NPTEL ONLINE CERTIFICATION COURSES

Blockchain and its applications

Prof. Sandip Chakraborty

Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur

Lecture 21: Beyond PoW

CONCEPTS COVERED

- **Open Consensus beyond PoW**

KEYWORDS

- Proof of Stake (PoS)
- Proof of Burn (PoB)
- Proof of Elapsed Time (PoET)

The Limit of PoW

- **The Good:** A fully decentralized consensus for permissionless models
  - works good for cryptocurrencies – serves its purposes

The Limit of PoW

- **The Bad:** Do not trust the individuals, but trust the society as a whole
  - You need a real large network to prevent the 51% attack – **not at all suitable for enterprise applications**

The Limit of PoW

- **The Ugly**: Low transaction throughput, Overuse of computing power !!
  - (Bitcoin) 3.3 to 7 transactions per second, (Ethereum) ~15 transactions per second
  - Millions of miners – thousands tries, but only one gets the success

Bitcoin Energy Consumption

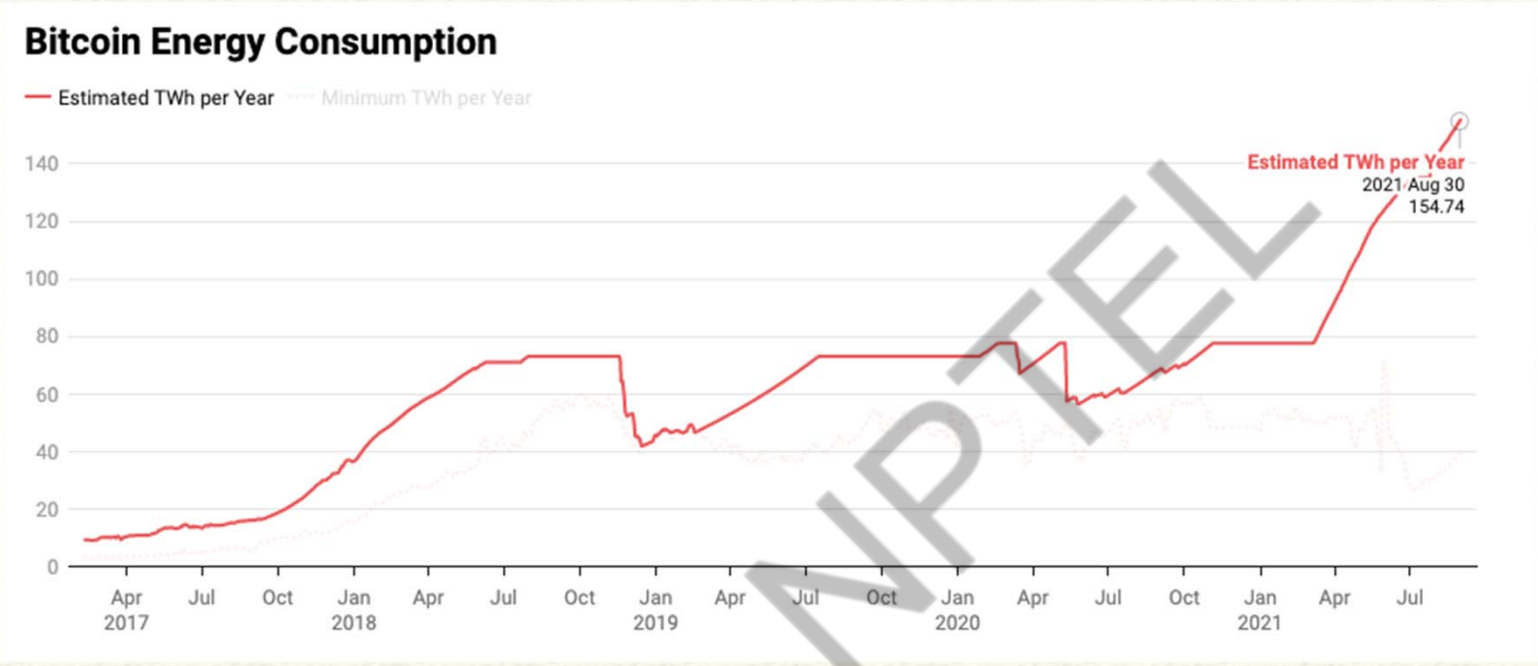
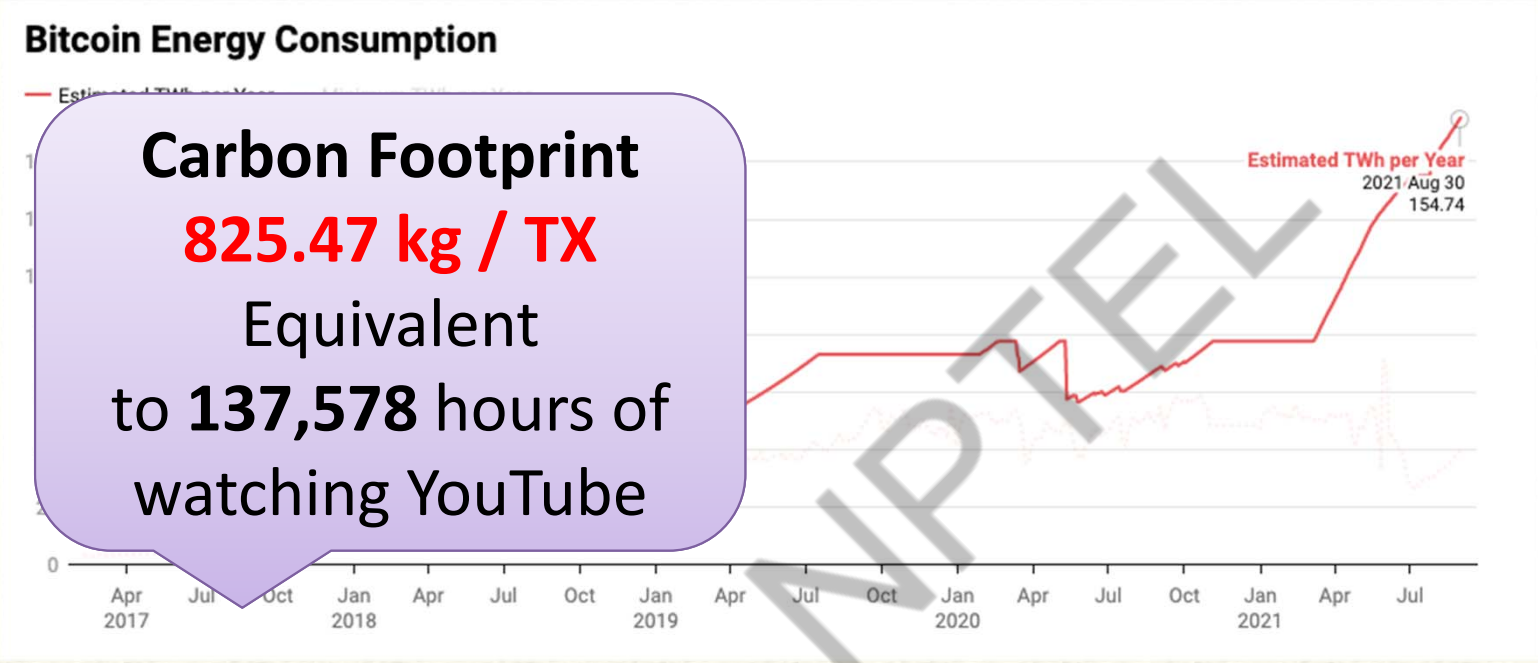


Image Source: Digiconomist Bitcoin Energy Consumption Index

Bitcoin Energy Consumption



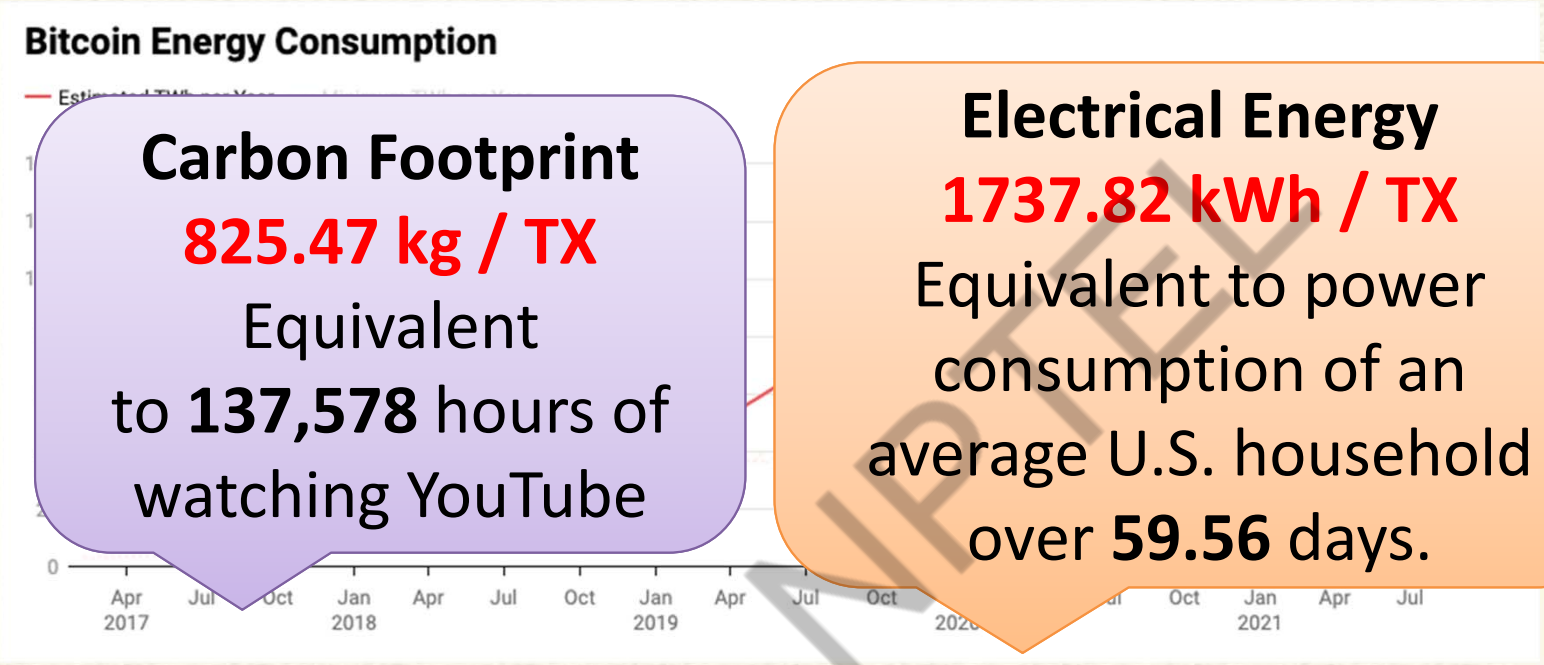
Carbon Footprint

825.47 kg / TX

Equivalent to **137,578** hours of watching YouTube

Image Source: Digiconomist Bitcoin Energy Consumption Index

Bitcoin Energy Consumption



Carbon Footprint

825.47 kg / TX

Equivalent to **137,578** hours of watching YouTube

Electrical Energy

1737.82 kWh / TX

Equivalent to power consumption of an average U.S. household over **59.56** days.

Image Source: Digiconomist Bitcoin Energy Consumption Index

Proof of Stake (PoS)

- Possibly proposed in 2011 by a Member in Bitcoin Forum - https://bitcointalk.org/index.php?topic=27787.0
  - Make a transition from PoW to PoS when bitcoins are widely distributed

Proof of Stake (PoS)

- PoW vs PoS
  - **PoW**: Probability of mining a block depends on the work done by the miner
  - **PoS**: Amount of bitcoin that the miner holds – Miner holding 1% of the Bitcoin can mine 1% of the PoS blocks.

Proof of Stake (PoS)

- Provides increased protection
  - Executing an attack is expensive, you need more Bitcoins
  - **Reduced incentive for attack** – the attacker needs to own a majority of bitcoins – an attack will have more affect on the attacker

Proof of Stake (PoS)

- Variants of “stake”
  - Randomization in combination of the stake (*used in Nxt and BlackCoin*)
  - **Coin-age**: Number of coins multiplied by the number of days the coins have been held (*used in Peercoin*)

Proof of Burn (PoB)

- Miners should show proof that they have *burned* some coins
  - Sent them to a verifiably un-spendable address
  - Expensive just like PoW, but no external resources are used other than the burned coins
- PoW vs PoB
  - Real resource vs virtual/digital resource
- PoB works by burning PoW mined cryptocurrencies

Proof of Burn (PoB)

- Mine … some coin
  - S…
  - B… used c…
- PoW…
  - P…
- PoB w… …cies

PoS and PoB

Ultimately depends on PoW mined cryptocurrencies

You cannot use them to bootstrap a new blockchain

Proof of Elapsed Time (PoET)

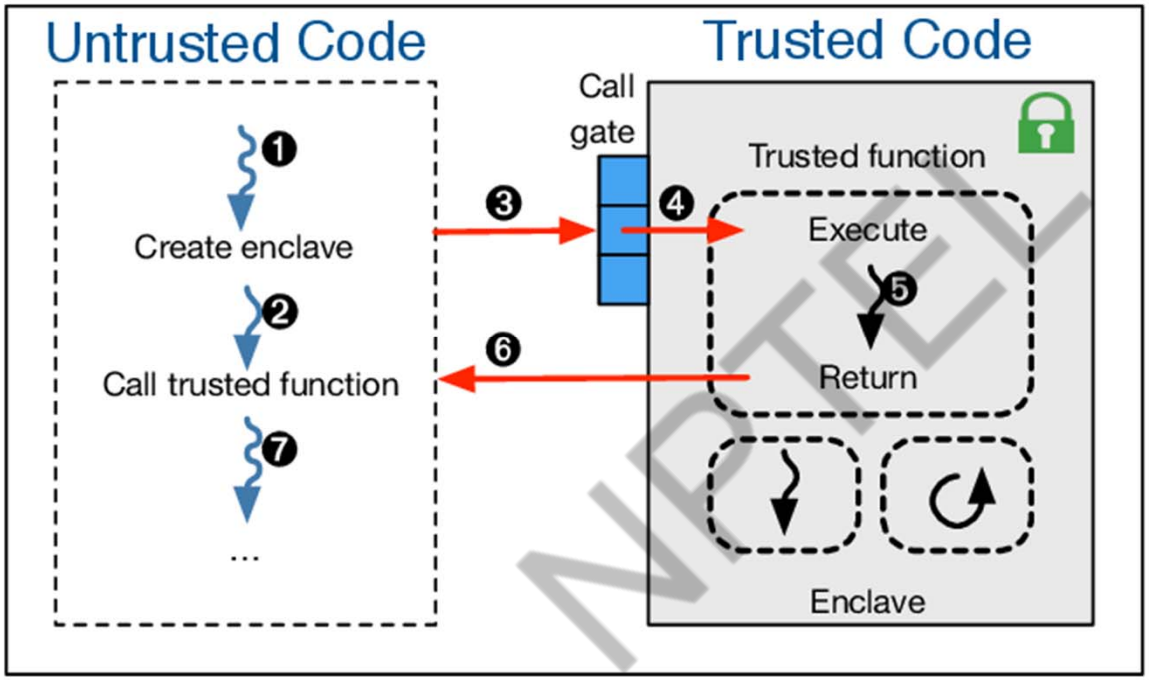
- Proposed by Intel, as a part of **Hyperledger Sawtooth** – a blockchain platform for building distributed ledger applications

- **Basic idea:**
  - Each participant in the blockchain network waits a random amount of time
  - The first participant to finish becomes the leader for the new block

Proof of Elapsed Time (PoET)

- How will one verify that the proposer has **really waited** ?
  - Utilize special CPU instruction set – *Intel Software Guard Extension* (SGX) – a trusted execution platform
  - The trusted code is private to the rest of the application
  - The specialized hardware provides an attestation that the trusted code has been set up correctly

Intel SGX


Untrusted Code
Trusted Code
Call gate
Trusted function
Create enclave
Execute
Call trusted function
Return
Enclave
...

Conclusion

- PoW is significantly costly
  - Reduce the cost by moving towards PoS/PoB
- Los-cost consensus from the bootstrap: PoET
  - Needs specialized hardware
- What about enterprise application?

NPTEL ONLINE CERTIFICATION COURSES

Blockchain and its applications

Bishakh Chandra Ghosh

Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur

Lecture 22: Ethereum 1

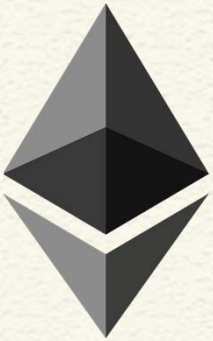
CONCEPTS COVERED

- Ethereum Introduction
- Ethereum Network
- Go Ethereum
- Query using RPC over HTTP

KEYWORDS

- Ethereum
- Geth
- Testnets

Ethereum



- "Ethereum is a technology that lets you send **cryptocurrency** to anyone for a small fee. It also **powers applications that everyone can use and no one can take down**."

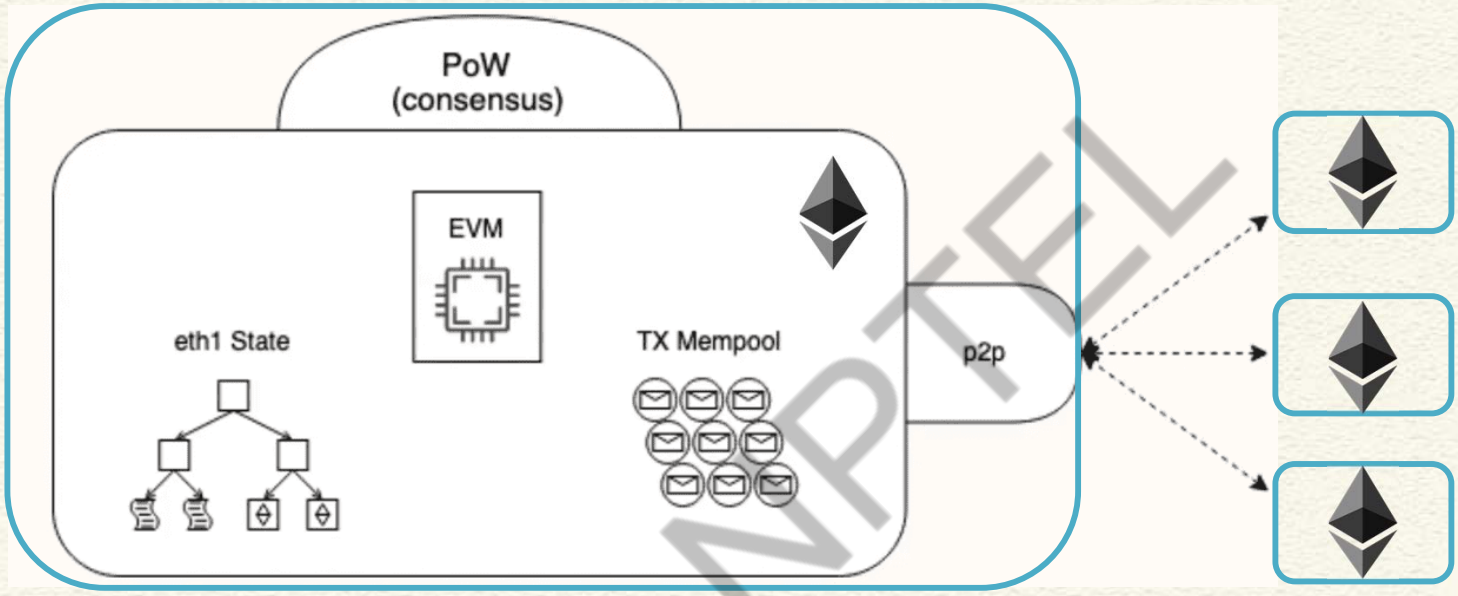
- **Permissionless** blockchain **capable of executing smart contracts**.

https://ethereum.org/en/what-is-ethereum/

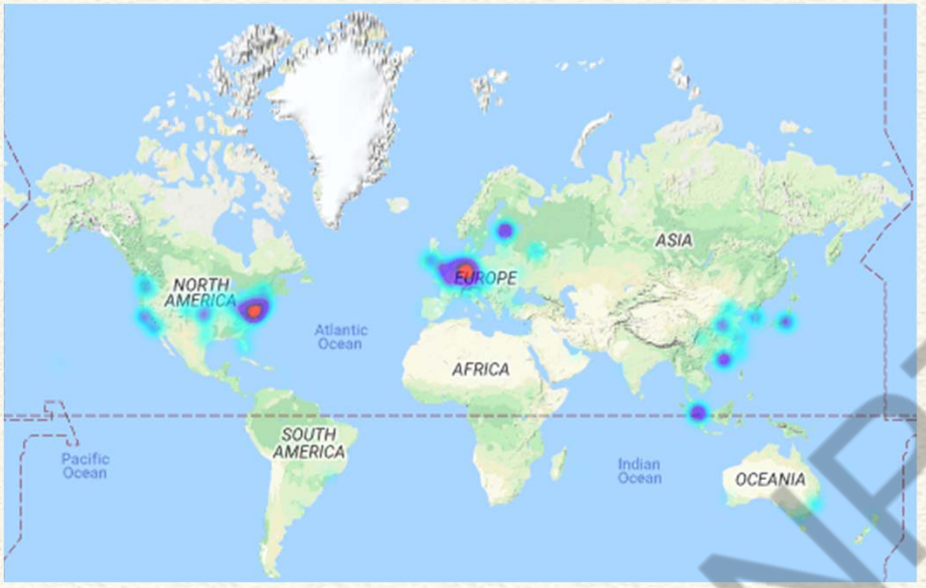
Ethereum Network

- **Distributed network of computers**, known as **nodes** that can verify blocks and transaction data.
- An application, known as a **client**, running on your computer is a node.

Ethereum Network



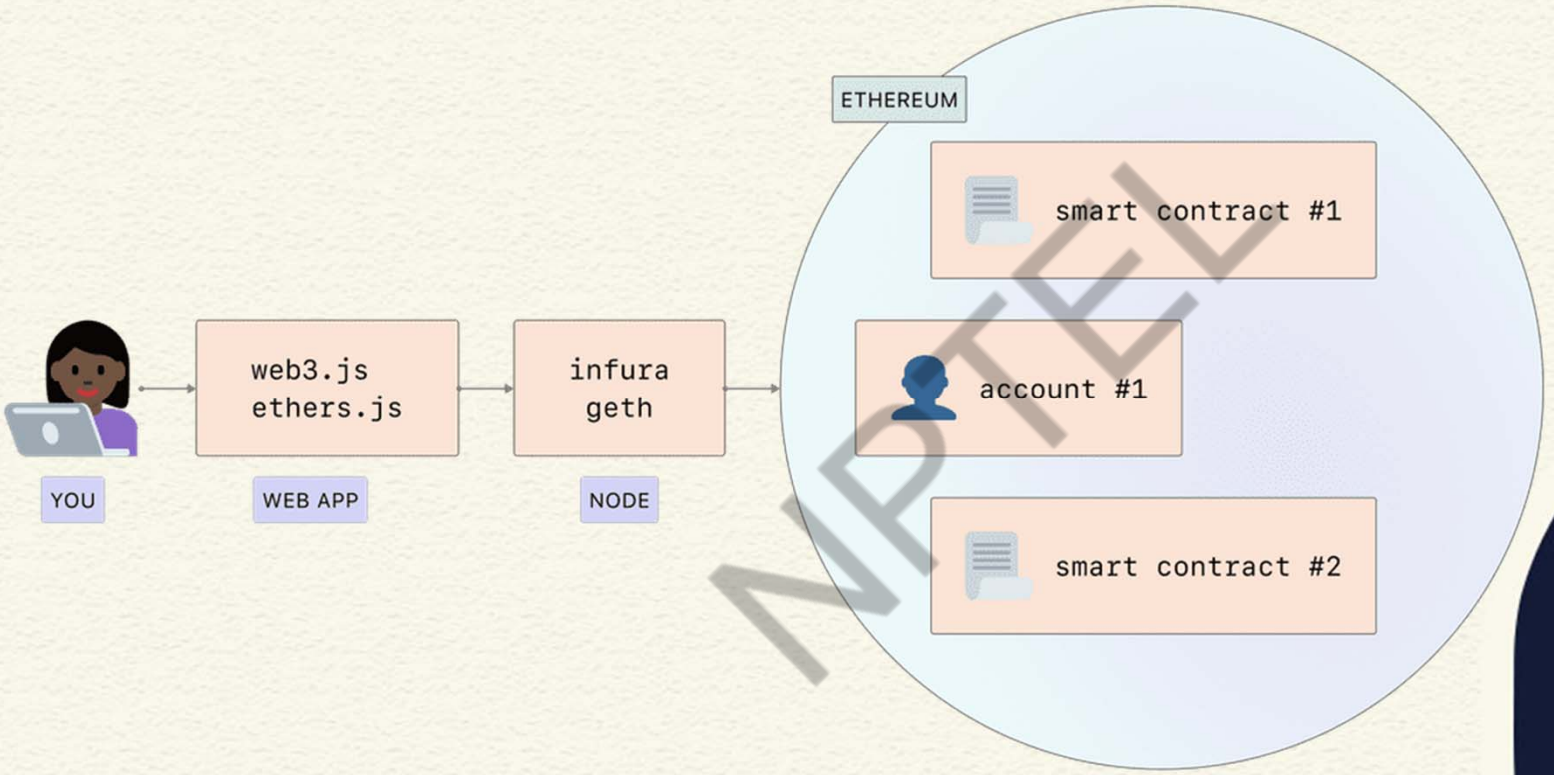
Ethereum Mainnet



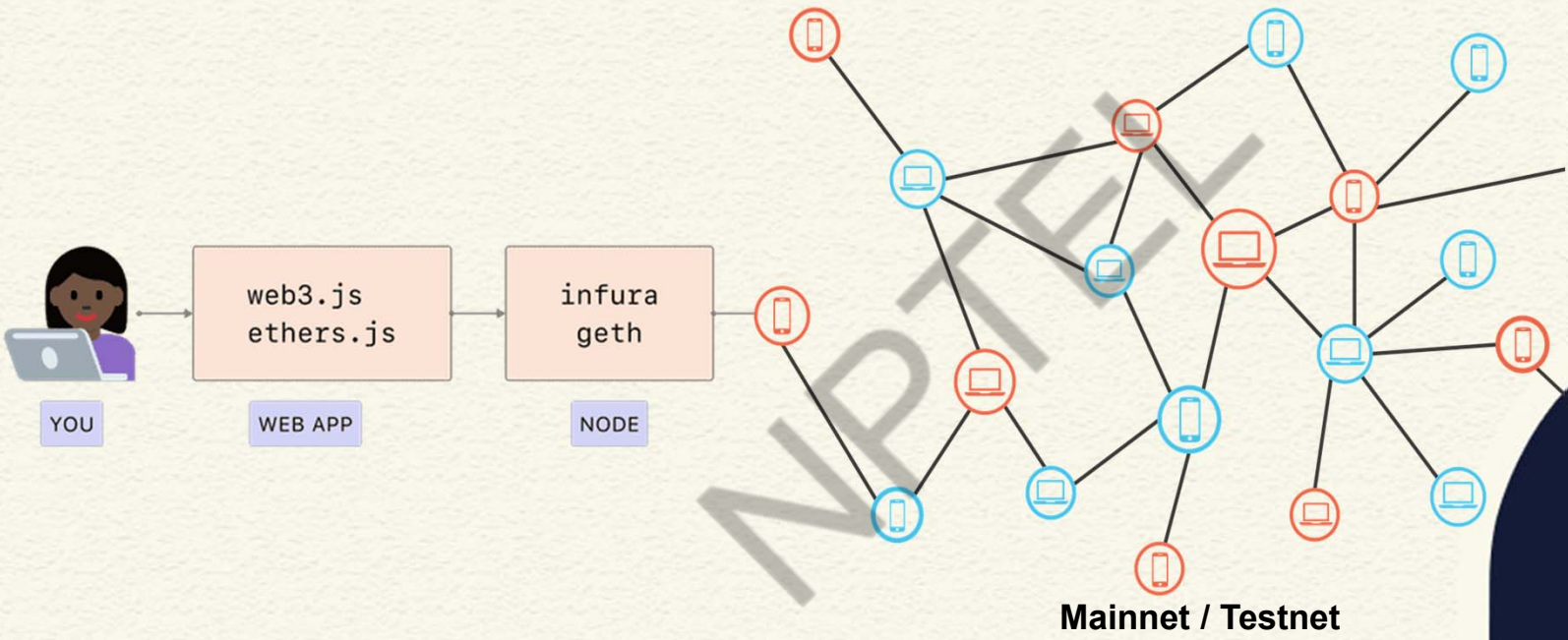
- **No central server**
- Independent nodes connected in a P2P network.

https://ethstats.net/
https://www.ethernodes.org/countries

Interacting with an Ethereum Network



Interacting with an Ethereum Network



Go Ethereum (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

1. Add ethereum repository:

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

2. Then install the stable version of go-ethereum:

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

Managing Ethereum Accounts

USAGE:

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

COMMANDS:

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

Create an account

geth account new

```
~/nptel geth account new
INFO [10-27|01:53:10.892] Maximum peer count                       ETH=50 LES=0 total=50
INFO [10-27|01:53:10.893] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no su
ch file or directory"
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:

Your new key was generated

Public address of the key:   0x8010319e8e3115EA522Dc2f00ab9bC28A5AbBf80
Path of the secret key file: /home/bishakh/.ethereum/keystore/UTC--2021-10-26T20-23-14.304161080Z--801031
9e8e3115ea522dc2f00ab9bc28a5abbf80

- 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

```
~/nptel geth account list
INFO [10-27|01:53:46.704] Maximum peer count                       ETH=50 LES=0 total=50
INFO [10-27|01:53:46.704] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no su
ch file or directory"
INFO [10-27|01:53:46.705] Set global gas cap                       cap=50,000,000
Account #0: {8010319e8e3115ea522dc2f00ab9bc28a5abbf80} keystore:///home/bishakh/.ethereum/keystore/UTC--2
021-10-26T20-23-14.304161080Z--8010319e8e3115ea522dc2f00ab9bc28a5abbf80
```

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

```
~/nptel cat ~/.ethereum/keystore/UTC--2021-10-26T20-23-14.304161080Z--8010319e8e3115ea522dc2f00ab9bc28
a5abbf80
{"address":"8010319e8e3115ea522dc2f00ab9bc28a5abbf80","crypto":{"cipher":"aes-128-ctr","ciphertext":"8adc
e6d4bb5200e47d41115200020de93eb1b4e529777ead7235c769613e748e","cipherparams":{"iv":"d1b38535f133c9cc8d787
2b06417fd4d"},"kdf":"scrypt","kdfparams":{"dklen":32,"n":262144,"p":1,"r":8,"salt":"99dcda6aa950e2c3fb6bf
adbe00ba12a6cf4dd72f5a90116b9587f7388c908a4"},"mac":"1d492f1a132284a38ccada2c6ab0cea697a9529df52489dc8ddf
8ecd3af84628"},"id":"403b8b04-886f-4fc5-b8cf-f12f26ee0565","version":3}%
```

Connect to a 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 https://goerli.etherscan.io/

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:

- **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 Goerli testnet

geth –goerli --syncmode "light"

```
~ geth --goerli --syncmode "light"
INFO [11-02|12:47:58.049] Starting Geth on Görli testnet...
INFO [11-02|12:47:58.049] Dropping default light client cache      provided=1024 updated=128
INFO [11-02|12:47:58.053] Maximum peer count                       ETH=0 LES=10 total=50
INFO [11-02|12:47:58.053] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such file or directory"
INFO [11-02|12:47:58.053] Set global gas cap                       cap=50,000,000
INFO [11-02|12:47:58.054] Allocated cache and file handles         database=/home/bishakh/.ethereum/goerli/geth/lightchaindata
cache=64.00MiB handles=2048
INFO [11-02|12:47:58.100] Allocated cache and file handles         database=/home/bishakh/.ethereum/goerli/geth/les.client cach
e=16.00MiB handles=16
INFO [11-02|12:47:58.164] Persisted trie from memory database      nodes=361 size=51.17KiB time=4.610792ms gcnodes=0 gcsize=0.0
0B gctime=0s livenodes=1 livesize=0.00B
INFO [11-02|12:47:58.166] Initialised chain configuration          config="{ChainID: 5 Homestead: 0 DAO: <nil> DAOSupport: true
 EIP150: 0 EIP155: 0 EIP158: 0 Byzantium: 0 Constantinople: 0 Petersburg: 0 Istanbul: 1561651, Muir Glacier: <nil>, Berlin: 446
0644, London: 5062605, Engine: clique}"
INFO [11-02|12:47:58.180] Added trusted checkpoint                 block=5,275,647 hash=b5a666..34b5a5
INFO [11-02|12:47:58.180] Loaded most recent local header          number=5,777,531 hash=fcf2f2..274e52 td=8,469,843 age=42s
INFO [11-02|12:47:58.181] Configured checkpoint oracle             address=0x18CA0E045F0D772a851BC7e48357Bcaab0a0795D signers=5
 threshold=2
```

Copy account to testnetwork

cp ~/.ethereum/keystore/UTC--2021-10-26T20-23-14.304161080Z--8010319e8e3115ea522dc2f00ab9bc28a5abbf80 ~/.ethereum/rinkeby/keystore/

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

geth --goerli --syncmode "light" --http

```
~ geth --goerli --syncmode "light" --http
INFO [11-02|12:49:39.576] Starting Geth on Görli testnet...
INFO [11-02|12:49:39.576] Dropping default light client cache      provided=1024 updated=128
INFO [11-02|12:49:39.578] Maximum peer count                       ETH=0 LES=10 total=50
INFO [11-02|12:49:39.578] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such file or directory"
INFO [11-02|12:49:39.579] Set global gas cap                       cap=50,000,000
INFO [11-02|12:49:39.579] Allocated cache and file handles         database=/home/bishakh/.ethereum/goerli/geth/lightchaindata cache=64.00MiB han
dles=2048
INFO [11-02|12:49:39.614] Allocated cache and file handles         database=/home/bishakh/.ethereum/goerli/geth/les.client cache=16.00MiB handles
=16
INFO [11-02|12:49:39.665] Persisted trie from memory database      nodes=361 size=51.17KiB time=1.009893ms gcnodes=0 gcsize=0.00B gctime=0s liven
odes=1 livesize=0.00B
INFO [11-02|12:49:39.666] Initialised chain configuration          config="{ChainID: 5 Homestead: 0 DAO: <nil> DAOSupport: true EIP150: 0 EIP155:
 0 EIP158: 0 Byzantium: 0 Constantinople: 0 Petersburg: 0 Istanbul: 1561651, Muir Glacier: <nil>, Berlin: 4460644, London: 5062605, Engine: cliqu
e}"
```

```
WARN [11-02|12:49:39.668] Unclean shutdown detected                booted=2021-09-08T13:04:34+0530 age=1mo3w3d
WARN [11-02|12:49:39.668] Unclean shutdown detected                booted=2021-10-27T03:48:19+0530 age=6d9h1m
INFO [11-02|12:49:39.668] Starting peer-to-peer node               instance=Geth/v1.10.8-stable-26675454/linux-amd64/go1.16.4
INFO [11-02|12:49:39.720] New local node record                    seq=69 id=c0aaffb0082e603d ip=127.0.0.1 udp=30303 tcp=30303
INFO [11-02|12:49:39.720] Started P2P networking                   self=enode://c768b3ba0b37788ddf7ebcaa4c7677df0e3a40a9c5a92a28361ca8df726f47544
9287242095136a7ac96ef51eb8ae85d9b8dda8b7f6b9936b0ff9f7bf0f7b5d7f0127.0.0.1:30303
INFO [11-02|12:49:39.721] IPC endpoint opened                      url=/home/bishakh/.ethereum/goerli/geth.ipc
INFO [11-02|12:49:39.721] HTTP server started                      endpoint=127.0.0.1:8545 prefix= cors= vhosts=localhost
WARN [11-02|12:49:39.721] Light client mode is an experimental feature
INFO [11-02|12:49:49.883] Block synchronisation started
```

Query Balance

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
}
```

Query using curl

Querying the balance of an account:

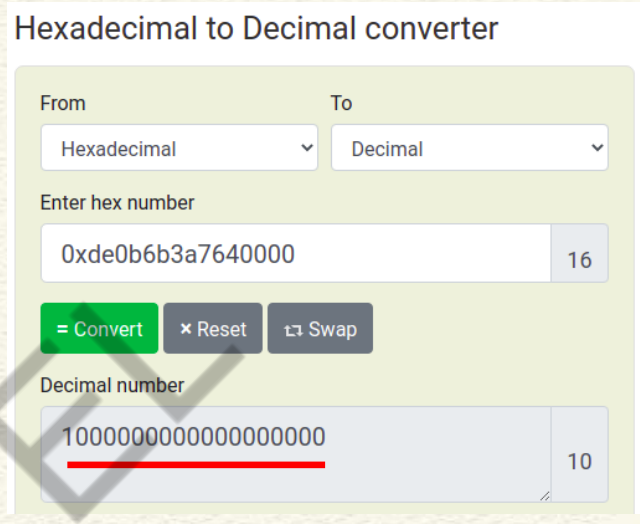
POST request using curl:

```
curl -X POST http://localhost:8545 \
-H "Content-Type: application/json" \
--data '{"jsonrpc":"2.0", "method":"eth_getBalance",
"params":["0x5342722156fcd4b0e0b140c4fb9cd63dcea347f4","latest"], "id":1}'
```

```
~/nptel curl -X POST http://localhost:8545 \
    -H "Content-Type: application/json" \
   --data '{"jsonrpc":"2.0", "method":"eth_getBalance", "params":["0x5342722156fcd4b0e0b140c4fb9cd63dcea3
47f4","latest"], "id":1}'
{"jsonrpc":"2.0","id":1,"result":"0xde0b6b3a7640000"}
```

Ethereum Units

Unit	Wei Value	Wei
wei	1 wei	1
Kwei (babbage)	1e3 wei	1,000
Mwei (lovelace)	1e6 wei	1,000,000
Gwei (shannon)	1e9 wei	1,000,000,000
microether (szabo)	1e12 wei	1,000,000,000,000
milliether (finney)	1e15 wei	1,000,000,000,000,000
ether	1e18 wei	1,000,000,000,000,000,000



Conclusion

- Ethereum
  - Permissionless, smart contract support
  - Go ethereum
  - Main network, test networks
  - Accounts
- Query using RPC over HTTP

NPTEL ONLINE CERTIFICATION COURSES

Blockchain and its applications

Bishakh Chandra Ghosh

Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur

Lecture 23: Ethereum 2

CONCEPTS COVERED

- Obtaining Ethereum for testnets
- Unlocking account
- Geth transactions using RPC over HTTP

KEYWORDS

- Ethereum Transactions
- Gas
- Faucet

Query Balance

Ensure Geth is running.

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
}
```

Query Balance

Querying the balance of an account:

POST request using curl:

```
curl -X POST http://localhost:8545 \
-H "Content-Type: application/json" \
--data '{"jsonrpc":"2.0", "method":"eth_getBalance",
"params":["0x5342722156fcd4b0e0b140c4fb9cd63dcea347f4","latest"],
"id":1}'
```

```
~/nptel curl -X POST http://localhost:8545 \
    -H "Content-Type: application/json" \
   --data '{"jsonrpc":"2.0", "method":"eth_getBalance", "params":["0x5342722156fcd4b0e0b140c4fb9cd63dcea3
47f4","latest"], "id":1}'
{"jsonrpc":"2.0","id":1,"result":"0xde0b6b3a7640000"}
```

Ethereum Transactions

- Transactions are **cryptographically signed instructions from accounts**.
- An account will initiate a transaction to update the state of the Ethereum network.
- The simplest transaction is transferring ETH from one account to another.

Ethereum Transactions

- Transactions are **cryptographically signed instructions from accounts**.
- An account will initiate a transaction to update the state of the Ethereum network.
- The simplest transaction is transferring ETH from one account to another.

Requires:
1. Access to account private key
2. Gas

Gas

- Each Ethereum **transaction requires computational resources** to execute.
- Each transaction requires a **fee**.
- **Gas refers to the fee required to conduct a transaction on Ethereum successfully.**
- Gas is paid as amounts of Ethers.

Get ether in testnet

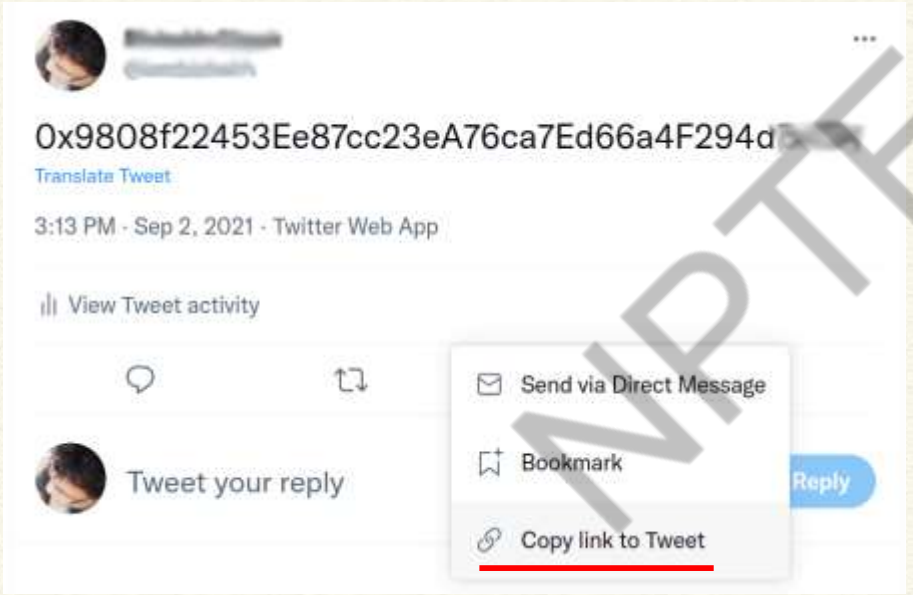
- For executing transactions in a testnet, we can obtain ethers from faucet.

- Goerli faucet:
  - Social faucet: https://faucet.goerli.mudit.blog/
  - Simple faucet: https://goerli-faucet.slock.it/

- Rinkeby faucet: https://faucet.rinkeby.io/

Get ether in Goerli testnet

- Post a tweet containg ethereum account address.



Get ether in Goerli testnet

- Paste tweet link in faucet.

Unlock Account

- Create new account for goerli testnet, suppose password is set to "123"
- Write account password in a text file. Here we name it pwd.txt

```
geth --goerli account new
echo "123" > pwd.txt
```

- Unlock account while starting geth

```
geth --goerli --syncmode "light" --http --allow-insecure-unlock --unlock
d84a0607843b28c3f468857f82f784d9ff743bf8 --password pwd.txt
```

eth_sendTransaction

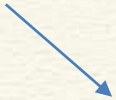
POST sendTransaction
Creates new message call transaction or a contract creation, if the data field contains code.

Parameters
Object - The transaction object
from: DATA, 20 Bytes - The address the transaction is send from.
to: DATA, 20 Bytes -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) 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.

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

Prepare Transaction

```
{
"jsonrpc": "2.0",
"method": "eth_sendTransaction",
"params": [
{
"from": "0xd84a0607843b28c3f468857f82f784d9ff743bf8",
"to": "0x5f2316ff45c1c782e027b548f6f6e604655148f1",
"value": "0x2386F26FC10000"
}
],
"id": 0
}
```



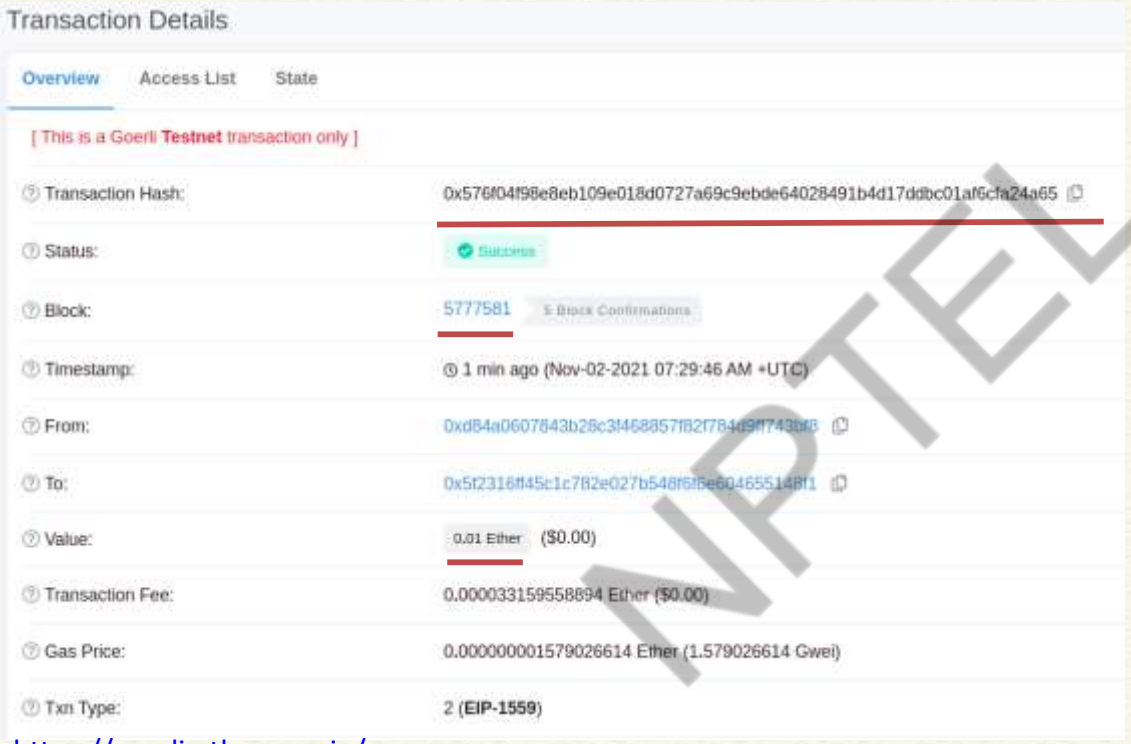
0.1 eth

Submit Transaction

curl -X POST http://localhost:8545 \
-H "Content-Type: application/json" \
--data '{"jsonrpc":"2.0", "method":"eth_sendTransaction", "params":[{"from": "0xd84a0607843b28c3f468857f82f784d9ff743bf8","to": "0x5f2316ff45c1c782e027b548f6f6e604655148f1","value": "0x2386F26FC10000"}], "id":0}'

```
~/nptel curl -X POST http://localhost:8545 \
    -H "Content-Type: application/json" \
    --data '{"jsonrpc":"2.0", "method":"eth_sendTransaction", "params":[{"from": "0xd84a0607843b28c3f4
68857f82f784d9ff743bf8","to": "0x5f2316ff45c1c782e027b548f6f6e604655148f1","value": "0x2386F26FC10000
"}], "id":0}'

{"jsonrpc":"2.0","id":0,"result":"0x576f04f98e8eb109e018d0727a69c9ebde64028491b4d17ddbc01af6cfa24a65"
}
```

View Transaction on Etherscan



https://goerli.etherscan.io/

Get Transaction by Hash

```
{
"jsonrpc": "2.0",
"method": "eth_getTransactionByHash",
"params": [
"0x576f04f98e8eb109e018d0727a69c9ebde64028491b4d17ddb
c01af6cfa24a65"
],
"id": 1
}
```

Get Transaction by Hash

curl -X POST http://localhost:8545 \
-H "Content-Type: application/json" \
--data '{"jsonrpc":"2.0", "method":"eth_getTransactionByHash", "params":["0x576f04f98e8eb109e018d0727a69c9ebde64028491b4d17ddbc01af6cfa24a65"], "id":1}'

```
~/nptel curl -X POST http://localhost:8545 \
    -H "Content-Type: application/json" \
   --data '{"jsonrpc":"2.0", "method":"eth_getTransactionByHash", "params":["0x576f04f98e8eb109e018d0
727a69c9ebde64028491b4d17ddbc01af6cfa24a65"], "id":1}'

{"jsonrpc":"2.0","id":1,"result":{"blockHash":"0x9fffe0c3f03c0be6ca11966984d49b7b4761378ba8e185a521a8
aff2558c685e","blockNumber":"0x5828ad","from":"0xd84a0607843b28c3f468857f82f784d9ff743bf8","gas":"0x5
208","gasPrice":"0x5e1e08b6","maxFeePerGas":"0x5e1e08bd","maxPriorityFeePerGas":"0x5e1e08af","hash":"
0x576f04f98e8eb109e018d0727a69c9ebde64028491b4d17ddbc01af6cfa24a65","input":"0x","nonce":"0x2","to":"
0x5f2316ff45c1c782e027b548f6f6e604655148f1","transactionIndex":"0x3","value":"0x2386f26fc10000","type
":"0x2","accessList":[],"chainId":"0x5","v":"0x1","r":"0xd588ec79252a1f4e1c7f9848fc273a087fe1944a6ad0
1c37dba31a98042cc4f","s":"0x211f623b3345abcf4c88031c52a0faa5825460063c5b8e7d88916f07b7649451"}}
```

Query Blocks

```
{
"jsonrpc": "2.0",
"method": "eth_getBlockByNumber",
"params": [
"0x5828AD",
true
],
"id": 1
}
```

```
curl -X POST http://localhost:8545 \
-H "Content-Type: application/json" \
--data '{"jsonrpc":"2.0", "method":"eth_getBlockByNumber",
"params":["0x5828AD", true], "id":1}'
```

Query Blocks

```
~/nptel curl -X POST http://localhost:8545 \
    -H "Content-Type: application/json" \
   --data '{"jsonrpc":"2.0", "method":"eth_getBlockByNumber", "params":["0x5828AD", true], "id":1}'
{"jsonrpc":"2.0","id":1,"result":{"baseFeePerGas":"0x7","difficulty":"0x1","extraData":"0xf09f928e204
07072796c616273206e6f64652d3020f09f928e0000000000000066703a3fb40887573482e9738bfea68041c6154390b26f71
4c31d350b4c26f9d4edd5405013bd12c8eeb04267abc43140edf30b6efbf0ca292b39623cc48124100","gasLimit":"0x1c9
5111","gasUsed":"0x6803b","hash":"0x9fffe0c3f03c0be6ca11966984d49b7b4761378ba8e185a521a8aff2558c685e"
,"logsBloom":"0x00000000010000000000000008000c800000001400000000000010a0000000000000000000000000001
0080000000001200004000000000000000000040044000000000800000000020008004000000000000000040000000000000
0000000410001000000000000040104800009000002000000008000000001000000000000000240000000080020100020002
0006000001000000010000000000000000000002000000008800000000000000000000000000000000000040000000000000
0002000000000000004000000402000000001000080020000000004000082024200000100000600000000000000000000000
0010000100000004020000","miner":"0xd9a5179f091d85051d3c982785efd1455cec8699","mixHash":"0x0000000000
0000000000000000000000000000000000000000000000000000000000","nonce":"0x0000000000000000","number":"0x5828
ad","parentHash":"0xb19e8782b5c5bfc62c2385a7c95ea9f74148d5906a3869191db340cbe3c09312","receiptsRoot":
"0x677d285193bfb476444a099e6605374740ba9719124ee5999838180a9f515965","sha3Uncles":"0x1dcc4de8dec75d7a
ab85b567b6ccd41ad312451b948a7413f0a142fd40d49347","size":"0xcaa","stateRoot":"0xa528d2fc9ce38e7c16bbe
```

Many More RPC Methods

eth_getBlockByHash	Returns information about a block by hash.
eth_getBlockByNumber	Returns information about a block by number.
eth_getBlockTransactionCountByHash	Returns the number of transactions in a block from a block matching the given block hash.
eth_getBlockTransactionCountByNumber	Returns the number of transactions in a block matching the given block number.
eth_getUncleCountByBlockHash	Returns the number of uncles in a block from a block matching the given block hash.
eth_getUncleCountByBlockNumber	Returns the number of transactions in a block matching the given block number.
eth_protocolVersion	Returns the current ethereum protocol version.
eth_syncing	Returns an object with data about the sync status or false.
eth_coinbase	Returns the client coinbase address.
eth_accounts	Returns a list of addresses owned by client.
eth_blockNumber	Returns the number of most recent block.
eth_call	Executes a new message call immediately without creating a transaction on the block chain.
eth_estimateGas	Generates and returns an estimate of how much gas is necessary to allow the transaction to complete.

https://playground.open-rpc.org/?schemaUrl=https://raw.githubusercontent.com/ethereum/eth1.0-apis/assembled-spec/openrpc.json

Conclusion

- Ethereum Transactions
- Obtaining Ether for testnets
- Unlocking Account
- Submit Transactions and monitor transactions

NPTEL ONLINE CERTIFICATION COURSES

Blockchain and its applications

Bishakh Chandra Ghosh

Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur

Lecture 24: Ethereum 3

CONCEPTS COVERED

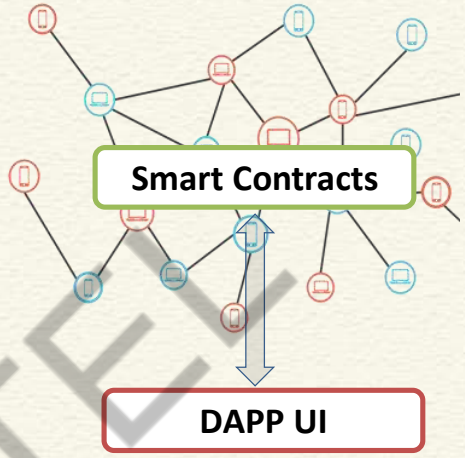
- **Ethereum applications - DAPPS**
- **Using web3.js to programmatically access Ethereum network**

KEYWORDS

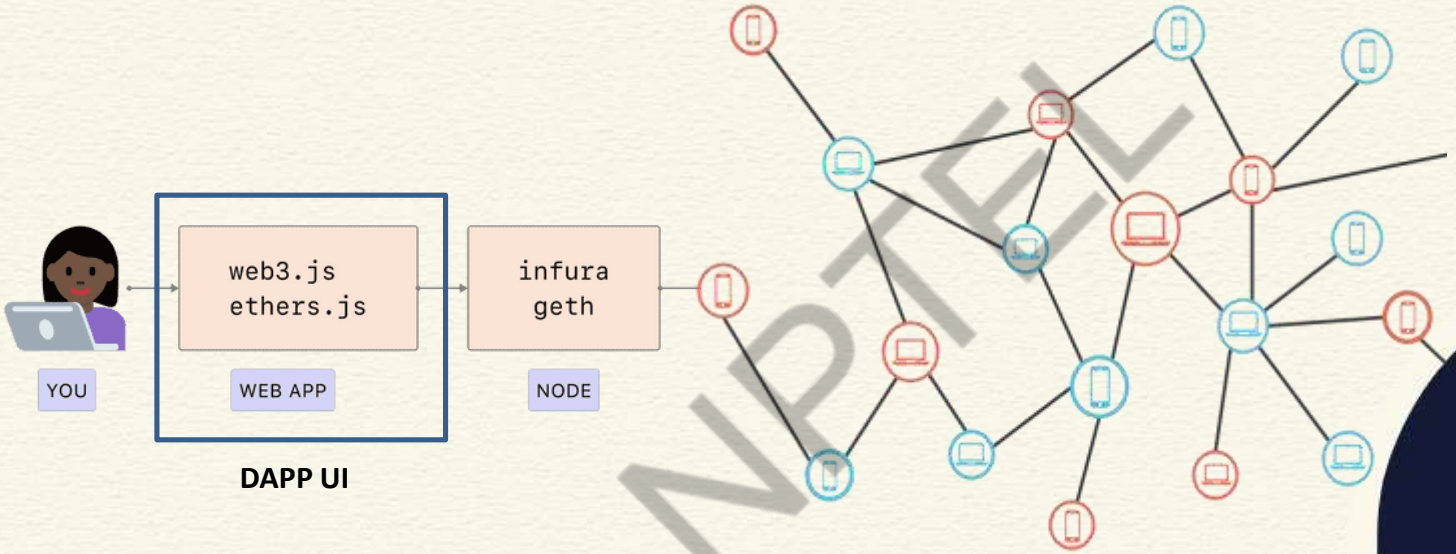
- **web3.js**
- **DAPPS**

DAPPS in Ethereum

- DAPP - decentralized application
- Application that is built for decentralized networks like Ethereum
- Combines two components:
    1. Smart Contracts
    2. User interface for executing transactions and contracts



Programmatically access Ethereum networks

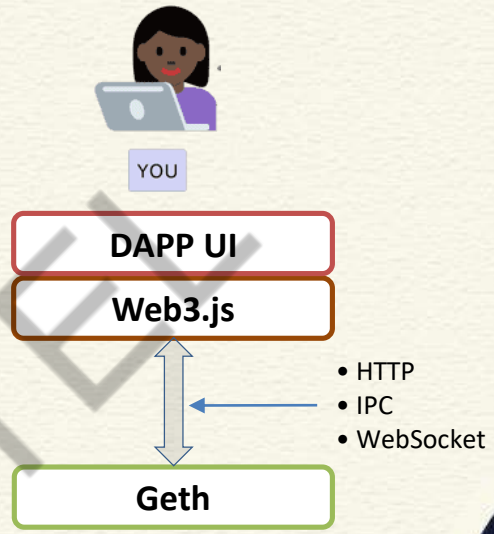


web3.js

- Ethereum JavaScript API
  https://web3js.readthedocs.io/

- Collection of libraries that allow you to interact with a local or remote ethereum nodes.

- It can connect using:
  - HTTP
  - IPC
  - WebSocket



Install web3.js

- Install Node.js
  https://nodejs.org/en/

```
wget https://nodejs.org/dist/v16.13.0/node-v16.13.0-linux-x64.tar.xz

tar -xvf node-v16.13.0-linux-x64.tar.xz

export PATH=/home/<username>/nptel/node-v16.13.0-linux-x64/bin:$PATH
```

- Check node and npm version

```
node --version

npm --version
```

```
~/nptel node --version
v16.13.0
~/nptel npm --version
8.1.0
```

Install web3.js

- Install web3.js

```
npm install web3
```

- Verify that you can import web3

```
node
var Web3 = require('web3');
console.log(Web3.version)
```

Connecting web3.js to Geth

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

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

Query Balance

- Create a file with .js extension. Example: "balance.js"

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

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

// Query balance of an account
console.log("Balance of the account 0x35F18427567108F800BDC2784277B9246eED37fA is:");
web3.eth.getBalance("0x35F18427567108F800BDC2784277B9246eED37fA").then(console.log
);
```

Query Balance

- Create a file with .js extension. Example: "balance.js"

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

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

// Query balance of an account
console.log("Balance of the account 0x35F18427567108F800BDC2784277B9246eED37fA is:");
web3.eth.getBalance("0x35F18427567108F800BDC2784277B9246eED37fA").then(console.log
);
```

```
~/nptel node balance.js
Balance of the account 0x35F18427567108F800BDC2784277B9246eED37fA is:
10000000000000000
```

Transfer Ether

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

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

// Prepare transaction
var transaction = {
from: "0x7dad3a076678a05b2b4e2b93206dbecef0d7bbf0",
to: "0x35F18427567108F800BDC2784277B9246eED37fA",
value: Web3.utils.numberToHex(1000000000000000)
}

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

Transfer Ether

```
~/nptel  node transfer.js
{
  blockHash: '0x51122fbfd03760ac1c22fbbffd274a4ed31a47635f7f01110c580472bdd2ab75',
  blockNumber: 9537443,
  contractAddress: null,
  cumulativeGasUsed: 5399124,
  effectiveGasPrice: '0x1e9119f9b',
  from: '0x7dad3a076678a05b2b4e2b93206dbecef0d7bbf0',
  gasUsed: 21000,
  logs: [],
  logsBloom: '0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000',
  status: true,
  to: '0x35f18427567108f800bdc2784277b9246eed37fa',
  transactionHash: '0x9b933227855ad79ccae62535db3cc12f598ccadab6f7be93526db1704040646b',
  transactionIndex: 6,
  type: '0x2'
}
```

Conclusion

- Web3.js for DAPP Development
- Build interface for Ethereum applications

NPTEL ONLINE CERTIFICATION COURSES

Blockchain and its applications

Bishakh Chandra Ghosh

Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur

Lecture 25: Ethereum 4

CONCEPTS COVERED

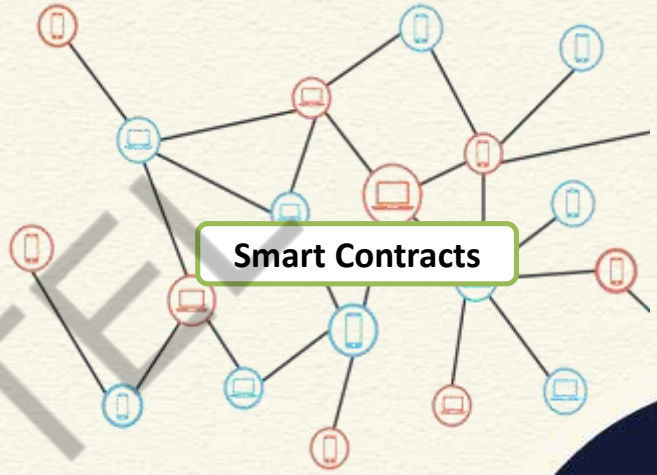
- Ethereum smart contracts
- Ethereum Virtual Machine (EVM)
- Solidity language
- Deploy and execute contracts

KEYWORDS

- **Ethereum Smart Contracts**
- **EVM**
- **Solidity**

Smart Contracts

- Program that is executed in a decentralized setting.
- Acts on distributed ledger data.
- Output of the program depends on consensus of independent participants executing it.



Ethereum Smart Contracts

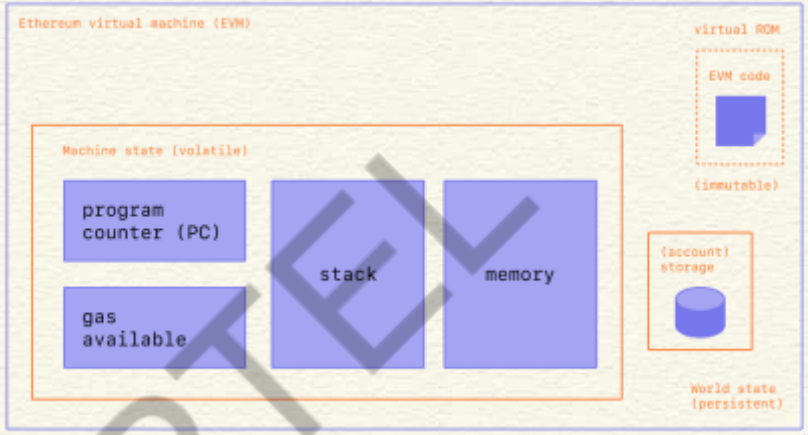
- Program that reads data from the Ethereum ledger, performs operations on it, and writes back the output to the ledger.

- Smart contracts are a type of Ethereum account.
  - **have a balance**
  - **can send transactions over the network**
  - not controlled by a user, run as programmed

- User accounts interact with a smart contract by **submitting transactions** that **execute a function** defined on the smart contract.

Ethereum Virtual Machine - EVM

- Ethereum implements a **distributed state machine / replicated state machine**.
- Ledger data holds the state - accounts, balances, and other variables.
- Transactions deterministically change the machine from one state to another.



Compiler

Smart Contract Code
- **Solidity**
- **Vyper**

Compiler

```
pragma solidity 0.4.24;

contract Greeter {

    function greet() public constant returns (string) {
        return "Hello";
    }

}
```

Bytecode

```
PUSH1 0x80 PUSH1 0x40 MSTORE PUSH1 0x4 CALLDATASIZE LT PUSH2 0x41
JUMPI PUSH1 0x0 CALLDATALOAD PUSH29
0x100000000000000000000000000000000000000000000000000000000
DIV PUSH4 0xFFFFFFFF AND DUP1 PUSH4 0xCFAE3217 EQ PUSH2 0x46 JUMPI
JUMPDEST PUSH1 0x0 DUP1 REVERT JUMPDEST CALLVALUE DUP1 ISZERO
PUSH2 0x52 JUMPI PUSH1 0x0 DUP1 REVERT JUMPDEST POP PUSH2 0x58
PUSH2 0xD6 JUMP JUMPDEST PUSH1 0x40 MLOAD DUP1 DUP1 PUSH1 0x20 ADD
DUP3 DUP2 SUB DUP3 MSTORE DUP4 DUP2 DUP2 MLOAD DUP2 MSTORE PUSH1
```

Deploy

Smart Contract

Application Binary Interface (ABI)

Web3.js

Execute

https://vyper.readthedocs.io/en/stable/
https://docs.soliditylang.org/en/v0.8.9/

Solidity

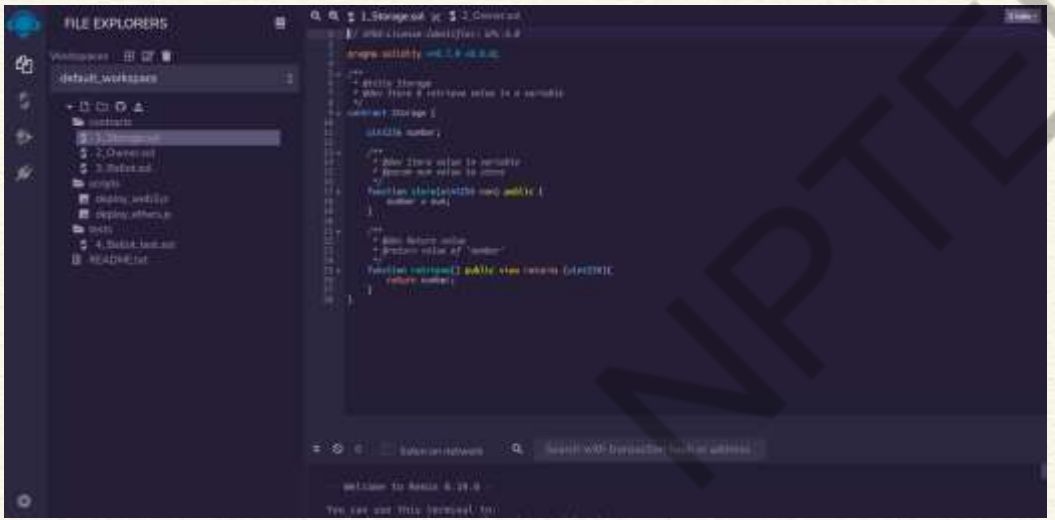
- High-level language for implementing smart contracts
    - Statically typed
    - Supports inheritance
    - Libraries
    - Complex user-defined types
- Syntax influenced by Javascript and C++

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.16 <0.9.0;

contract SimpleStorage {

uint storedData;

function set(uint x) public {
storedData = x;
 }

function get() public view returns (uint) {
 return storedData;
}

}
```

https://docs.soliditylang.org/en/v0.8.9/

Compiling Solidity

- **solc compiler**
  - Compile .sol files to bytecode
  - Command line tool
- **Remix editor**
  - Browser based
  - Compile, emulate, deploy contracts
  - https://remix.ethereum.org/



https://docs.soliditylang.org/en/v0.8.9/installing-solidity.html#installing-solidity

Simple Storage Contract

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

contract Storage {

        uint256 positivenumber;

        function store(uint256 inputnumber) public {
                positivenumber = inputnumber;
        }

        function readdata() public view returns (uint256){
                return positivenumber;
        }

}
```

Executing with Web3

```
var Web3 = require('web3');
var Contract = require('web3-eth-contract');

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

var myContract = new Contract(<ABI>,

"0xb3f36458FFc0C686DB4f2FF6002a55bFD85C03C8",
{
from: "0xd84a0607843b28c3f468857f82f784d9ff743bf8",
gasPrice: "20000000000"
}

);


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

Read data

Executing with Web3

```
var Web3 = require('web3');
var Contract = require('web3-eth-contract');

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

var myContract = new Contract(<ABI>,

"0xb3f36458FFc0C686DB4f2FF6002a55bFD85C03C8",
{
from: "0xd84a0607843b28c3f468857f82f784d9ff743bf8",
gasPrice: "20000000000"
}

);


myContract.methods.store("11").send().then(function (output) { console.log(output) });
```

Write data

Conclusion

- Ethereum smart contracts
- Remix editor
- Executing smart contracts from Web3 for DAPPS