Development session:

INTRODUCTION TO WEB3JS

7 pm – 8.30 pm 4 Jan

Cuong Truong Cong

BLOCKCHAIN
AT NTU

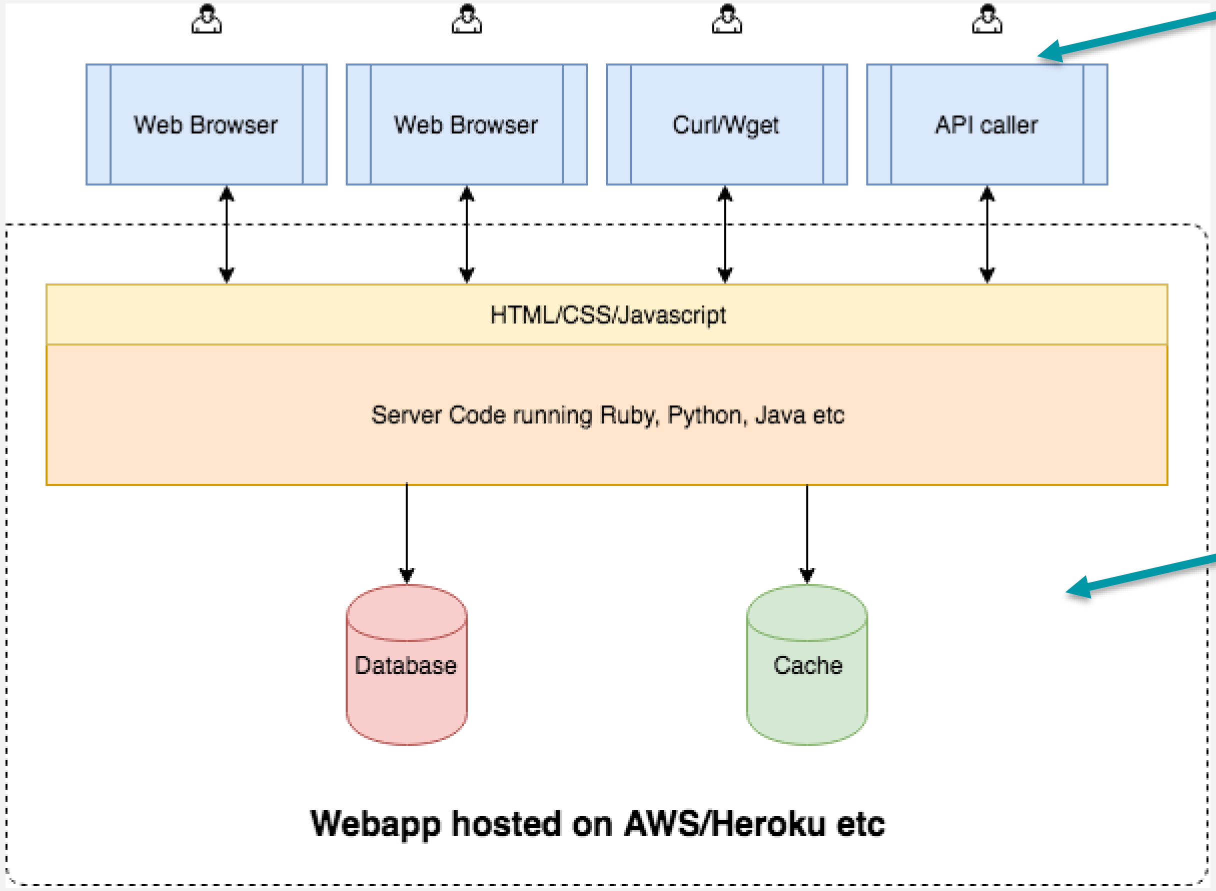
Agenda

- ❑ Ethereum DApp Architecture
- ❑ Web3JS Overview
- ❑ Web3JS Usage
- ❑ Development pillar

Ethereum DApp Architecture

BLOCKCHAIN AT NTU

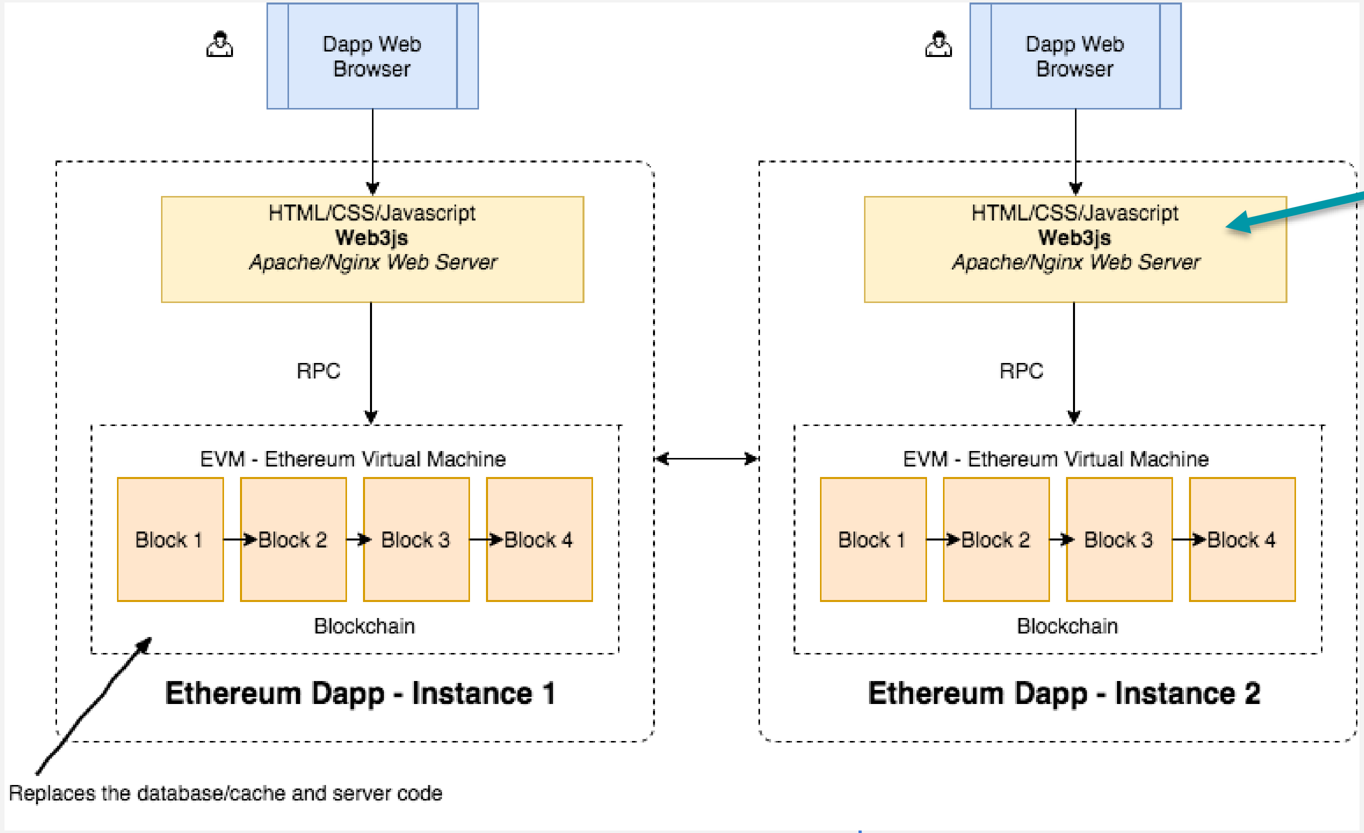
STANDARD WEB


Web Browser
Web Browser
Curl/Wget
API caller
HTML/CSS/Javascript
Server Code running Ruby, Python, Java etc
Database
Cache
Webapp hosted on AWS/Heroku etc

User interaction here

Centralized WebApp

ETHEREUM ARCHITECTURE



What we will discuss today

Source: https://www.zastrin.com/courses/ethereum-primer/lessons/1-5

Web3 Overview

WHAT IS WEB3?

Poke poke

web3

a collection of libraries that allows a user to **interact** with the Ethereum platform and smart contracts

DIFFERENT IMPLEMENTATIONS

Web3 Dot…

Python Web3.py

Haskell hs-web3

Java web3j

Easy to make, most used languages now have some kind of implementation in progress, but we will mainly focus on:

JavaScript web3.js

WHAT CAN WEB3.JS DO?
Everything

- It is the official Dapp API that is run on all the Ethereum nodes
- Main Capabilities:
  - ↳ Interact with contract functions
  - ↳ Interfaces with Ethereum nodes from the network using JSON-RPC calls
  - ↳ Send data to contracts and initiate transactions
  - ↳ Query the blockchain for data (includes logged events by any contract along with block data)
- Can have our back-end on chain, and our interface off chain

BLOCKCHAIN AT NTU

WHAT IS RPC?

Poke poke

RPC

Remote Procedure Call (RPC) is a protocol that one program can use to **request a service** from a program located in another computer in a network without having to understand network details

JSON-RPC API

Context behind Ethereum's API

- JSON is a lightweight data-interchange format
- RPC is used to make remote function calls
- JSON-RPC is a stateless, lightweight remote procedure call (RPC) protocol
    - ↳ Defines several data structures and the rules around their processing.
    - ↳ It is transport agnostic in that the concepts can be used within the same process, over sockets, over HTTP, or in many various message passing environments

ETHEREUM ABI

Exposing contract methods

ABI = Application Binary Interface

- An ABI is how you **call** functions in a contract and **get data back**
  - ↳ It determines how functions are called and in which binary format information should be passed from one program component to the next
- Why is it necessary?
  - ↳ You need a way to specify **which function** in the contract to invoke as well as guarantee to **what type of data is returned**
- Not part of the core Ethereum protocol, you can define your own ABI – but easier to comply with format provided by web3.js

ETHEREUM ABI

Another analogy

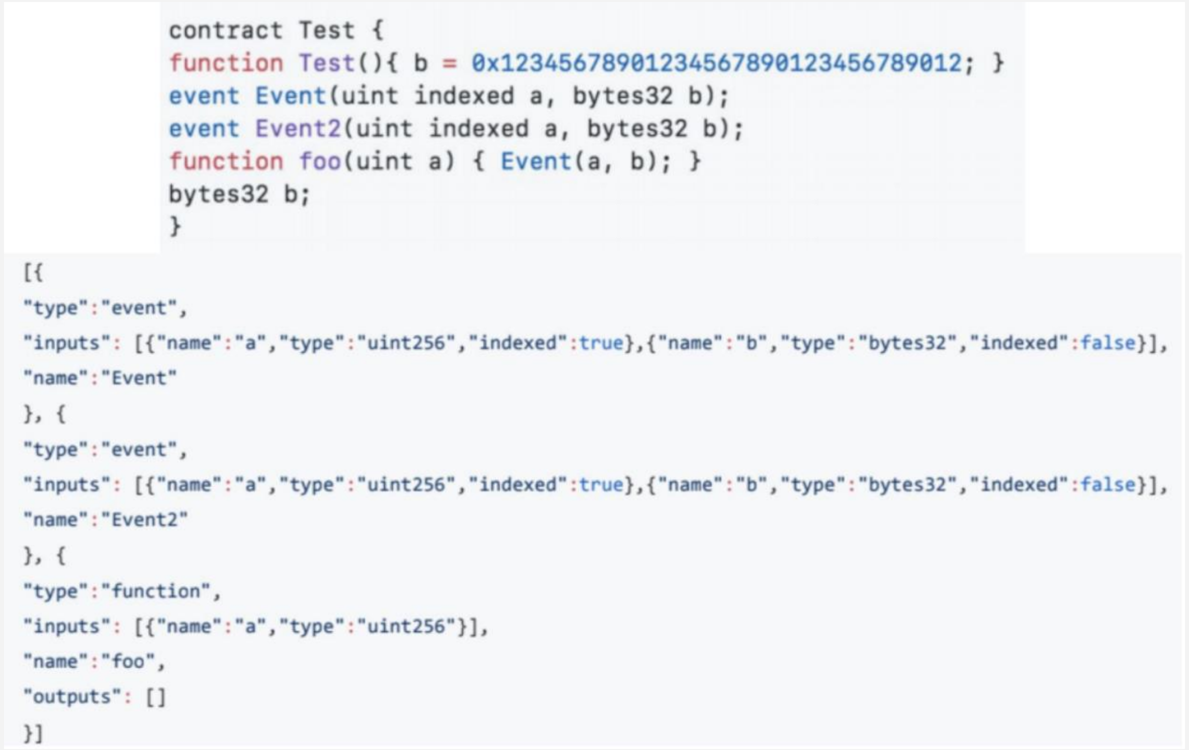
API = Application **Programming** Interface

ABI = Application **Binary** Interface

- So therefore an ABI is an API at a lower level?
- Contract code is stored as bytecode in binary form on the blockchain under a specific address
    - ↳ You can access the binary data in the contract through the ABI
    - ↳ The ABI defines the structures and methods you will use to interact with binary contract (just like an API)

ETHEREUM ABI

The lovely ABI format



```
contract Test {
function Test(){ b = 0x12345678901234567890123456789012; }
event Event(uint indexed a, bytes32 b);
event Event2(uint indexed a, bytes32 b);
function foo(uint a) { Event(a, b); }
bytes32 b;
}
```

```
[{
"type":"event",
"inputs": [{"name":"a","type":"uint256","indexed":true},{"name":"b","type":"bytes32","indexed":false}],
"name":"Event"
}, {
"type":"event",
"inputs": [{"name":"a","type":"uint256","indexed":true},{"name":"b","type":"bytes32","indexed":false}],
"name":"Event2"
}, {
"type":"function",
"inputs": [{"name":"a","type":"uint256"}],
"name":"foo",
"outputs": []
}]
```

Web3.js Usage

BLOCKCHAIN
AT NTU

INSTALLING DEPENDENCIES

Node Package Manager: node –v

Web3.js Library: npm install web3

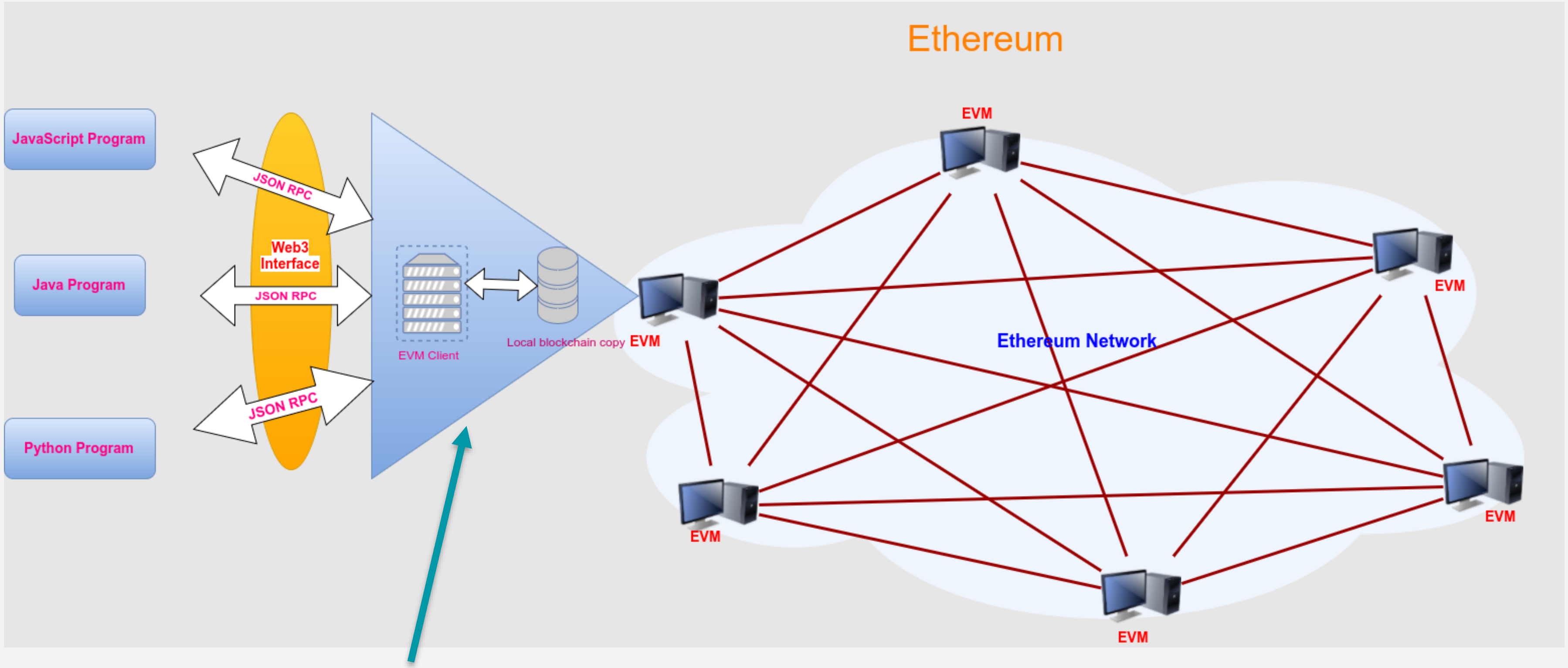
npm install dotenv

CONNECTING TO BLOCKCHAIN NODE

- Web3.JS is our key to reading and writing the Ethereum Blockchain
- Uses JSON RPC to talk to chain (Remote Procedure Call)
- Need to connect to an Ethereum node – can use:
  - ↳ Can run your own Ethereum node with **Geth** or **Parity**. However, this requires you to download a lot of data from blockchain and keep it in sync
  - ↳ **Infura** provides a remote Ethereum node for free and is more convenient

BLOCKCHAIN AT NTU

CONNECT TO BLOCKCHAIN NODE



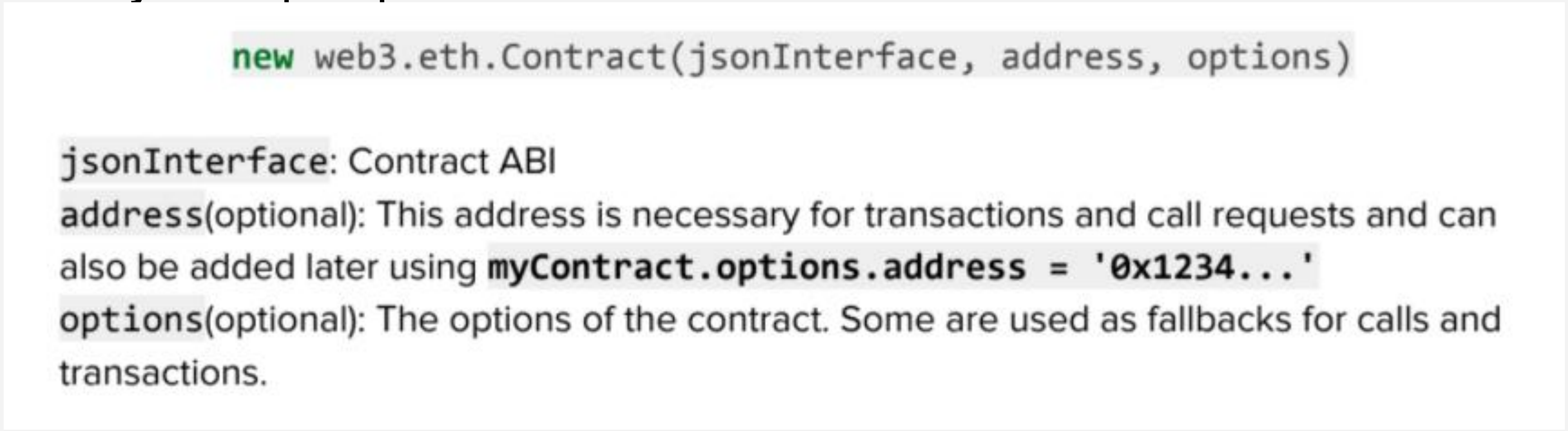
Infura here

CHECKING ACCOUNT BALANCE

Demo

READ DATA FROM SMART CONTRACT

1. A JavaScript representation of the smart contract we want to interact with



```
new web3.eth.Contract(jsonInterface, address, options)
```

`jsonInterface`: Contract ABI
`address`(optional): This address is necessary for transactions and call requests and can also be added later using **`myContract.options.address = '0x1234...'`**
`options`(optional): The options of the contract. Some are used as fallbacks for calls and transactions.

2. A way to call the functions on the smart contract when reading the data

```
contract.methods.myFunction([arguments]).call()
```

READ DATA FROM SMART CONTRACT

Demo

INSIDE ETHEREUM TRANSACTIONS

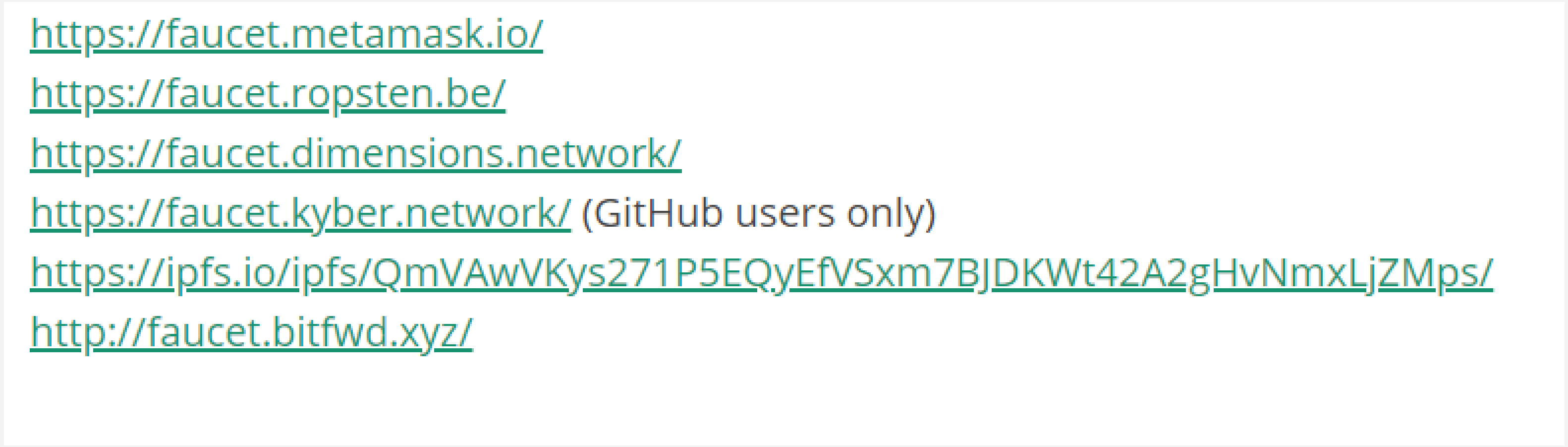
- Whenever we create a transaction, we are writing data to the blockchain and updating its state
- There are several ways to do this
  - ↳ Send Ether from one account to another
  - ↳ Call a smart contract function that writes data
  - ↳ Deploy a smart contract to the blockchain
- In order to broadcast transactions to the network, we'll need to sign them locally first.

```
npm install ethereumjs-tx
```

INSIDE ETHEREUM TRANSACTIONS

Let's get some fake Ether!

- Use the Ropsten test network
- Obtain the fake Ether from the following faucet:



https://faucet.metamask.io/
https://faucet.ropsten.be/
https://faucet.dimensions.network/
https://faucet.kyber.network/ (GitHub users only)
https://ipfs.io/ipfs/QmVAwVKys271P5EQyEfVSxm7BJDKWt42A2gHvNmxLjZMps/
http://faucet.bitfwd.xyz/

If all the link for Ropsten does not work, you can use this link for kovan network: https://faucet.kovan.network/ and switch to Kovan network later in the codes

BLOCKCHAIN AT NTU

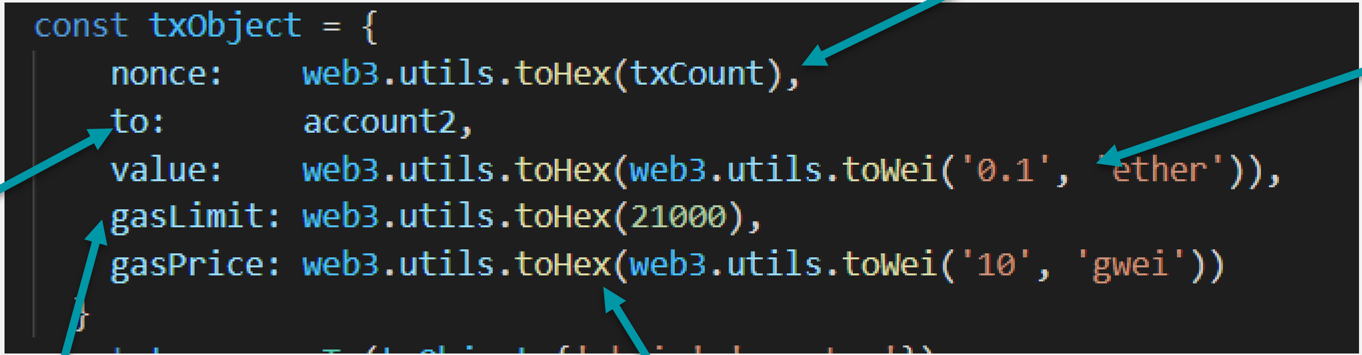
INSIDE ETHEREUM TRANSACTIONS

- Four steps to get your transaction done
  - ↳ Build a transaction object
  - ↳ Sign the transaction
  - ↳ Broadcast the transaction to the network
  - ↳ Wait for magic to happen on the chain ☺

INSIDE ETHEREUM TRANSACTIONS

Build your transaction object

Previous transaction count for the given account



```
const txObject = {
    nonce:    web3.utils.toHex(txCount),
    to:       account2,
    value:    web3.utils.toHex(web3.utils.toWei('0.1', 'ether')),
    gasLimit: web3.utils.toHex(21000),
    gasPrice: web3.utils.toHex(web3.utils.toWei('10', 'gwei'))
}
```

The amount of Ether we want to send, must be expressed in Wei and converted to hexadecimal

The account we're sending Ether to

The maximum amount of gas consumed by the transaction. A basic transaction like this always cost 21000 units of gas

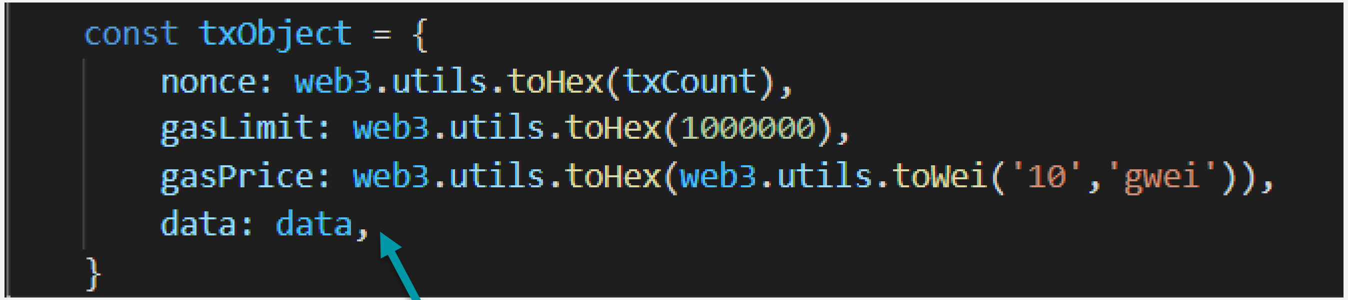
The amount we want to pay for each unit of gas. 10 Gwei is used here

INSIDE ETHEREUM TRANSACTIONS

Demo

DEPLOYING SMART CONTRACTS

Transaction object and demo



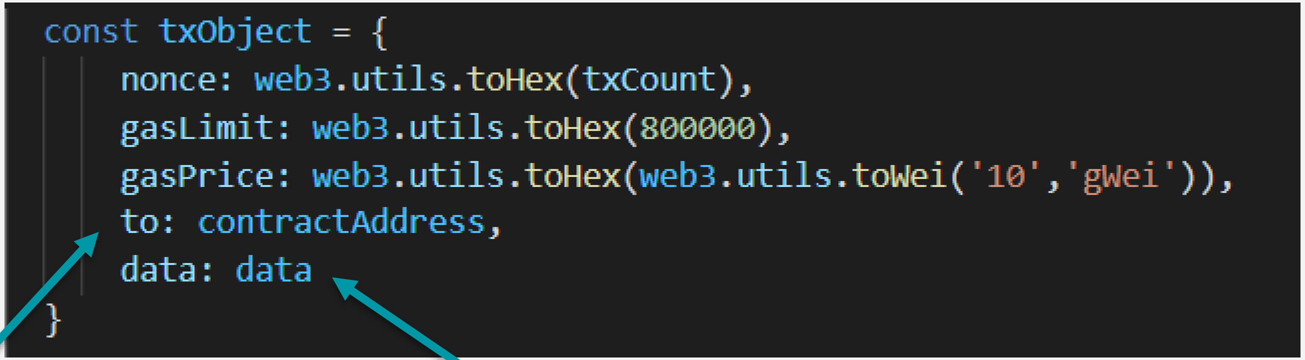
```
const txObject = {
    nonce: web3.utils.toHex(txCount),
    gasLimit: web3.utils.toHex(1000000),
    gasPrice: web3.utils.toHex(web3.utils.toWei('10','gwei')),
    data: data,
}
```

Bytecode of the smart contract that we want to deploy

BLOCKCHAIN AT NTU

CALLING SMART CONTRACT FUNCTION

Transaction object and demo



```
const txObject = {
    nonce: web3.utils.toHex(txCount),
    gasLimit: web3.utils.toHex(800000),
    gasPrice: web3.utils.toHex(web3.utils.toWei('10','gWei')),
    to: contractAddress,
    data: data
}
```

The address of the deployed contract

Hexadecimal representation of the function we want to call on the smart contract

Source code and references

Github repo:

https://github.com/cuongquangnam/web3_tutorial_BNS

References:

Blockchain at Berkeley slides:

https://drive.google.com/file/d/1v-hEK5Py7xgoj5s979zx1jS9lQ86jSc_/view

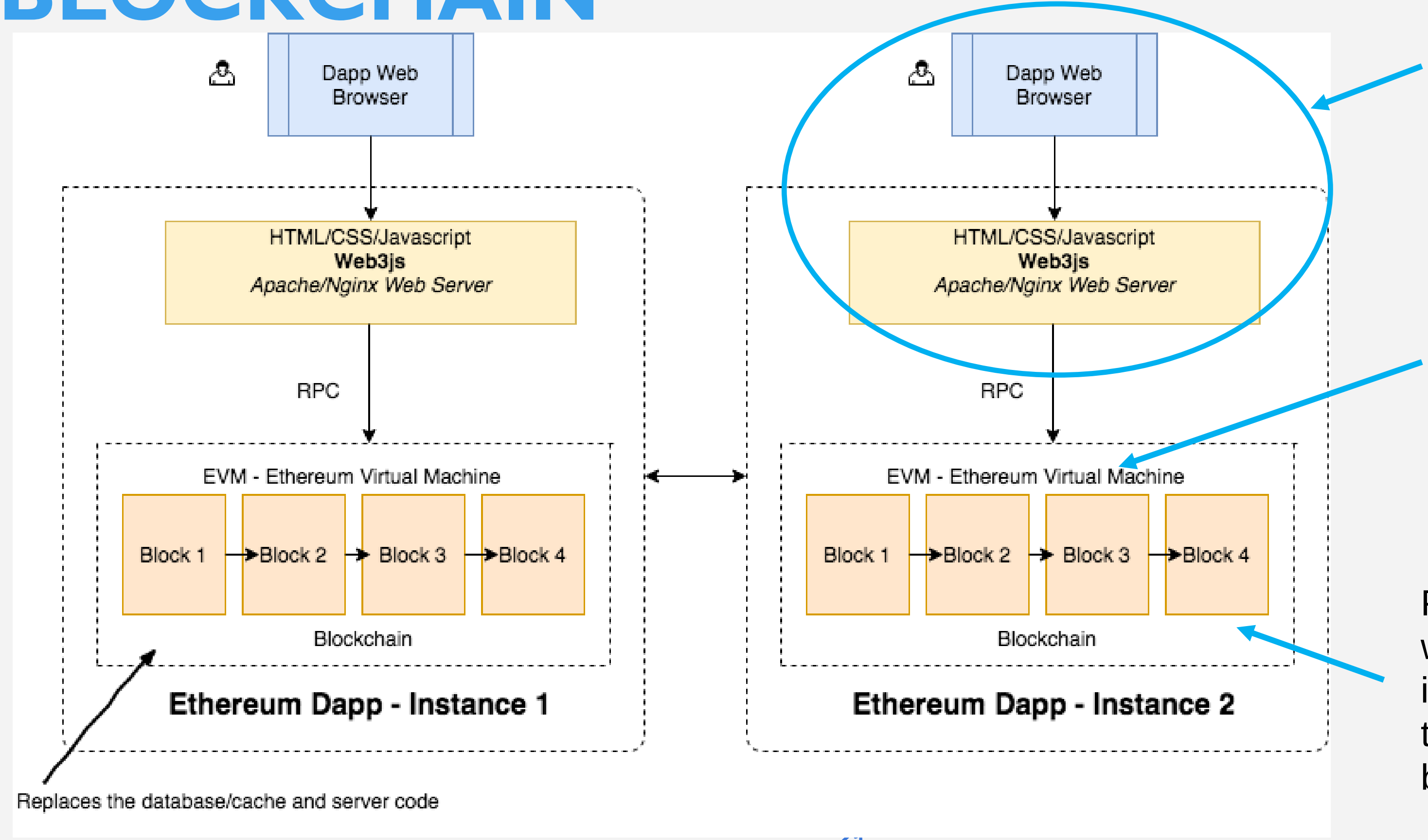
Introduction to Web3.js – DApp university (my demo is based on this)

https://www.dappuniversity.com/articles/web3-js-intro

Dev pillar

BLOCKCHAIN AT NTU

MY VIEWS OF POPULAR JOBS IN BLOCKCHAIN



Front-end/ backend engineer

Smart contract stays here on blockchain here and smart contract engineer works with this

Protocol engineer works on the implementation of the rules of blockchain

MY VIEWS OF POPULAR JOBS IN BLOCKCHAIN

- Besides, we also have researchers, security auditors of smart contracts/ protocols, etc.

PROJECTS

- Mostly build a dApp (but you can suggest any idea of your own)
- Do it at your pace (but hopefully we can have something out at the end of this semester)
- Will try to mentor or find a mentor for you guys

Example works

- Popular dApps: https://www.stateofthedapps.com/, https://www.dapp.com
- My development:
  - ↳ Contact Tracing: https://www.youtube.com/watch?v=aWEqCqk5xwk
  - ↳ Dutch Auction: https://www.youtube.com/watch?v=49oLh1VYZfE&t=1s

Topics to choose

https://drive.google.com/file/d/13tuL9hdxiKMq3Fzn0wMdaM0sCGypu0iN/view

Resources

- CryptoZombie: https://cryptozombies.io/
- Ethereum and Solidity on Udemy: https://www.udemy.com/course/ethereum-and-solidity-the-complete-developers-guide/
- dApp university: https://www.dappuniversity.com/
- Berkeley courses: https://blockchain.berkeley.edu/courses/archive/

Join our dev pillar!!!

SCAN ME

https://t.me/joinchat/HSRsP6vg4_VUOVqz

BLOCKCHAIN
AT NTU

Thank You!

🔨 with 💙 by

👷 Cuong Truong Cong

💬 @williamvn or t.me/ntublockchain

✉ BlockchainNTU@e.ntu.edu.sg