BLOCKCHAIN APP DEVELOPMENT: BUILDING ON ETHEREUM

Nishaad Navkal

Introduction

Nishaad Navkal

- he/him/his
- 4th year, economics major, CS minor, SCET certificate
- PM of Education at Blockchain at Berkeley, member since Spring 2017
- Energy and Blockchain Analyst at EDF, inc.

What will be covered in this lecture?

- **Tools used in (backend) Ethereum development + what they do**
  - Solidity
  - Truffle/Ganache
  - Remix IDE
  - MetaMask
- **Developing 101**
  - Anatomy of a smart contract
  - Writing a basic smart contract + interacting with it

prep work

If you haven’t done this yet, please follow the instructions linked here regarding programs/packages to download, etc.

https://tinyurl.com/yd3bur25

ETHEREUM

What is it?

- **Public blockchain with a Turing complete smart contract programming language**
  - **Public blockchain**: no barriers to entry, anyone can read from or write to the chain, anyone can validate blocks, etc. contrasted with private and permissioned chains
  - **Smart contracts**: contracts that execute themselves in a predetermined way → basically think of a legal document but its execution is enforced by computational logic.
- **Native cryptocurrency: Ether**

dAPPS - HIGH LEVEL

What are they? How do they work?

dAPP: Decentralized Application:

- Front end web page (with HTML + CSS + JavaScript)
  - Can be stored on your own computer or using a decentralized storage network such as IPFS
- Back end the decentralized p2p network you're using, plus the relevant smart contracts deployed on it
  - Ethereum Virtual Machine (EVM) - think of it as one big computer for now

dAPPS - HIGH LEVEL

What are they? How do they work?

- **Front end:** The content that the user sees, arrangement of website, etc.
- **Back end:** The functionality of the website

dAPPS - HIGH LEVEL

What are they? How do they work?

- **Front end:** The content that the user sees, arrangement of website, etc.
- **Back end:** The functionality of the website

<u>App concept: Tomato Soup</u>

- Back end: the tomato puree, tomatoes, water(?), ketchup(?)
- Front end: Salt, pepper, basil(?), onions(?)

dAPPS - HIGH LEVEL

What are they? How do they work?

- **Back end:** The functionality of the website
- **Front end:** The content that the user sees, arrangement of website, etc.

App concept: Tomato Soup

- Back end: the tomato puree, tomatoes, water(?), ketchup(?)
- Front end: Salt, pepper, basil(?), onions(?)

Takeaways:

- Both parts are necessary
- I don’t make a lot of soup

Let’s dive in!!

Truffle

- **Framework/development environment for smart contracts**
    - Launching smart contracts can be tedious, and requires a lot of files that can be difficult to keep track of - truffle makes that a lot easier, and also provides a test blockchain (Ganache) for us to use, as well as a simple interface for testing our smart contract

SOLIDITY

Standard coding stuff but also:

- Events ➔ logged to the blockchain, can be read from it to determine what happened
- Modifiers ➔ what is necessary for a function to be used
- Structs ➔ like objects
- Addresses ➔ these represent identities on a blockchain
- msg.sender ➔ the address that called the function
- msg.value ➔ the ETH sent in the function call

[show Apples.sol]

Let’s deploy our own smart contract!!

USING BLOCKCHAIN TO MAKE [A] BANK

We want to be able to:

- Deposit money (ETH)
- Withdraw money (but only up to the amount we’ve deposited)
- Check our balance

Let’s deploy our own smart contract!!

USING BLOCKCHAIN TO MAKE [A] BANK

- Open your terminal
- Create a new directory somewhere, call it whatever you want, maybe myProject, myBank, whatever ➔ mkdir <name>
- Go into that directory ➔ cd <name>
- truffle init

Let’s deploy our own smart contract!!

USING BLOCKCHAIN TO MAKE [A] BANK

- Open text editor, open the project folder you just made
- Make a new file in the contracts directory: bank.sol
- pragma solidity ^0.4.17;
  - This goes at the top of every solidity file, denotes the version of solidity you want to use.
  - <u>every instruction should be ended with a semicolon</u>
- contract <contractname> {

  }

Let’s deploy our own smart contract!!

USING BLOCKCHAIN TO MAKE [A] BANK

- Declare your variables!!
    - Solidity defaults all uninitialized variables to zero.
- contract <bank, or whatever you called it> {

```
        mapping(address => uint256) balances;
}
```

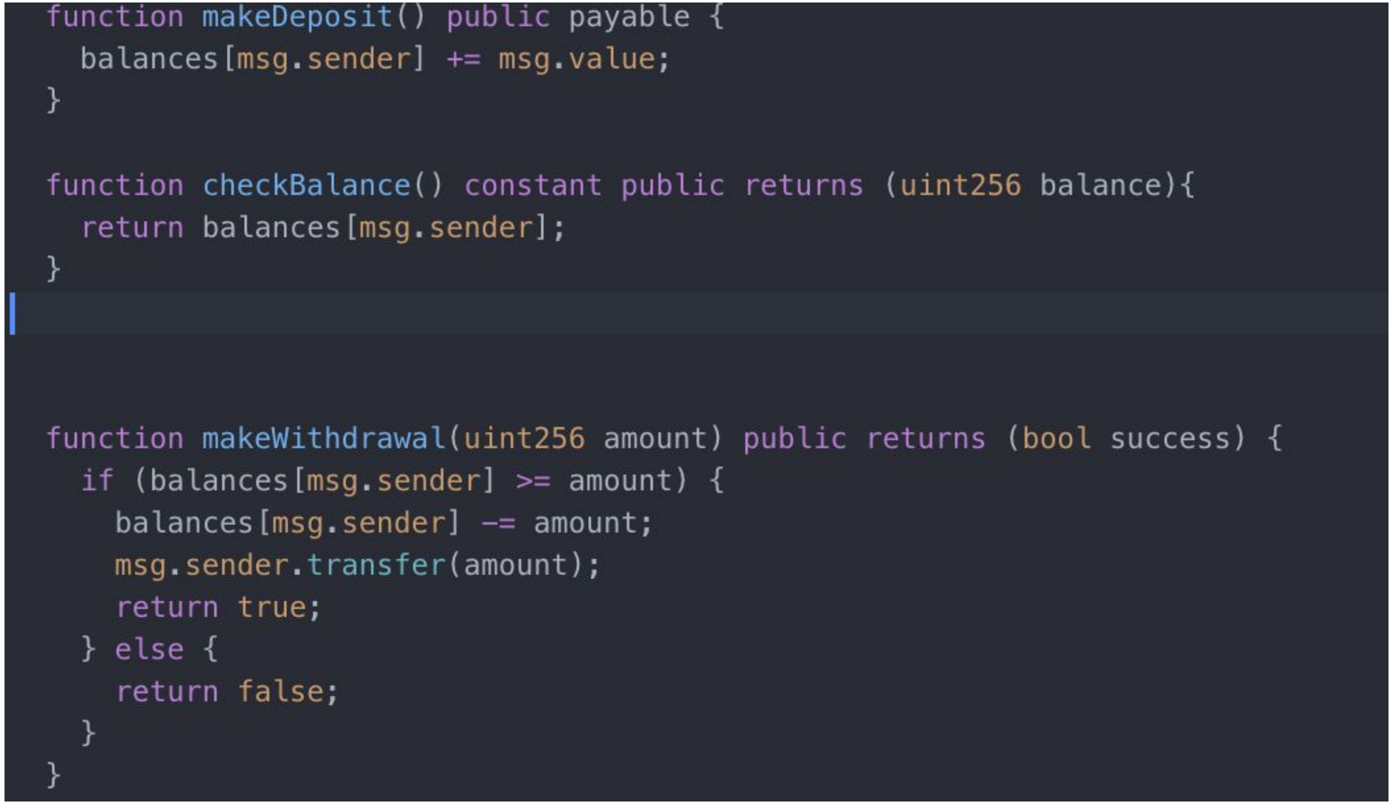
Let’s deploy our own smart contract!!

USING BLOCKCHAIN TO MAKE [A] BANK

```
function makeDeposit() public payable {


  }
  function checkBalance() constant public returns (uint256 balance){


  }
 function withdraw(uint256 amount) public returns (bool success){


  }
```

Let’s deploy our own smart contract!!

USING BLOCKCHAIN TO MAKE [A] BANK

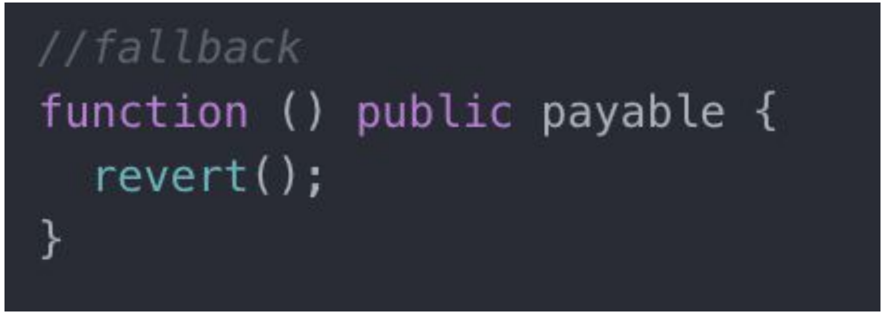


```
function makeDeposit() public payable {
  balances[msg.sender] += msg.value;
}

function checkBalance() constant public returns (uint256 balance){
  return balances[msg.sender];
}



function makeWithdrawal(uint256 amount) public returns (bool success) {
  if (balances[msg.sender] >= amount) {
    balances[msg.sender] -= amount;
    msg.sender.transfer(amount);
    return true;
  } else {
    return false;
  }
}
```

Let’s deploy our own smart contract!!

USING BLOCKCHAIN TO MAKE [A] BANK

Fallback function: called if an entity calls a function that is not defined in the contract



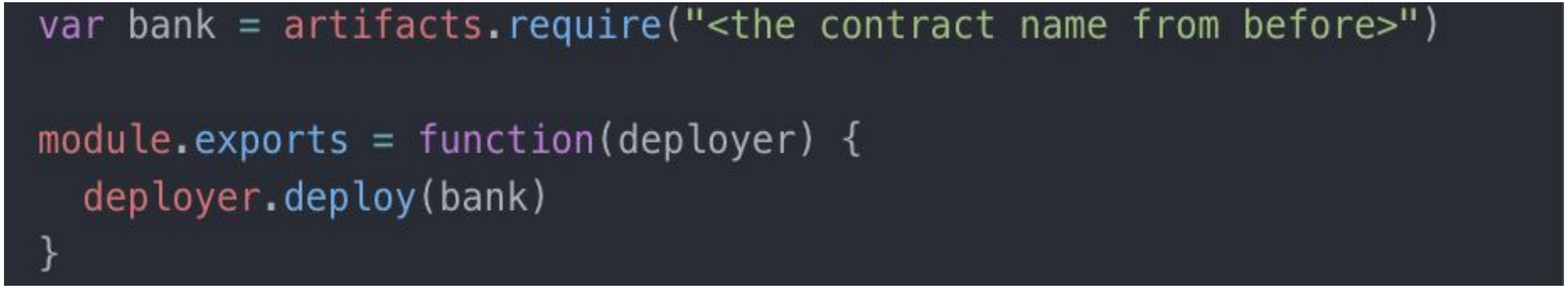
```
//fallback
function () public payable {
  revert();
}
```

You just wrote your first smart contract

Congrats!!! Now how do we use it?

Truffle minutia

- Create a file in the migrations folder
    - 2_deploy_contracts.js

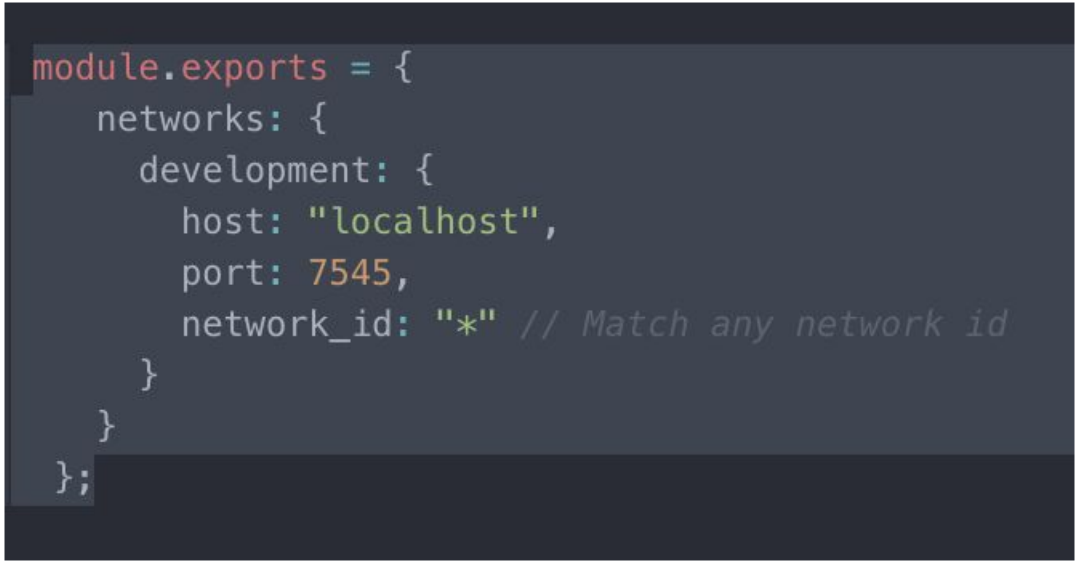


```
var bank = artifacts.require("<the contract name from before>")

module.exports = function(deployer) {
  deployer.deploy(bank)
}
```

You just wrote your first smart contract

Congrats!!! Now how do we use it?

- Open truffle.js
- Paste this in



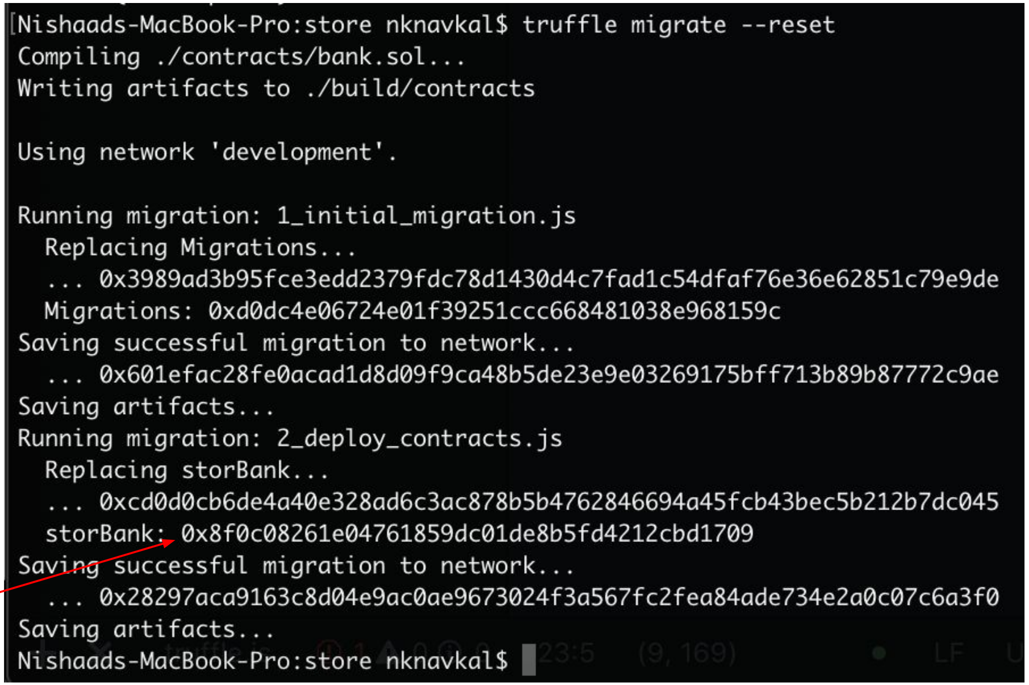
```
module.exports = {
  networks: {
    development: {
      host: "localhost",
      port: 7545,
      network_id: "*" // Match any network id
    }
  }
};
```

You just wrote your first smart contract

Congrats!!! Now how do we use it?

Now we get to the fun part!!!

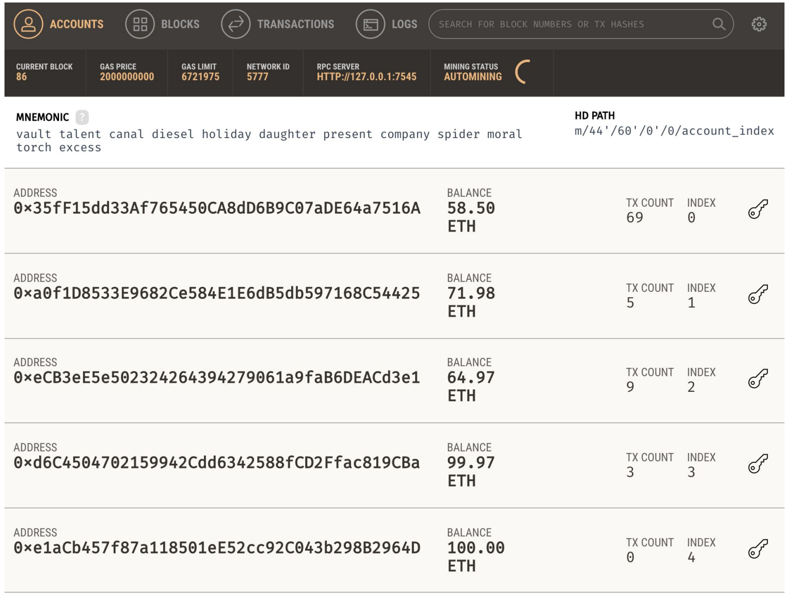
- Open up Ganache first
- Go into your project directory in terminal
- Truffle compile
- Truffle migrate --reset
- Look for the contract name, copy the address next to it
- **Contract address!!!!**
- **You just deployed a smart contract!**



```
Nishaads-MacBook-Pro:store nknavkal$ truffle migrate --reset
Compiling ./contracts/bank.sol...
Writing artifacts to ./build/contracts

Using network 'development'.

Running migration: 1_initial_migration.js
  Replacing Migrations...
  ... 0x3989ad3b95fce3edd2379fdc78d1430d4c7fad1c54dfaf76e36e62851c79e9de
  Migrations: 0xd0dc4e06724e01f39251ccc668481038e968159c
Saving successful migration to network...
  ... 0x601efac28fe0acad1d8d09f9ca48b5de23e9e03269175bff713b89b87772c9ae
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Replacing storBank...
  ... 0xcd0d0cb6de4a40e328ad6c3ac878b5b4762846694a45fcb43bec5b212b7dc045
  storBank: 0x8f0c08261e04761859dc01de8b5fd4212cbd1709
Saving successful migration to network...
  ... 0x28297aca9163c8d04e9ac0ae9673024f3a567fc2fea84ade734e2a0c07c6a3f0
Saving artifacts...
Nishaads-MacBook-Pro:store nknavkal$
```

Look at ganache

- Most of the addresses should have 100
- Top one slightly less - why?
    - Gas used in deploying a contract



CURRENT BLOCK	GAS PRICE	GAS LIMIT	NETWORK ID	RPC SERVER	MINING STATUS
86	20000000000	6721975	5777	HTTP://127.0.0.1:7545	AUTOMINING

MNEMONIC
vault talent canal diesel holiday daughter present company spider moral torch excess

HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0x35fF15dd33Af765450CA8dD6B9C07aDE64a7516A	58.50 ETH	69	0
0xa0f1D8533E9682Ce584E1E6dB5db597168C54425	71.98 ETH	5	1
0xeCB3eE5e502324264394279061a9faB6DEACd3e1	64.97 ETH	9	2
0xd6C4504702159942Cdd6342588fCD2Ffac819CBa	99.97 ETH	3	3
0xe1aCb457f87a118501eE52cc92C043b298B2964D	100.00 ETH	0	4

Play around with this for a while

- Truffle console

```
ContractName.at('ContractAddress').functionName(parameter1, parameter 2,
{from: 'senderAddress', value: amount})
```

- ***the units for value are wei, each wei is 1^-18 ETH (why?), so you have to type out 1000000000000000000 in that field to see a difference of 1 eth on ganache
- You don’t need to include the ‘value’ or ‘from’ fields unless you want, but try them out, see what happens

Ganache

What is it?

- Local testnet - you’re each using your own tiny blockchain
- Good for testing
  - One of the most important parts of development on the blockchain
  - Gas ➜ everything on chain is expensive
    - Main reason for testnets
    - Program conservatively
- Testnet ether has unlimited supply, so has no monetary value :(

Using Public networks

- Better because it is closer to the ethereum main network
- You need to be able to query the actual blockchain, and so does everyone else on this network
- Ethereum Clients
    - Geth
    - Parity

These are pretty serious, will eat up tons of memory and drain your battery in like 20 minutes

Infura

- They run an Ethereum node for you!!
- Infura.io
- Probably the best tool out there to deploy onto Ropsten (if you’re not running a node)

Public testnets

- Ropsten
  - Proof of Work
  - Most popular/real, supported by both Geth and Parity (EVM Clients)
- Rinkeby
  - Proof of Authority
  - Supported by Geth
- Kovan
  - Proof of Authority
  - Supported by Parity

IDEs

Alternatives to Truffle/Ganache

Remix is most popular

Remix.ethereum.org

Fully functional, handles all the migration stuff we had to do ourselves in truffle

What you develop on is entirely based personal preferences

- I think remix is a little intimidating, too much going on design wise for me
- In browser ➔ can't work as effectively offline

Metamask

- Chrome/firefox extension; wallet
- Gateway to web3.0 ➔ this allows you to interact with the decentralized web
- If you need to spend some eth to do something, metamask will open a popup, and then you put in your information

More tools

- Faucets
    - https://faucet.ropsten.be/
- Etherscan
- myetherwallet

Thanks!!!

I hope this was helpful!!!

Feel free to contact me with any questions, nknavkal@berkeley.edu

Also, shameless plug: join Blockchain at Berkeley!

This lecture will be posted, in addition to further resources