [command options] [arguments...]
```

COMMANDS:

<code>list</code>	Print summary of existing accounts
<code>new</code>	Create a new account
<code>update</code>	Update an existing account
<code>import</code>	Import a private key into a new account

Create an account

```
geth account new
```

```
Repeat password:
```

```
Your new key was generated
```

```
Public address of the key: 0x8AF72Ec4f53704FffF24737f0445ddB40483eebd1
```

```
Path of the secret key file: /home/bishakh/.ethereum/keystore/UTC--2021-09-01T19-16-19.04670487
```

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!

List accounts

```
geth account list
```

```
~ ➤ geth account list
INFO [09-02|00:48:28.106] Maximum peer count                ETH=50
LES=0 total=50
INFO [09-02|00:48:28.106] Smartcard socket not found, disabling err="st
at /run/pcscd/pcscd.comm: no such file or directory"
INFO [09-02|00:48:28.106] Set global gas cap                cap=50,
000,000
Account #0: {8af72ec4f53704fff24737f0445ddb40483eebd1} keystore:///home/bi
shakh/.ethereum/keystore/UTC--2021-09-01T19-16-19.046704871Z--8af72ec4f537
04fff24737f0445ddb40483eebd1
```

Note the keystore is /home/<user>/.ethereum/keystore

View account key

The key is encrypted using a passphrase

```
cat ~/.ethereum/keystore/<key file>
```

```
~ ➔ cat ~/.ethereum/keystore/UTC--2021-09-01T19:16:19.046704871Z--8af72ec4f53704fff24737f0445ddb40483eebd1
{"address":"8af72ec4f53704fff24737f0445ddb40483eebd1","crypto":{"cipher":"aes-128-ctr","ciphertext":"178a301ac272b11b3959e634dc464bcb6bde49263829f4d4d2fd1671ac602cf3","cipherparams":{"iv":"e90eac9df577270ada8af670829b40eb"},"kdf":"scrypt","kdfparams":{"dklen":32,"n":262144,"p":1,"r":8,"salt":"5efa1904b69e900ed6521b22ae93bb021fd3b8052c83dd4e5ed749ead33cf6c4"},"mac":"b88db7597cddd7572835f3676c22fbb80b044cd4752822f0303f4e92a29f62b2"},"id":"48a27b2f-f244-4845-8980-a5ed57ab51fe","version":3}%
```

Connecting to the network

- **Starting geth without any flag connects to the Ethereum mainnet.**
- In addition to the mainnet, geth recognizes a few testnets which you can connect to via the respective flags:
 - --ropsten, **Ropsten** proof-of-work test network <https://ropsten.etherscan.io/>
 - --rinkeby, **Rinkeby** proof-of-authority test network <https://rinkeby.etherscan.io/>
 - --goerli, **Goerli** proof-of-authority test network

Sync modes

You can start Geth in one of three different sync modes using the `--syncmode "<mode>"` argument that determines what sort of node it is in the network.

These are:

- **full**: Downloads all blocks (including headers, transactions, and receipts) and generates the state of the blockchain incrementally by executing every block.
- **fast**: Downloads all blocks (including headers, transactions and receipts), verifies all headers, and downloads the state and verifies it against the headers.
- **snap** (Default): Same functionality as fast, but with a faster algorithm.
- **light**: Downloads all block headers, block data, and verifies some randomly.

Connecting to rinkeby testnet

```
geth --rinkeby --syncmode "light" --http
```

```
~ geth --rinkeby --syncmode "light" --http
INFO [09-02|01:19:00.802] Starting Geth on Rinkeby testnet...
INFO [09-02|01:19:00.803] Dropping default light client cache
INFO [09-02|01:19:00.804] Maximum peer count
INFO [09-02|01:19:00.804] Smartcard socket not found, disabling
or directory"
INFO [09-02|01:19:00.804] Set global gas cap
INFO [09-02|01:19:00.805] Allocated cache and file handles
/lightchaindata cache=64.00MiB handles=2048
INFO [09-02|01:19:00.854] Allocated cache and file handles
/les.client cache=16.00MiB handles=16
INFO [09-02|01:19:00.893] Persisted trie from memory database
des=0 gcsiz=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [09-02|01:19:00.893] Initialised chain configuration
AOSupport: true EIP150: 2 EIP155: 3 EIP158: 3 Byzantium: 1035301 Constantinople: 3660663 Petersburg: 4321234 Istanbul: 5435345, Muir Glacier: <nil>, Berlin: 8290928, London: 8897988, Engine: clique}"
INFO [09-02|01:19:00.894] Added trusted checkpoint
INFO [09-02|01:19:00.894] Loaded most recent local header
8,730 age=15s

provided=1024 updated=128
ETH=0 LES=10 total=50
err="stat /run/pcscd/pcscd.comm: no such file
or directory"
cap=50,000,000
database=/home/bishakh/.ethereum/rinkeby/geth
database=/home/bishakh/.ethereum/rinkeby/geth
nodes=355 size=50.43KiB time="825.171µs" gcno=1
config="{ChainID: 4 Homestead: 1 DAO: <nil> D
block=8,749,055 hash=f5655c..5c8688
number=9,218,305 hash=909774..6e257f td=15,62
```

Interacting with Geth

You can interact with Geth in two ways:

- Directly with the node using the JavaScript console over IPC or
- Connecting to the node remotely over HTTP using RPC.

IPC allows you to do more, especially when it comes to creating and interacting with accounts, but you need direct access to the node.

RPC allows remote applications to access your node but has limitations and security considerations.

Using RPC over HTTP

Querying the balance of an account:

POST request to the HTTP endpoint (default: 127.0.0.1:8545)

JSON Payload:

```
{
  "jsonrpc": "2.0",
  "method": "eth_getBalance",
  "params": [
    "0x9808f22453Ee87cc23eA76ca7Ed66a4F294d54D4",
    "latest"
  ],
  "id": 1
}
```

Using RPC over HTTP

```
curl -X POST http://localhost:8545 \  
  -H "Content-Type: application/json" \  
  --data '{"jsonrpc":"2.0", "method":"eth_getBalance",  
"params":["0x9808f22453Ee87cc23eA76ca7Ed66a4F294d54D4","latest"], "id":1}'
```

```
~ curl -X POST http://localhost:8545 \  
  -H "Content-Type: application/json" \  
  --data '{"jsonrpc":"2.0", "method":"eth_getBalance", "params":["0x9808f22453Ee87cc23eA76c  
a7Ed66a4F294d54D4","latest"], "id":1}'  
  
{"jsonrpc":"2.0", "id":1, "result": "0x102c9030853089bed"}
```

18 647439066391944173

→ 18 decimal places

Query Accounts

NOTE: create an account in the your chosen testnet before proceeding:

```
geth --rinkeby account new
```

Alternatively you can copy your existing key:

```
cp .ethereum/keystore/<keyfile> .ethereum/rinkeby/keystore
```

Query Accounts:

```
curl -X POST http://localhost:8545 \  
  -H "Content-Type: application/json" \  
  --data '{"jsonrpc":"2.0", "method":"eth_accounts", "id":0}'
```

```
~ curl -X POST http://localhost:8545 \  
  -H "Content-Type: application/json" \  
  --data '{"jsonrpc":"2.0", "method":"eth_accounts", "id":0}'  
{ "jsonrpc": "2.0", "id": 0, "result": ["0x8af72ec4f53704ffff24737f0445ddb40483eebd1"] }
```

Execute Transactions

```
curl -X POST http://localhost:8545 \  
  -H "Content-Type: application/json" \  
  --data '{"jsonrpc":"2.0", "method":"eth_sendTransaction", "params":[{"from":  
"0x8af72ec4f53704ffff24737f0445ddb40483eebd1","to":  
"0x35F18427567108F800BDC2784277B9246eED37fA","value": "0xDE0B6B3A7640000"}], "id":0}'
```

```
{  
  "jsonrpc": "2.0",  
  "method": "eth_sendTransaction",  
  "params": [  
    {  
      "from": "0x8af72ec4f53704ffff24737f0445ddb40483eebd1",  
      "to": "0x35F18427567108F800BDC2784277B9246eED37fA",  
      "value": "0xDE0B6B3A7640000"  
    }  
  ],  
  "id": 0  
}
```

Execute Transactions

```
curl -X POST http://localhost:8545 \  
  -H "Content-Type: application/json" \  
  --data '{"jsonrpc":"2.0", "method":"eth_sendTransaction", "params":[{"from":  
"0x8af72ec4f53704fff24737f0445ddb40483eebd1", "to":  
"0x35F18427567108F800BDC2784277B9246eED37fA", "value": "0xDE0B6B3A7640000"}], "id":0}'
```

```
{"jsonrpc":"2.0","id":0,"error":{"code":-32000,"message":"err: insufficient funds for gas *  
price + value: address 0x8AF72Ec4f53704FfF24737f0445ddB40483eebd1 have 0 want 14995837239933  
392 (supplied gas 14995837)"} }
```

Use testnet faucet to get funds.

Get Eth balance from faucet

<https://faucet.rinkeby.io/>


<https://faucet.goerli.mudit.blog/>



Rinkeby Authenticated Faucet


Social network URL containing your Ethereum address...

Give me Ether ▼

 peers

 blocks

 Ethers

 funded

Unlock Account

```
~ curl -X POST http://localhost:8545 \
-H "Content-Type: application/json" \
--data '{"jsonrpc":"2.0", "method":"eth_sendTransaction", "params":[{"from": "0x8af72ec4f53704fff24737f0445ddb40483eebd1", "to": "0x35F18427567108F800BDC2784277B9246eED37fA", "value": "0xDE0B6B3A7640000"}], "id":0}'

{"jsonrpc":"2.0","id":0,"error":{"code":-32000,"message":"authentication needed: password or unlock"}}
```

```
geth --rinkeby --syncmode "light" --http --allow-insecure-unlock --unlock
8af72ec4f53704fff24737f0445ddb40483eebd1 --password pwd.txt
```

```
~ curl -X POST http://localhost:8545 \
-H "Content-Type: application/json" \
--data '{"jsonrpc":"2.0", "method":"eth_sendTransaction", "params":[{"from": "0x8af72ec4f53704fff24737f0445ddb40483eebd1", "to": "0x35F18427567108F800BDC2784277B9246eED37fA", "value": "0xDE0B6B3A7640000"}], "id":0}'

{"jsonrpc":"2.0","id":0,"result":"0xdcaa9ffaaf07a9d63376653e89892e2dbb1689ec92aad7a843ca393ce42f9209"}
```

NOTE: we are using the insecure mode only for tutorial purposes, and in a testnet.


Observe transaction in etherscan

Goerli Testnet Network

Transaction Details

[Overview](#)[Access List](#)[State](#)

[This is a Goerli **Testnet** transaction only]

? Transaction Hash: [0xdcaa9ffaaf07a9d63376653e89892e2dbb1689ec92aad7a843ca393ce42f9209](#) 

? Status: ✓ Success

? Block: [5426161](#) 5 Block Confirmations

? Timestamp: ⌚ 1 min ago (Sep-02-2021 06:20:27 AM +UTC)

? From: [0x8af72ec4f53704fff24737f0445ddb40483eebd1](#) 

? To: [0x35f18427567108f800bdc2784277b9246eed37fa](#) 

? Value: 1 Ether (\$0.00)

? Transaction Fee: 0.000021004838001 Ether (\$0.00)

? Gas Price: 0.000000001000230381 Ether (1.000230381 Gwei)

? Txn Type: 2 (EIP-1559)

eth_sendTransaction Parameters

POST eth_sendTransaction

Parameters

from: DATA, 20 Bytes - The address the transaction is send from.

to: DATA, 20 Bytes - (optional when creating new contract) The address the transaction is directed to.

gas: QUANTITY - (optional, default: 90000) Integer of the gas provided for the transaction execution. It will return unused gas.

gasPrice: QUANTITY - (optional, default: To-Be-Determined) Integer of the gasPrice used for each paid gas

value: QUANTITY - (optional) Integer of the value sent with this transaction

data: DATA - The compiled code of a contract OR the hash of the invoked method signature and encoded parameters. For details see Ethereum Contract ABI

nonce: QUANTITY - (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.

```
params: [{ "from": " 0xb60e8dd61c5d32be8058bb8eb970870f07233155", "to": " 0xd46e8dd67c5d32be8058bb8eb970870f07244567", "gas": "0x76c0", // 30400 "gasPrice":  
"0x9184e72a000", // 10000000000000 "value": "0x9184e72a", // 2441406250 "data":  
"0xd46e8dd67c5d32be8d46e8dd67c5d32be8058bb8eb970870f072445675058bb8eb970870f072445675" }]
```

Returns

DATA, 32 Bytes - the transaction hash, or the zero hash if the transaction is not yet available.

Use eth_getTransactionReceipt to get the contract address, after the transaction was mined, when you created a contract.

<https://documenter.getpostman.com/view/4117254/ethereum-json-rpc/RVu7CT5J#b3dfaaef-e32b-545c-70ec-783daa8e19ee>

web3.js

web3.js - Ethereum JavaScript API

<https://web3js.readthedocs.io/>

web3.js is a collection of libraries that allow you to interact with a local or remote ethereum node using HTTP, IPC or WebSocket.

Installation

```
npm install web3
```

(requires nodejs and npm)

Connecting web3 to Geth

```
// Import web3
var Web3 = require('web3');

// Connect to local Geth client
var web3 = new Web3('http://localhost:8545');

// Query balance of an account
web3.eth.getBalance("0x8AF72Ec4f53704FfF24737f0445ddB40483eebd1").then(console.log);
```

Executing Transactions with web3

```
// Import web3
var Web3 = require('web3');

// Connect to local Geth client
var web3 = new Web3('http://localhost:8545');

// Prepare transaction
var transaction = {
  from: "0x8af72ec4f53704fff24737f0445ddb40483eebd1",
  to: "0x9808f22453Ee87cc23eA76ca7Ed66a4F294d54D4",
  value: "0xDE0B6B3A7640000"
}

// Send the transaction
web3.eth.sendTransaction(transaction).then(console.log);
```

Invoking Smart Contracts

Consider a smart contract as follows:

```
pragma solidity >=0.7.0 <0.9.0;
```

```
contract Storage {
```

```
    uint256 number;
```

```
    function store(uint256 num) public {
```

```
        number = num;
```

```
    }
```

```
    function retrieve() public view returns (uint256){
```

```
        return number;
```

```
    }
```

```
}
```

Invoking Smart Contracts

To interact with a smart contract, two information are required:

- **Address** of the deployed contract.

0xfb7322195ff15788393d2a2826c5d3c315c139a5

- **ABI** of the smart contract - Application Binary Interface

Can be generated from the source code of the smart contract.

- Given a source code, generate ABI using:

1. Remix editor: <https://remix.ethereum.org/>

Or

2. Using the **solc** compiler: `solc filename.sol --abi`

<https://docs.soliditylang.org/en/v0.5.3/abi-spec.html>

Example ABI

```
[
  {
    "inputs": [],
    "name": "retrieve",
    "outputs": [
      {
        "internalType": "uint256",
        "name": "",
        "type": "uint256"
      }
    ],
    "stateMutability": "view",
    "type": "function"
  },
  {
    "inputs": [
      {
        "internalType": "uint256",
        "name": "num",
        "type": "uint256"
      }
    ],
    "name": "store",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function"
  }
]
```

Executing Smart Contract using Web3js

```
// Import web3
var Web3 = require('web3');

var Contract = require('web3-eth-contract');

// set provider
Contract.setProvider(new Web3.providers.HttpProvider('http://127.0.0.1:8545'));

var myContract = new Contract([paste ABI here], '0xfb7322195ff15788393d2a2826c5d3c315c139a5', // address of the contract
{
  from: '0x8af72ec4f53704fff24737f0445ddb40483eebd1', // address from which you want to transact
  gasPrice: '20000000000' // default gas price in wei, 20 gwei in this case
});

myContract.methods.store(66).send()
.then(function(receipt) {
  console.log(receipt)
});
```


Query Smart Contract

Query public view functions as follows:

```
// Import web3
var Web3 = require('web3');

var Contract = require('web3-eth-contract');

// set provider
Contract.setProvider(new Web3.providers.HttpProvider('http://localhost:8545'));

var myContract = new Contract([paste abi here], '0xfb7322195ff15788393d2a2826c5d3c315c139a5', // address of the contract
{
  from: '0x8af72ec4f53704fff24737f0445ddb40483eebd1', // address from which you want to transact
  gasPrice: '20000000000' // default gas price in wei, 20 gwei in this case
});

myContract.methods.retrieve().call()
  .then(function(output) {
    console.log(output)
  });
```

References

JSON RPC doc:

[https://playground.open-rpc.org/?schemaUrl=https://raw.githubusercontent.com/ethereum/eth1.0-apis/ass
embled-spec/openrpc.json&uiSchema%5BappBar%5D%5Bui:splitView%5D=true&uiSchema%5BappBar
%5D%5Bui:input%5D=false&uiSchema%5BappBar%5D%5Bui:examplesDropdown%5D=false](https://playground.open-rpc.org/?schemaUrl=https://raw.githubusercontent.com/ethereum/eth1.0-apis/ass
embled-spec/openrpc.json&uiSchema%5BappBar%5D%5Bui:splitView%5D=true&uiSchema%5BappBar
%5D%5Bui:input%5D=false&uiSchema%5BappBar%5D%5Bui:examplesDropdown%5D=false)

[https://documenter.getpostman.com/view/4117254/ethereum-json-rpc/RVu7CT5J#b3dfaaef-e32b-545c-70
ec-783daa8e19ee](https://documenter.getpostman.com/view/4117254/ethereum-json-rpc/RVu7CT5J#b3dfaaef-e32b-545c-70
ec-783daa8e19ee)

Web3.js doc:

<https://web3js.readthedocs.io/en/v1.5.0/>

Evm:

<https://ethereum.org/en/developers/docs/evm/>

