Ethereum Programming

Lecture by Prof. Duc A. Tran
University of Massachusetts, Boston, MA
duc.tran@umb.edu
https://linkedin.com/in/ductran

About this Tutorial

- I tried to
  - Keep it simple
  - Not throw out too many names
  - Discuss only those needed for the practice
  - Demo on Mac OS

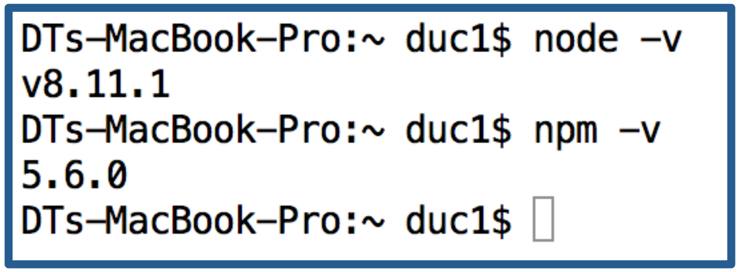
I learned a lot ~~myself~~ from

A 101 Noob Intro to Programming Smart Contracts on Ethereum

https://medium.com/@ConsenSys/a-101-noob-intro-to-programming-smart-contracts-on-ethereum-695d15c1dab4

JavaScript: install **nodejs**

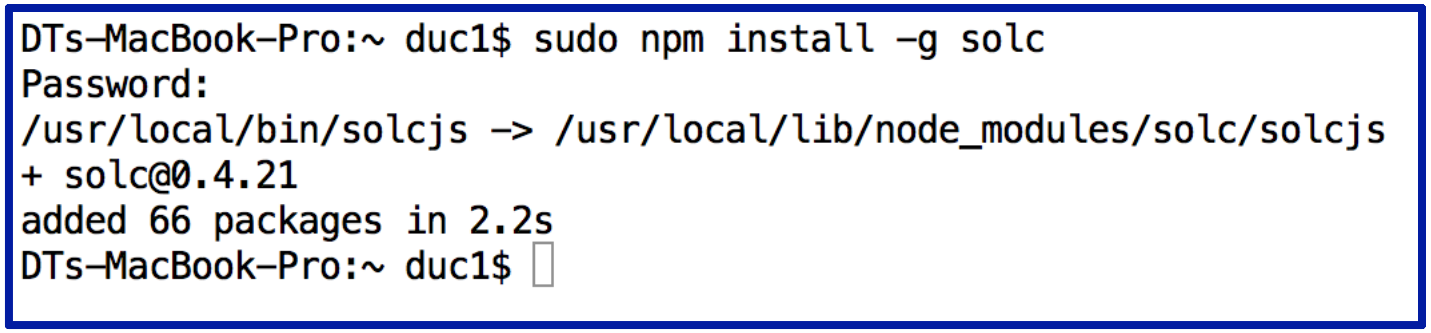
- An open-source, cross-platform JavaScript run-time environment that executes JavaScript code on server-side
- Download: nodejs.org (will place **Node.js** and **npm** both in **/usr/local/bin**)
- Has it been installed? type "**node –v**" and "**npm –v**" at command line to see the version



```
DTs-MacBook-Pro:~ duc1$ node -v
v8.11.1
DTs-MacBook-Pro:~ duc1$ npm -v
5.6.0
DTs-MacBook-Pro:~ duc1$ 
```

Smart Contracts: use **solidity**

- Most popular language for smart contracts
  - like JavaScript and has ".sol" as a file extension
- Install **solcjs** (a solidity compiler, from the C++ libraries)



```
DTs-MacBook-Pro:~ duc1$ sudo npm install -g solc
Password:
/usr/local/bin/solcjs -> /usr/local/lib/node_modules/solc/solcjs
+ solc@0.4.21
added 66 packages in 2.2s
DTs-MacBook-Pro:~ duc1$ 
```

IDE for Solidity

- … or you can use a Solidity IDE
- **REMIX**
  - By Ethereum
  - Online IDE: *http://remix.ethereum.org*
- **COSMO**
  - built in **MeteorJS** web framework
  - **MeteorJS** (web framework)
    - Free, open-source isomorphic JavaScript web framework, written using Node.js.
    - <u>Rapid prototyping</u> and produces cross-platform (Android, iOS, Web) code.

web3.js

- After smart contract enters the blockchain,

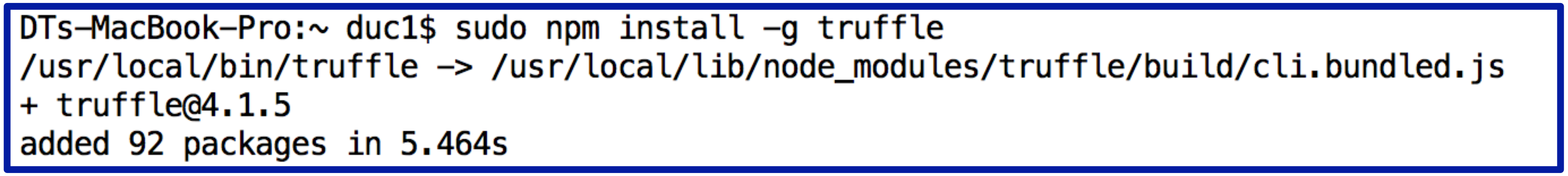
  <u>how to interact with the contract?</u>

- Use an API: Ethereum **web3.js** JavaScript API
  - A set of javascript libraries to allow us to interact with an Ethereum node, using HTTP or IPC

How to test? install **truffle**

- A test-driven development of smart contracts: <u>write-compile-deploy-test-build</u> DApps
  - Use a JavaScript promises framework (**pudding)** on top of **web3.js** (so it installs web3.js for you too).

- To install, type at command line



```
DTs-MacBook-Pro:~ duc1$ sudo npm install -g truffle
/usr/local/bin/truffle -> /usr/local/lib/node_modules/truffle/build/cli.bundled.js
+ truffle@4.1.5
added 92 packages in 5.464s
```

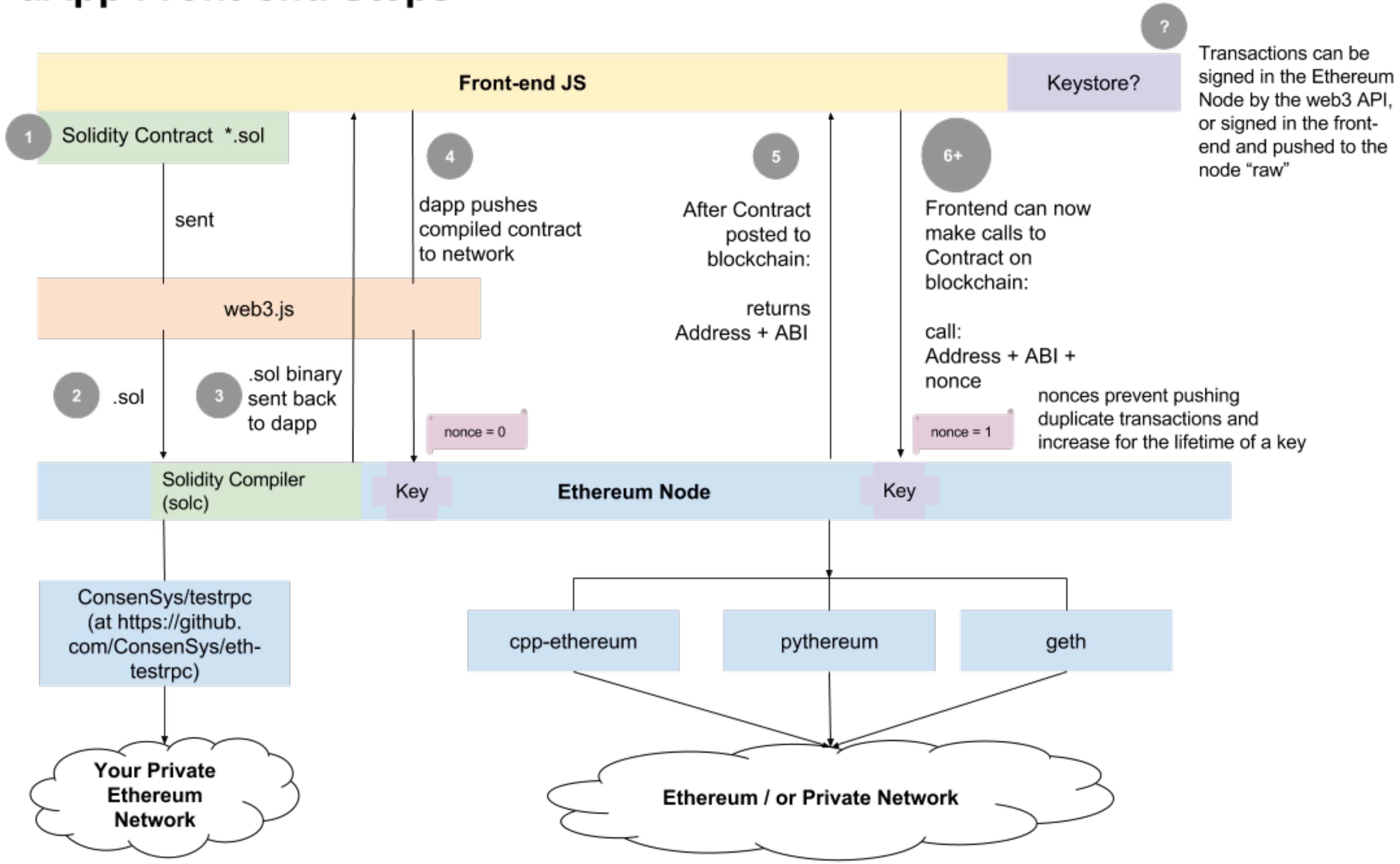
How to use a DApp?

- Option 1: run a local Ethereum node
  - Command line: several, depending on your preference for Go (**geth, most popular**), C++ (**eth**), Python (**pyethereum**), Java, or Haskell
    - For testing/development purposes: **ganache-cli**
  - GUI client: Ethereum's **AlethZero** or **AlethOne**

- Option 2: use a web browser without a node
  - **MetaMask**

dApp Front-end Steps

Source: medium.com



Front-end JS
Keystore?
?
Transactions can be signed in the Ethereum Node by the web3 API, or signed in the front-end and pushed to the node "raw"
1
Solidity Contract *.sol
sent
4
dapp pushes compiled contract to network
5
After Contract posted to blockchain:
returns Address + ABI
6+
Frontend can now make calls to Contract on blockchain:
call: Address + ABI + nonce
nonces prevent pushing duplicate transactions and increase for the lifetime of a key
web3.js
2
.sol
3
.sol binary sent back to dapp
nonce = 0
nonce = 1
Solidity Compiler (solc)
Key
Ethereum Node
Key
ConsenSys/testrpc (at https://github.com/ConsenSys/eth-testrpc)
cpp-ethereum
pythereum
geth
Your Private Ethereum Network
Ethereum / or Private Network

A **Contract Creation Transaction** is shown in steps 1-5 at above.

An **Ether Transfer** or **Function Call Transaction** is assumed in step 6.

Ready for An Example?

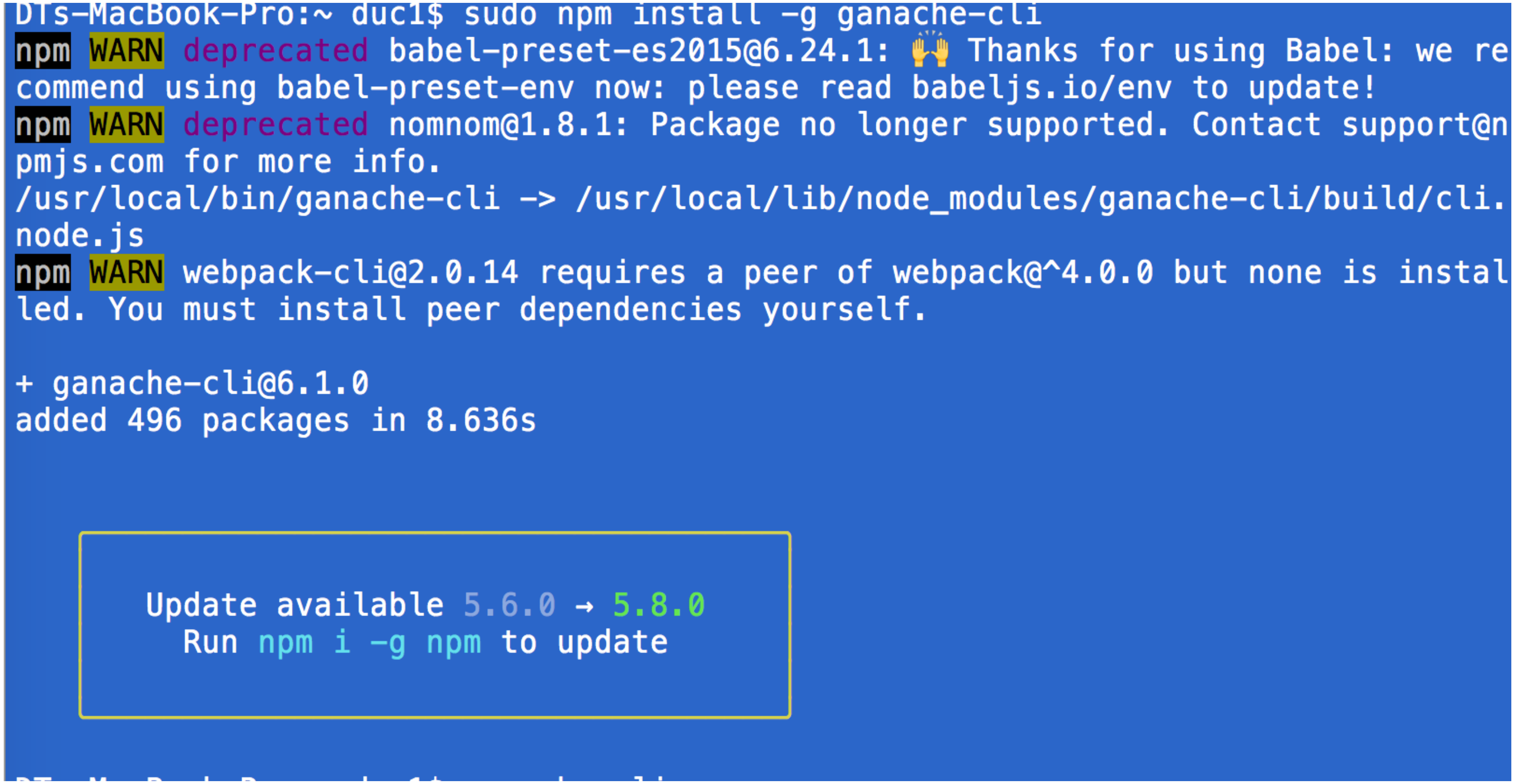
- We will need (in order):
  - Programming environment: Javascript (**node.js**)
  - Smart contract language: Solidity (**solc**)

    npm install solc
  - Ethereum development environment: **truffle**

    npm install –g truffle
  - To run a local Ethereum node: **ganache-cli**

    npm install -g ganache-cli
    - Note: this is the new name for "testrpc"
    - ganache-cli vs. geth:
      - ganache-cli: Node.js based Ethereum client for testing and development.
      - geth: full client in GO Language, connect to the real chain or start your own testnet server.

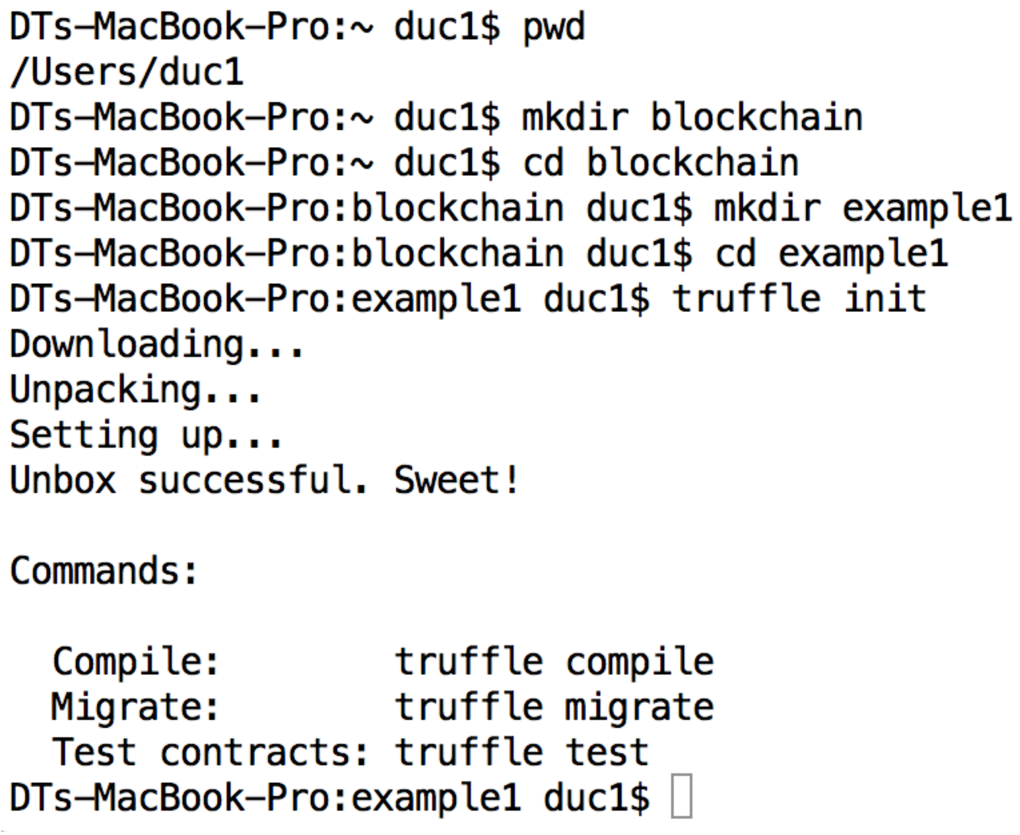
Install **ganache-cli**



```
DTs-MacBook-Pro:~ duc1$ sudo npm install -g ganache-cli
npm WARN deprecated babel-preset-es2015@6.24.1: 🙌 Thanks for using Babel: we re
commend using babel-preset-env now: please read babeljs.io/env to update!
npm WARN deprecated nomnom@1.8.1: Package no longer supported. Contact support@n
pmjs.com for more info.
/usr/local/bin/ganache-cli -> /usr/local/lib/node_modules/ganache-cli/build/cli.
node.js
npm WARN webpack-cli@2.0.14 requires a peer of webpack@^4.0.0 but none is instal
led. You must install peer dependencies yourself.

+ ganache-cli@6.1.0
added 496 packages in 8.636s



      Update available 5.6.0 → 5.8.0
        Run npm i -g npm to update
```

Preparation

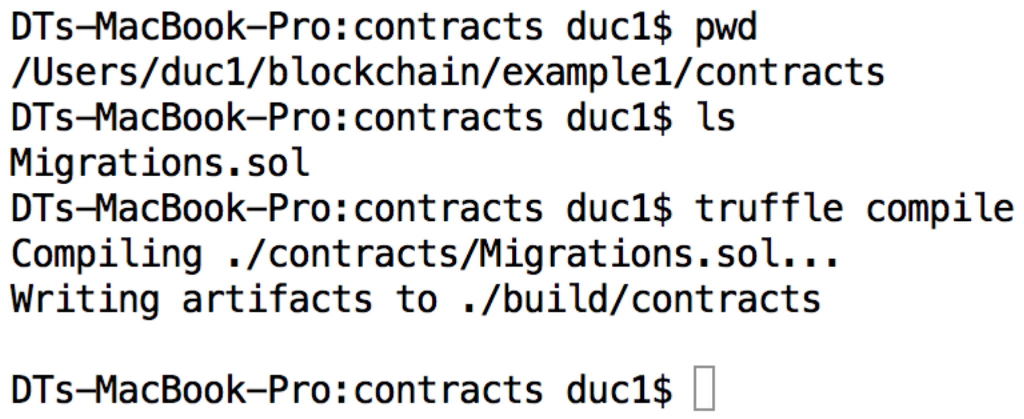
- <u>Create a folder</u> for our Ethereum project (anywhere)
- Go to this folder
- Run *"truffle init"* to <u>initialize</u> a new and empty project in this folder



```
DTs-MacBook-Pro:~ duc1$ pwd
/Users/duc1
DTs-MacBook-Pro:~ duc1$ mkdir blockchain
DTs-MacBook-Pro:~ duc1$ cd blockchain
DTs-MacBook-Pro:blockchain duc1$ mkdir example1
DTs-MacBook-Pro:blockchain duc1$ cd example1
DTs-MacBook-Pro:example1 duc1$ truffle init
Downloading...
Unpacking...
Setting up...
Unbox successful. Sweet!

Commands:

  Compile:        truffle compile
  Migrate:        truffle migrate
  Test contracts: truffle test
DTs-MacBook-Pro:example1 duc1$
```

Coding & Compiling

- Coding
  - Write the Solidity code for our smart contract
  - Save as a file, e.g., “*whatever.sol*”, in folder “*contracts/* “
- Compiling
  - type at the command line: *“truffle compile”*



```
DTs-MacBook-Pro:contracts duc1$ pwd
/Users/duc1/blockchain/example1/contracts
DTs-MacBook-Pro:contracts duc1$ ls
Migrations.sol
DTs-MacBook-Pro:contracts duc1$ truffle compile
Compiling ./contracts/Migrations.sol...
Writing artifacts to ./build/contracts

DTs-MacBook-Pro:contracts duc1$ 
```

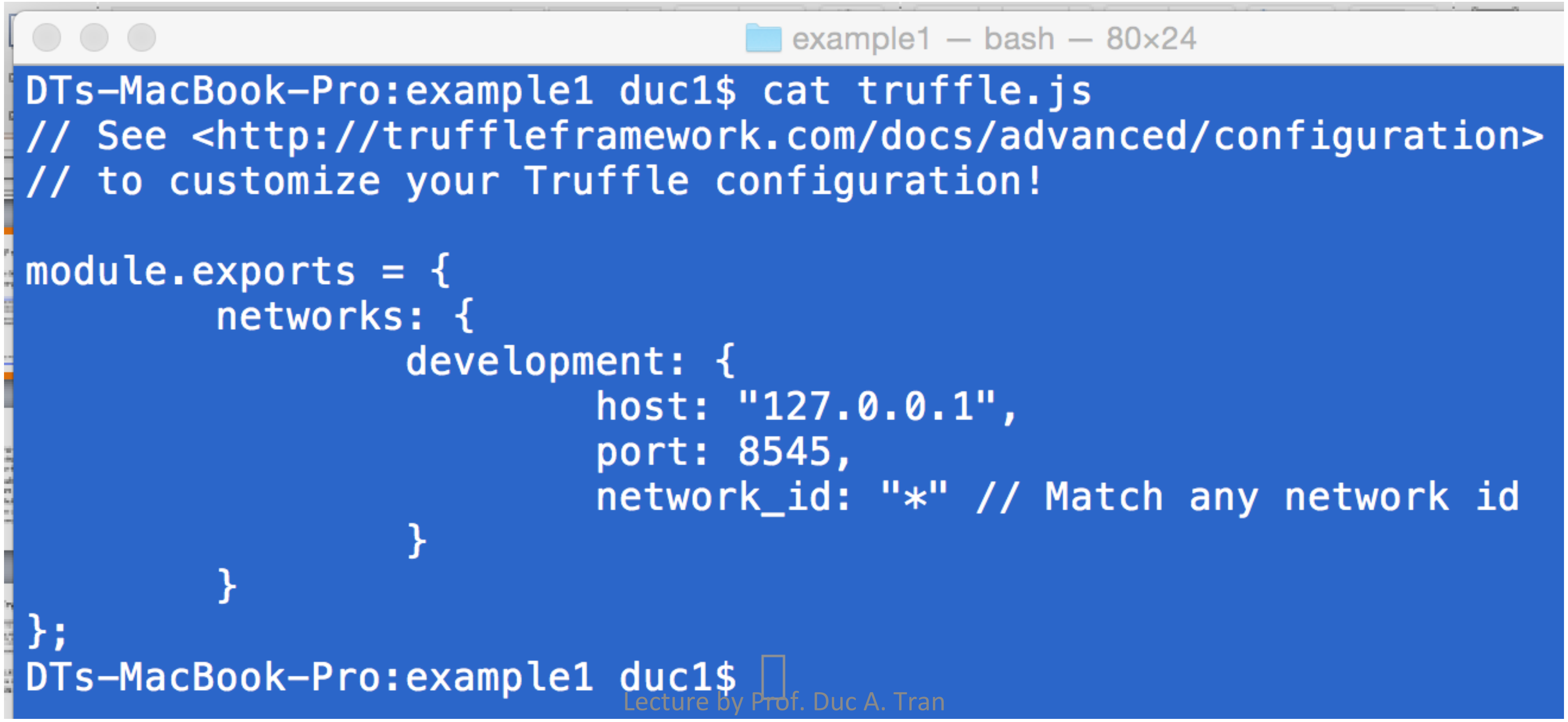
truffle

The default Truffle directory structure contains the following:

- `contracts/`: Contains the Solidity source files for our smart contracts. There is an important contract in here called `Migrations.sol`, which we'll talk about later.
- `migrations/`: Truffle uses a migration system to handle smart contract deployments. A migration is an additional special smart contract that keeps track of changes.
- `test/`: Contains both JavaScript and Solidity tests for our smart contracts
- `truffle.js`: Truffle configuration file

Project Configuration

- Set up a development network
  - by editing file *"truffle.js"* in project folder, e.g.,

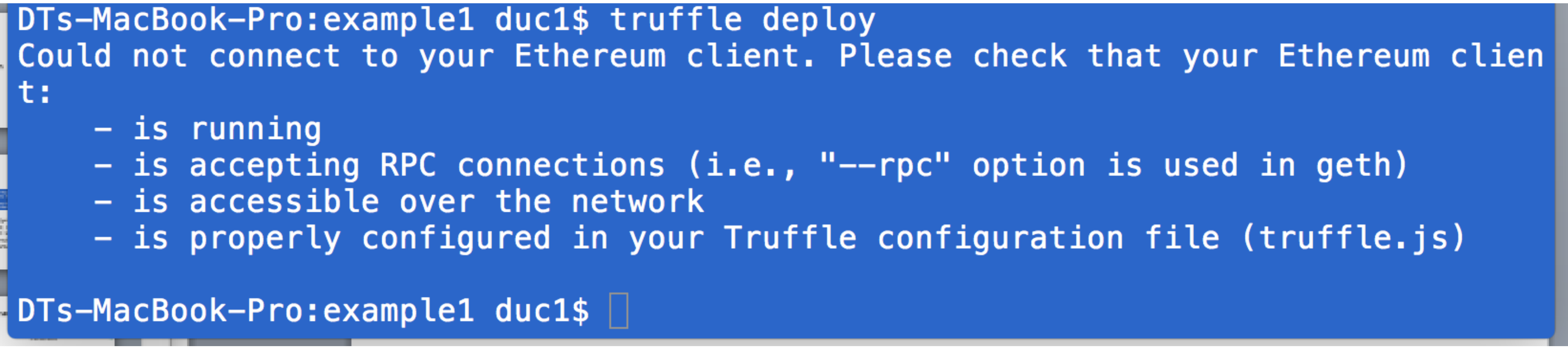


```
DTs-MacBook-Pro:example1 duc1$ cat truffle.js
// See <http://truffleframework.com/docs/advanced/configuration>
// to customize your Truffle configuration!

module.exports = {
        networks: {
                development: {
                        host: "127.0.0.1",
                        port: 8545,
                        network_id: "*" // Match any network id
                }
        }
};
DTs-MacBook-Pro:example1 duc1$
```

Deploy

- Assuming compilation succeeds, now type *"truffle deploy"* at command line to deploy the smart contract
- As a result, truffle tries to connect to a Ethereum node by a RPC manner
  - By default, *localhost:8545*
  - We can change this configuration (see last slide) when we want (for example, actual deployment)

Try 1: Did you see this?

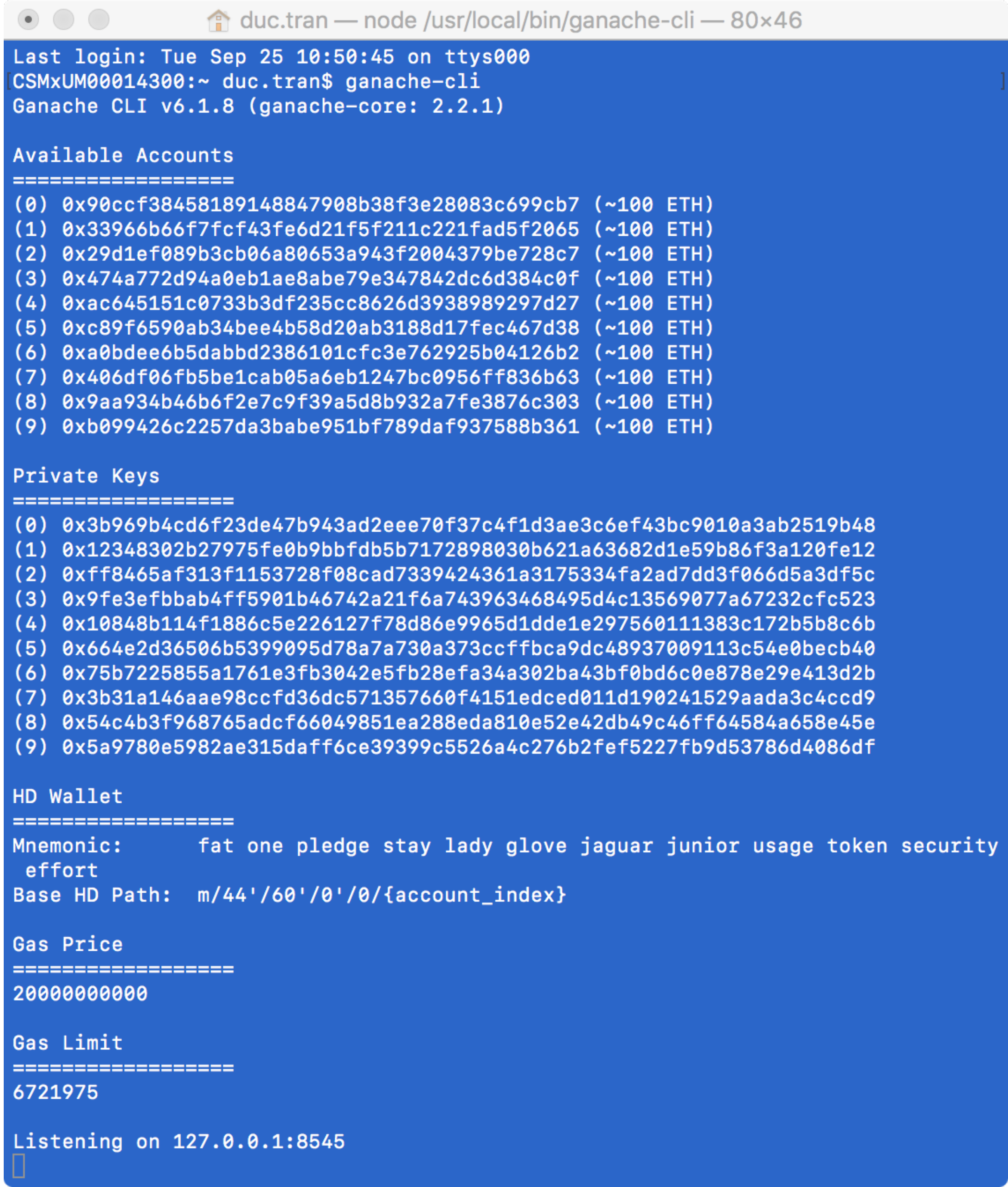


```
DTs-MacBook-Pro:example1 duc1$ truffle deploy
Could not connect to your Ethereum client. Please check that your Ethereum clien
t:
    - is running
    - is accepting RPC connections (i.e., "--rpc" option is used in geth)
    - is accessible over the network
    - is properly configured in your Truffle configuration file (truffle.js)

DTs-MacBook-Pro:example1 duc1$ 
```

- HIGHLY LIKELY because you have not run a Ethereum client node.
- <u>SOLUTION</u>: run *"ganache-cli"* on a separate console window

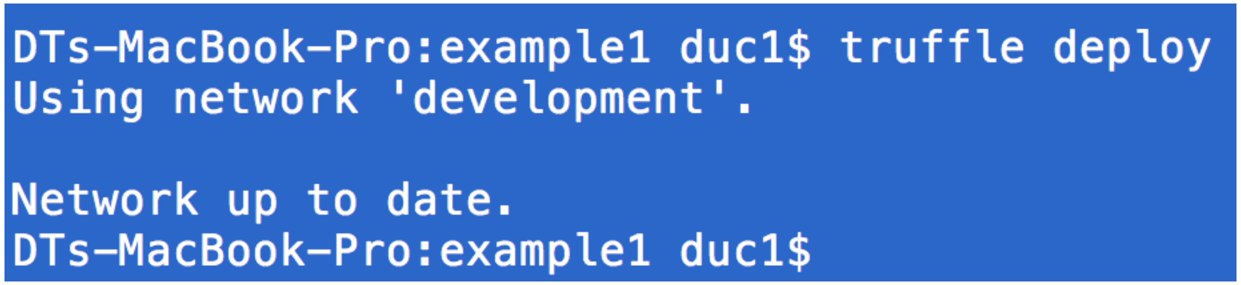
Run A Local Ethereum Node

10 test accounts, generated automatically, each preloaded with 100 (fake) ETH.



```
Last login: Tue Sep 25 10:50:45 on ttys000
CSMxUM00014300:~ duc.tran$ ganache-cli
Ganache CLI v6.1.8 (ganache-core: 2.2.1)

Available Accounts
==================
(0) 0x90ccf38458189148847908b38f3e28083c699cb7 (~100 ETH)
(1) 0x33966b66f7fcf43fe6d21f5f211c221fad5f2065 (~100 ETH)
(2) 0x29d1ef089b3cb06a80653a943f2004379be728c7 (~100 ETH)
(3) 0x474a772d94a0eb1ae8abe79e347842dc6d384c0f (~100 ETH)
(4) 0xac645151c0733b3df235cc8626d3938989297d27 (~100 ETH)
(5) 0xc89f6590ab34bee4b58d20ab3188d17fec467d38 (~100 ETH)
(6) 0xa0bdee6b5dabbd2386101cfc3e762925b04126b2 (~100 ETH)
(7) 0x406df06fb5be1cab05a6eb1247bc0956ff836b63 (~100 ETH)
(8) 0x9aa934b46b6f2e7c9f39a5d8b932a7fe3876c303 (~100 ETH)
(9) 0xb099426c2257da3babe951bf789daf937588b361 (~100 ETH)

Private Keys
==================
(0) 0x3b969b4cd6f23de47b943ad2eee70f37c4f1d3ae3c6ef43bc9010a3ab2519b48
(1) 0x12348302b27975fe0b9bbfdb5b7172898030b621a63682d1e59b86f3a120fe12
(2) 0xff8465af313f1153728f08cad7339424361a3175334fa2ad7dd3f066d5a3df5c
(3) 0x9fe3efbbab4ff5901b46742a21f6a743963468495d4c13569077a67232cfc523
(4) 0x10848b114f1886c5e226127f78d86e9965d1dde1e297560111383c172b5b8c6b
(5) 0x664e2d36506b5399095d78a7a730a373ccffbca9dc48937009113c54e0becb40
(6) 0x75b7225855a1761e3fb3042e5fb28efa34a302ba43bf0bd6c0e878e29e413d2b
(7) 0x3b31a146aae98ccfd36dc571357660f4151edced011d190241529aada3c4ccd9
(8) 0x54c4b3f968765adcf66049851ea288eda810e52e42db49c46ff64584a658e45e
(9) 0x5a9780e5982ae315daff6ce39399c5526a4c276b2fef5227fb9d53786d4086df

HD Wallet
==================
Mnemonic:      fat one pledge stay lady glove jaguar junior usage token security
 effort
Base HD Path:  m/44'/60'/0'/0/{account_index}

Gas Price
==================
20000000000

Gas Limit
==================
6721975

Listening on 127.0.0.1:8545
```

Deploy: Try Again

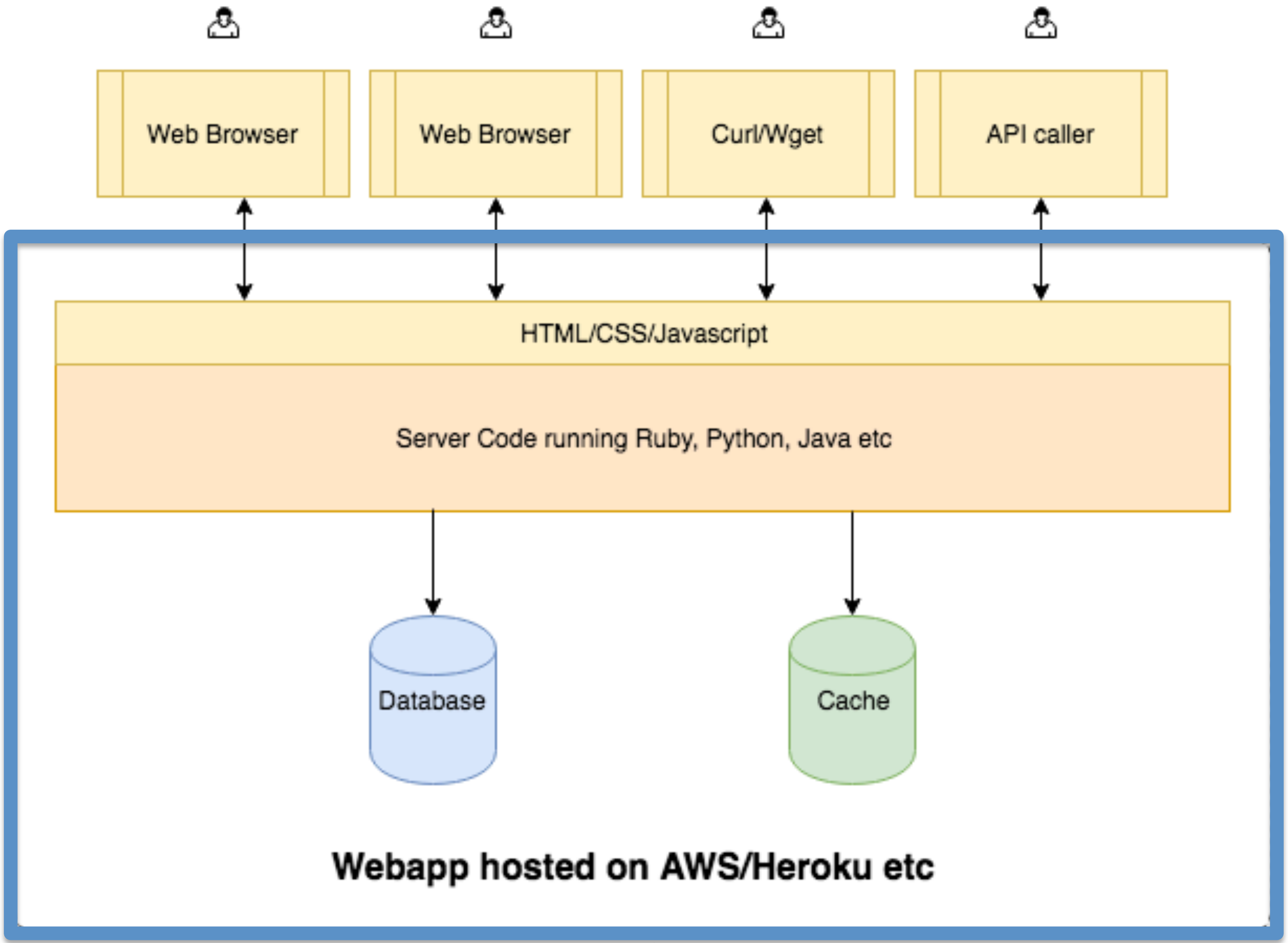


```
DTs-MacBook-Pro:example1 duc1$ truffle deploy
Using network 'development'.

Network up to date.
DTs-MacBook-Pro:example1 duc1$
```

- Success! (you must run it in the project's root folder)
- As a result, **contract's address** and **ABI** (JSON-version of the compiled contract) is added to the config directory
  - This is needed to run (later) *truffle test* and *truffle build*

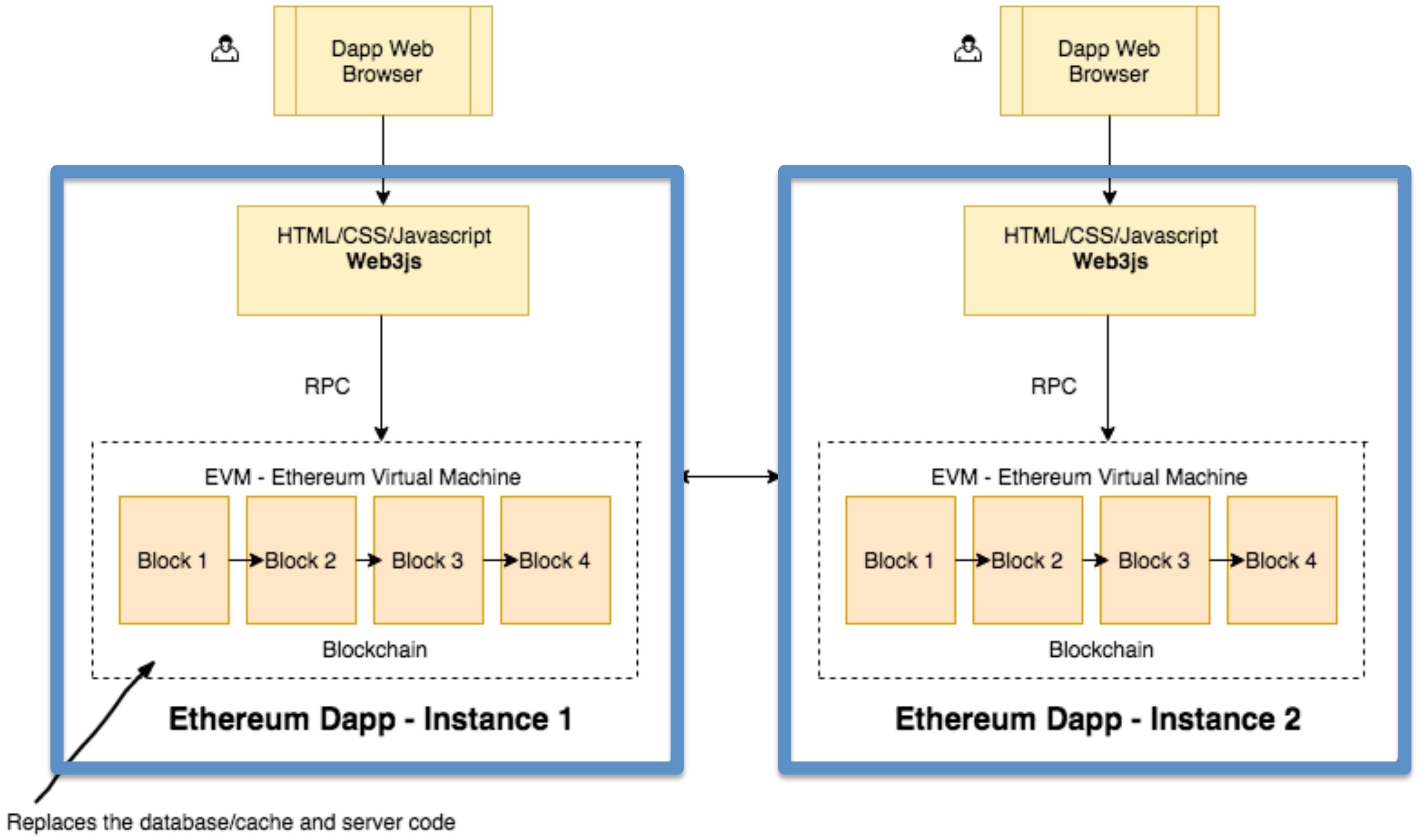
src: medium.com/@mvmurthy

Webapp Architecture


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

src: medium.com/@mvmurthy

DApp Architecture


Dapp Web Browser
HTML/CSS/Javascript **Web3js**
RPC
EVM - Ethereum Virtual Machine
Block 1
Block 2
Block 3
Block 4
Blockchain
Ethereum Dapp - Instance 1
Dapp Web Browser
HTML/CSS/Javascript **Web3js**
RPC
EVM - Ethereum Virtual Machine
Block 1
Block 2
Block 3
Block 4
Blockchain
Ethereum Dapp - Instance 2
Replaces the database/cache and server code

Smart Contract

- The word “contract” might be confusing to tech people
- Simply think of a “contract” as a “program”, or “application code” to run the logic of your application
- Every contract has an Ethereum address
- Knowing this address and the ABI of the contract we can interact with the contract using the methods specified in the contract

- Let us design our contract next!

Example: A Simple Storage

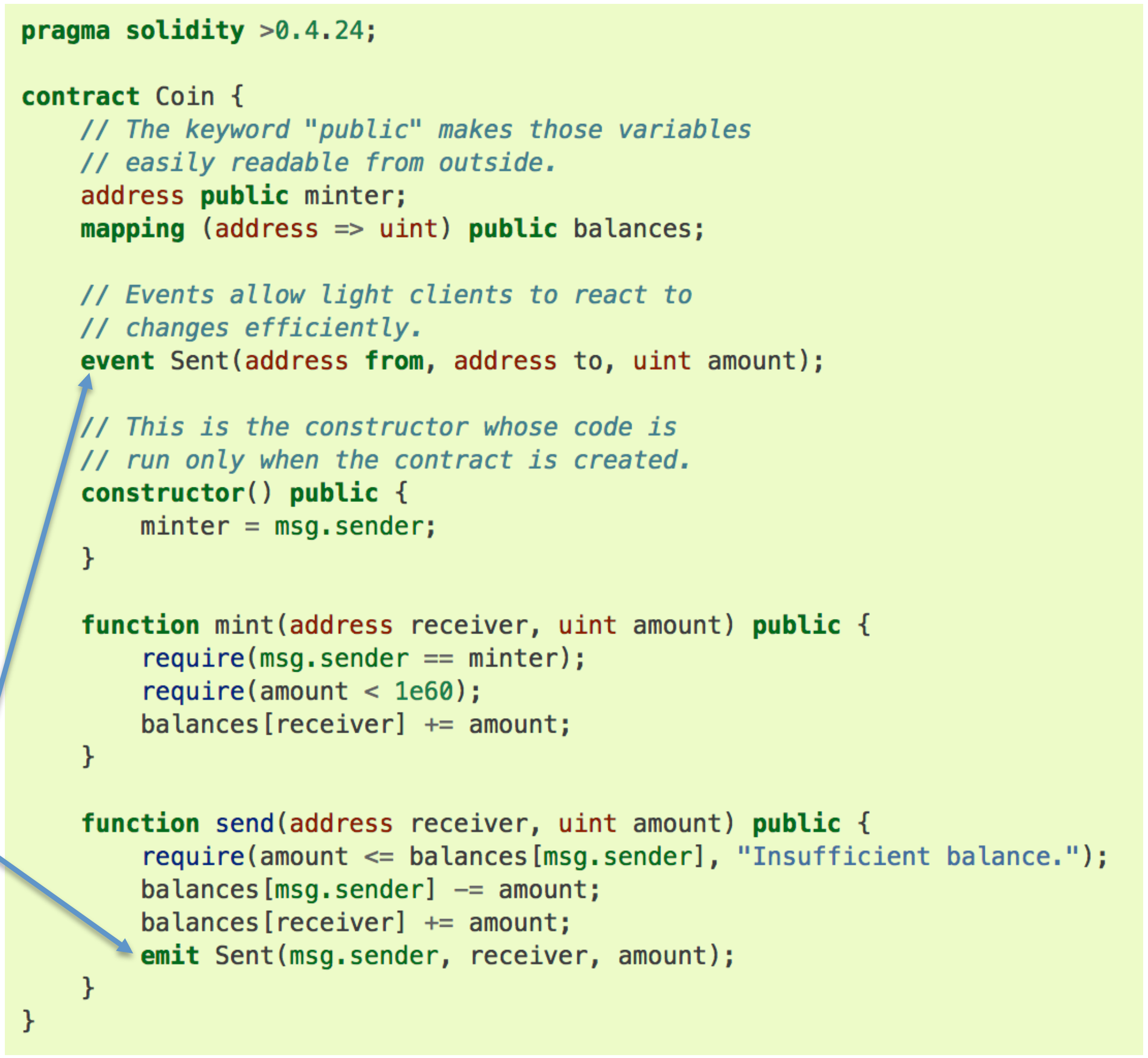
User can

1. Store a number on the blockchain, which will replace the previously stored number
2. Read the number currently stored on the blockchain

```
pragma solidity ^0.4.0;

contract SimpleStorage {
    uint storedData;

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

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

Example: A Simple Coin

- Keyword **public:** automatically generates a function that allows you to access the value. For example:
  - **minter()** gives the value of the public attribute minter
  - **balances.call(addr)** gives balances[addr]
- Keyword **event**: when an event is emitted, will trigger the Javascript code that has been listening on the event



```solidity
pragma solidity >0.4.24;

contract Coin {
    // The keyword "public" makes those variables
    // easily readable from outside.
    address public minter;
    mapping (address => uint) public balances;

    // Events allow light clients to react to
    // changes efficiently.
    event Sent(address from, address to, uint amount);

    // This is the constructor whose code is
    // run only when the contract is created.
    constructor() public {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        require(amount < 1e60);
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender], "Insufficient balance.");
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

Listening for an Event

```
Coin.Sent().watch({}, '', function(error, result) {
    if (!error) {
        console.log("Coin transfer: " + result.args.amount +
            " coins were sent from " + result.args.from +
            " to " + result.args.to + ".");
        console.log("Balances now:\n" +
            "Sender: " + Coin.balances.call(result.args.from) +
            "Receiver: " + Coin.balances.call(result.args.to));
    }
})
```

- This Javascript code is executed when the event Sent() is emitted (previous slide)

Example:

Ballot Voting

```
pragma solidity >=0.4.22 <0.6.0;

/// @title Voting with delegation.
contract Ballot {
    // This declares a new complex type which will
    // be used for variables later.
    // It will represent a single voter.
    struct Voter {
        uint weight; // weight is accumulated by delegation
        bool voted;  // if true, that person already voted
        address delegate; // person delegated to
        uint vote;   // index of the voted proposal
    }

    // This is a type for a single proposal.
    struct Proposal {
        bytes32 name;   // short name (up to 32 bytes)
        uint voteCount; // number of accumulated votes
    }

    address public chairperson;

    // This declares a state variable that
    // stores a `Voter` struct for each possible address.
    mapping(address => Voter) public voters;

    // A dynamically-sized array of `Proposal` structs.
    Proposal[] public proposals;
```

```
/// Create a new ballot to choose one of `proposalNames`.
constructor(bytes32[] memory proposalNames) public {
    chairperson = msg.sender;
    voters[chairperson].weight = 1;

    // For each of the provided proposal names,
    // create a new proposal object and add it
    // to the end of the array.
    for (uint i = 0; i < proposalNames.length; i++) {
        // `Proposal({...})` creates a temporary
        // Proposal object and `proposals.push(...)`
        // appends it to the end of `proposals`.
        proposals.push(Proposal({
            name: proposalNames[i],
            voteCount: 0
        }));
    }
}
```

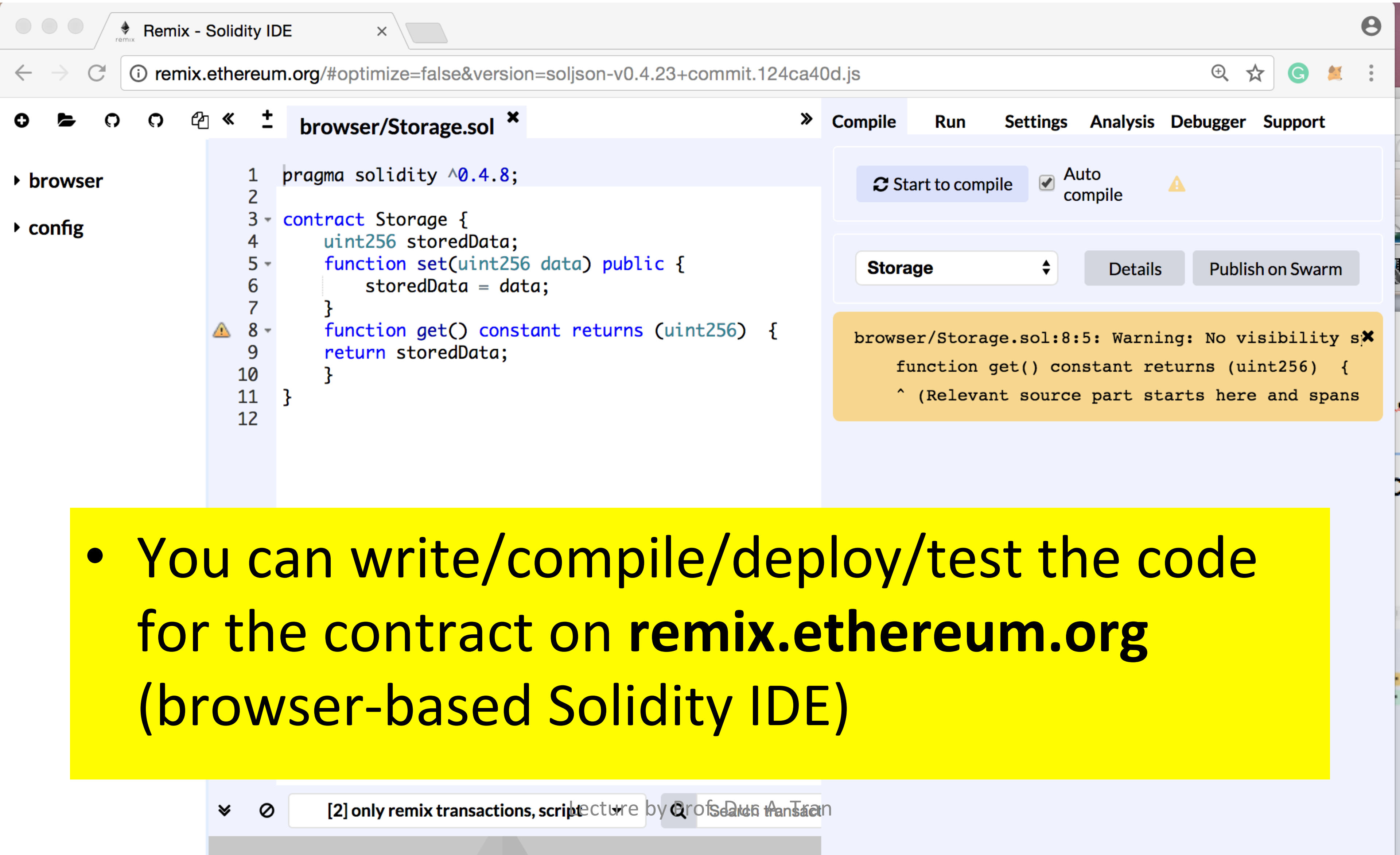
```
// Give `voter` the right to vote on this ballot.
// May only be called by `chairperson`.
function giveRightToVote(address voter) public {
    // If the first argument of `require` evaluates
    // to `false`, execution terminates and all
    // changes to the state and to Ether balances
    // are reverted.
    // This used to consume all gas in old EVM versions, but
    // not anymore.
    // It is often a good idea to use `require` to check if
    // functions are called correctly.
    // As a second argument, you can also provide an
    // explanation about what went wrong.
    require(
        msg.sender == chairperson,
        "Only chairperson can give right to vote."
    );
    require(
        !voters[voter].voted,
        "The voter already voted."
    );
    require(voters[voter].weight == 0);
    voters[voter].weight = 1;
}
```

```solidity
/// Delegate your vote to the voter `to`.
function delegate(address to) public {
    // assigns reference
    Voter storage sender = voters[msg.sender];
    require(!sender.voted, "You already voted.");

    require(to != msg.sender, "Self-delegation is disallowed.");

    // Forward the delegation as long as
    // `to` also delegated.
    // In general, such loops are very dangerous,
    // because if they run too long, they might
    // need more gas than is available in a block.
    // In this case, the delegation will not be executed,
    // but in other situations, such loops might
    // cause a contract to get "stuck" completely.
    while (voters[to].delegate != address(0)) {
        to = voters[to].delegate;

        // We found a loop in the delegation, not allowed.
        require(to != msg.sender, "Found loop in delegation.");
    }

    // Since `sender` is a reference, this
    // modifies `voters[msg.sender].voted`
    sender.voted = true;
    sender.delegate = to;
    Voter storage delegate_ = voters[to];
    if (delegate_.voted) {
        // If the delegate already voted,
        // directly add to the number of votes
        proposals[delegate_.vote].voteCount += sender.weight;
    } else {
        // If the delegate did not vote yet,
        // add to her weight.
        delegate_.weight += sender.weight;
    }
}
```

```
/// Give your vote (including votes delegated to you)
/// to proposal `proposals[proposal].name`.
function vote(uint proposal) public {
    Voter storage sender = voters[msg.sender];
    require(!sender.voted, "Already voted.");
    sender.voted = true;
    sender.vote = proposal;

    // If `proposal` is out of the range of the array,
    // this will throw automatically and revert all
    // changes.
    proposals[proposal].voteCount += sender.weight;
}
```

```
    /// @dev Computes the winning proposal taking all
    /// previous votes into account.
    function winningProposal() public view
            returns (uint winningProposal_)
    {
        uint winningVoteCount = 0;
        for (uint p = 0; p < proposals.length; p++) {
            if (proposals[p].voteCount > winningVoteCount) {
                winningVoteCount = proposals[p].voteCount;
                winningProposal_ = p;
            }
        }
    }

    // Calls winningProposal() function to get the index
    // of the winner contained in the proposals array and then
    // returns the name of the winner
    function winnerName() public view
            returns (bytes32 winnerName_)
    {
        winnerName_ = proposals[winningProposal()].name;
    }
}
```

Test Contract with remix.ethereum.org



```
pragma solidity ^0.4.8;

contract Storage {
    uint256 storedData;
    function set(uint256 data) public {
        storedData = data;
    }
    function get() constant returns (uint256)  {
    return storedData;
    }
}
```

```
browser/Storage.sol:8:5: Warning: No visibility s
    function get() constant returns (uint256)  {
    ^ (Relevant source part starts here and spans
```

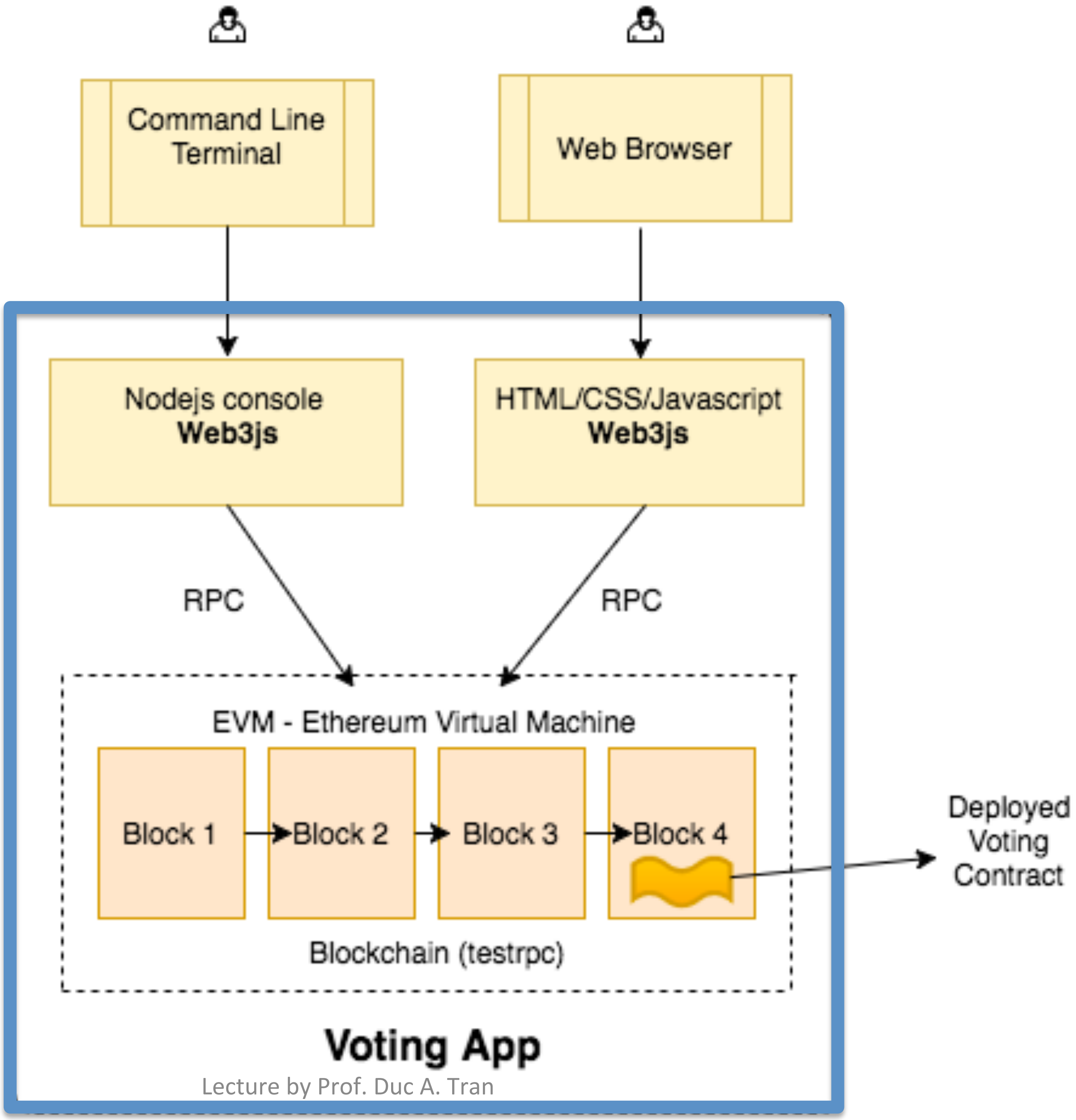
- You can write/compile/deploy/test the code for the contract on **remix.ethereum.org** (browser-based Solidity IDE)

Gas Limit, Gas Price

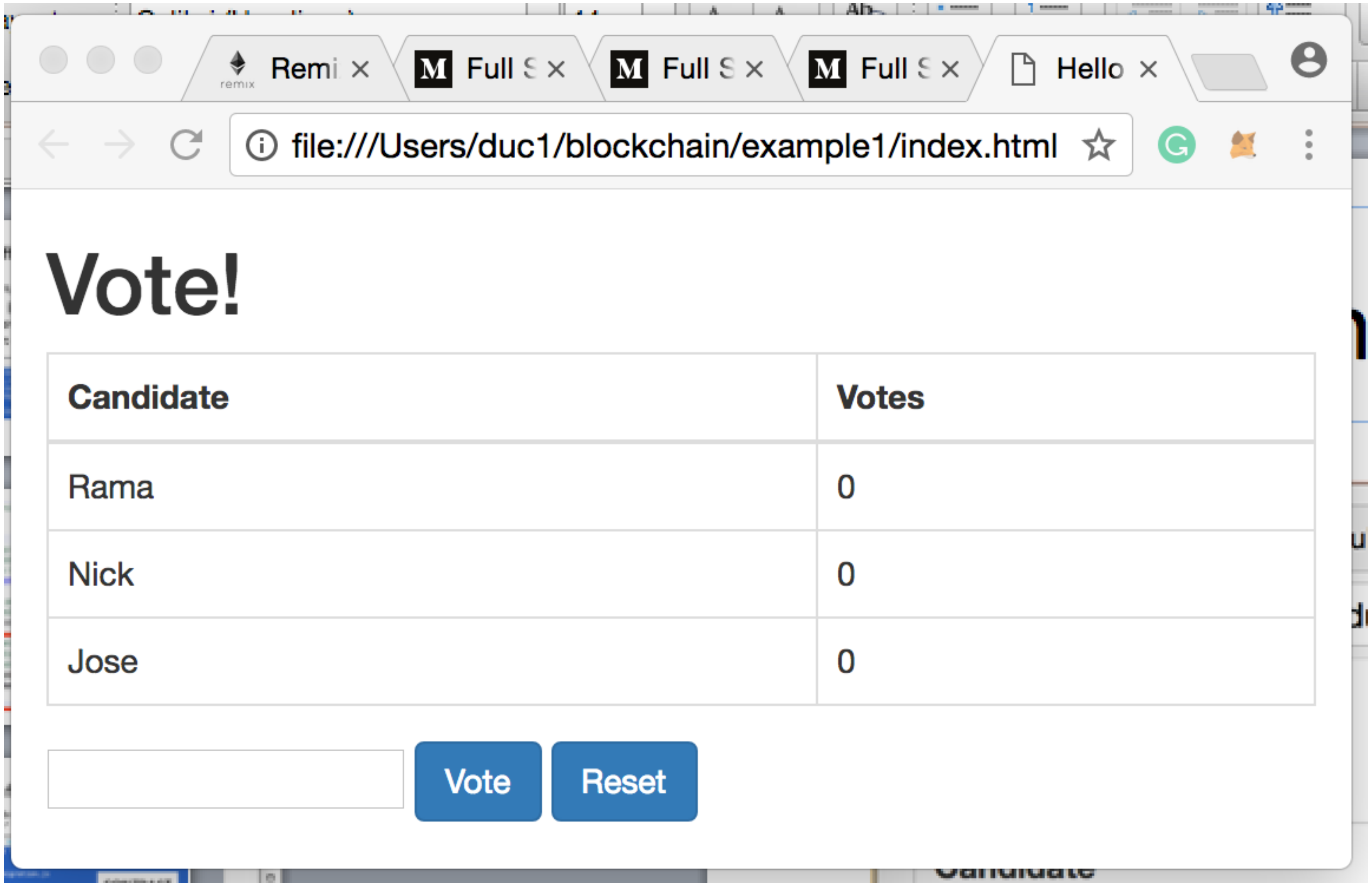
- Gas Limit (e.g., 3,000,000)
  - EVM charges some gas to run each instruction.
  - If a transaction's execution exceeds the Gas Limit, it is considered "out of gas"; it is canceled, reverting to original state.
- Gas Price (e.g., 2Gwei per gas)
  - If transaction requires 100 gas. Miner will earn 2Gwei * 100 = 200Gwei
  - The higher set, the more likely miner will include the TX in the new block

Example:

Another Voting Contract

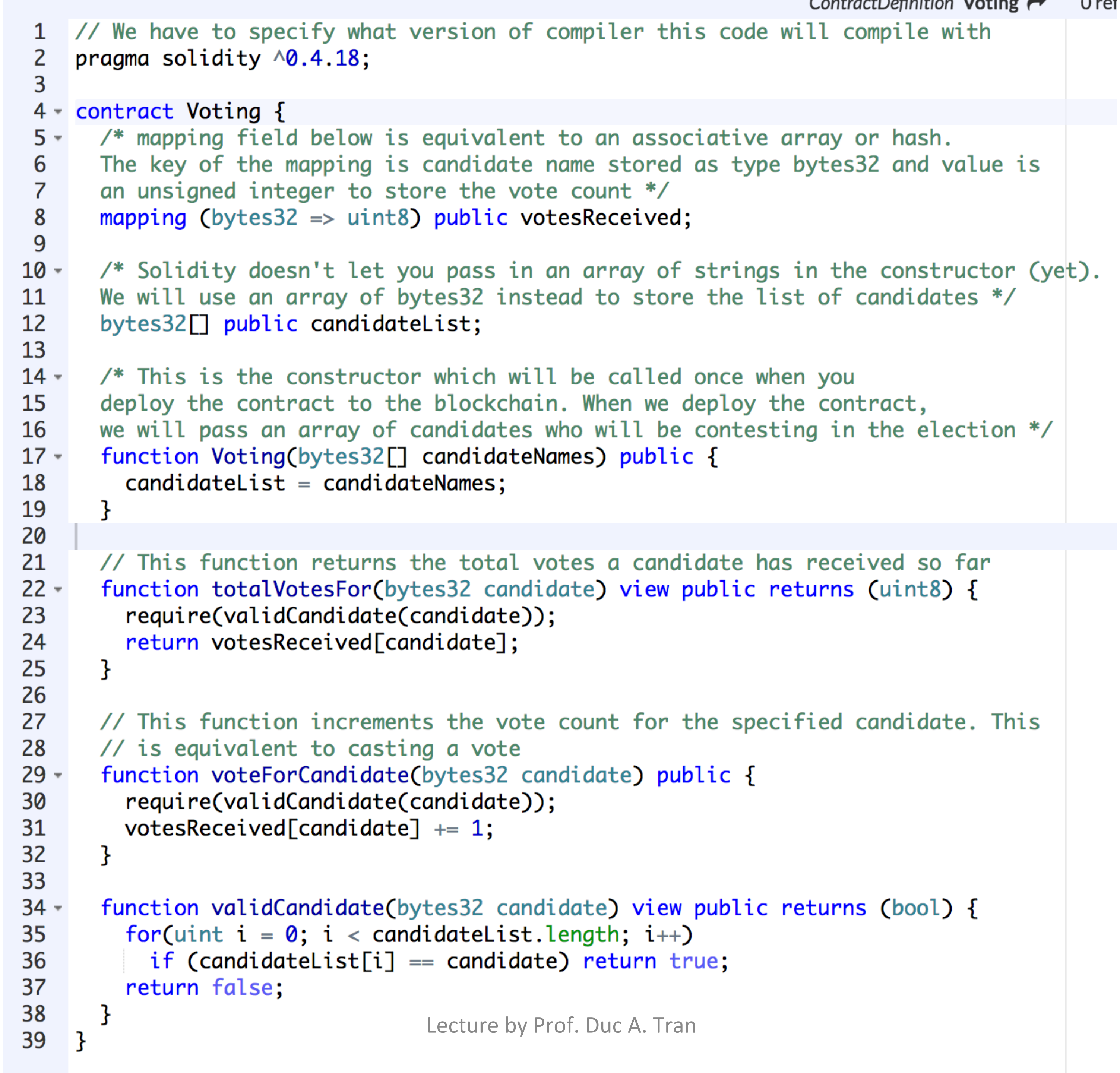


Frond-End Webpage



Vote!

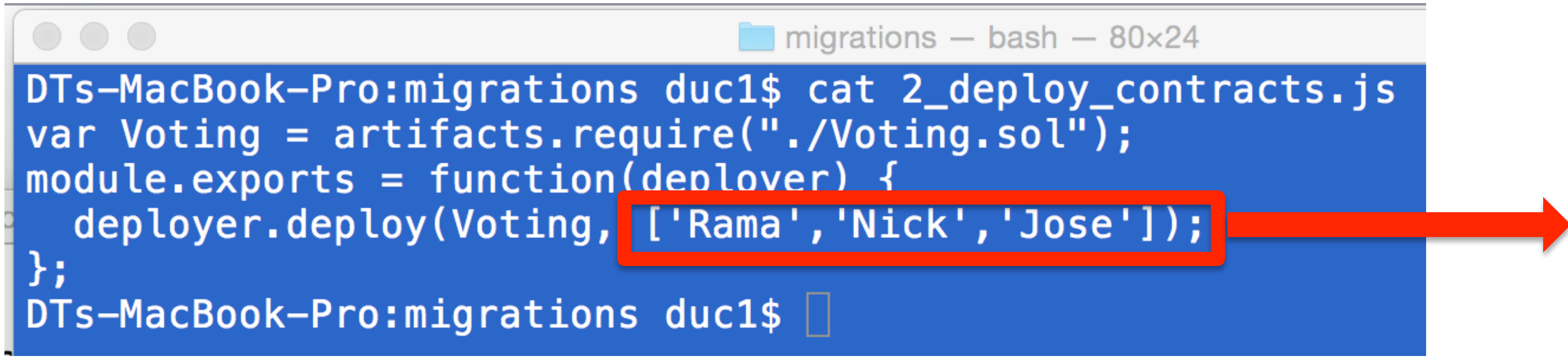
Candidate	Votes
Rama	0
Nick	0
Jose	0



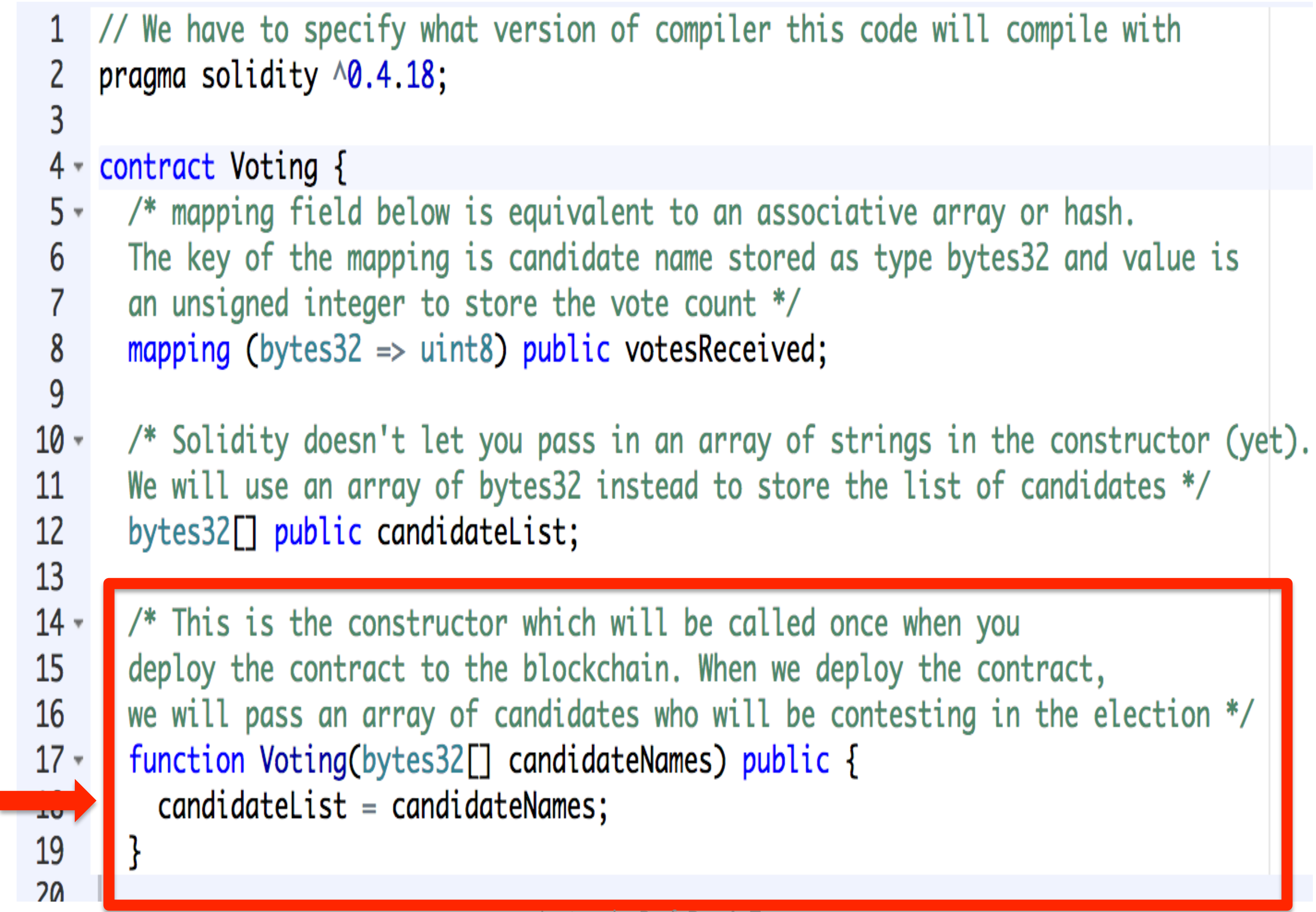
```
// We have to specify what version of compiler this code will compile with
pragma solidity ^0.4.18;

contract Voting {
  /* mapping field below is equivalent to an associative array or hash.
  The key of the mapping is candidate name stored as type bytes32 and value is
  an unsigned integer to store the vote count */
  mapping (bytes32 => uint8) public votesReceived;

  /* Solidity doesn't let you pass in an array of strings in the constructor (yet).
  We will use an array of bytes32 instead to store the list of candidates */
  bytes32[] public candidateList;

  /* This is the constructor which will be called once when you
  deploy the contract to the blockchain. When we deploy the contract,
  we will pass an array of candidates who will be contesting in the election */
  function Voting(bytes32[] candidateNames) public {
    candidateList = candidateNames;
  }

  // This function returns the total votes a candidate has received so far
  function totalVotesFor(bytes32 candidate) view public returns (uint8) {
    require(validCandidate(candidate));
    return votesReceived[candidate];
  }

  // This function increments the vote count for the specified candidate. This
  // is equivalent to casting a vote
  function voteForCandidate(bytes32 candidate) public {
    require(validCandidate(candidate));
    votesReceived[candidate] += 1;
  }

  function validCandidate(bytes32 candidate) view public returns (bool) {
    for(uint i = 0; i < candidateList.length; i++)
      if (candidateList[i] == candidate) return true;
    return false;
  }
}
```

truffle migrate

- Go to “*migrations/*” folder
- Create a new file (if not existing) named *“2_deploy_contracts.js”*, and add the following content (to initiate the contract):



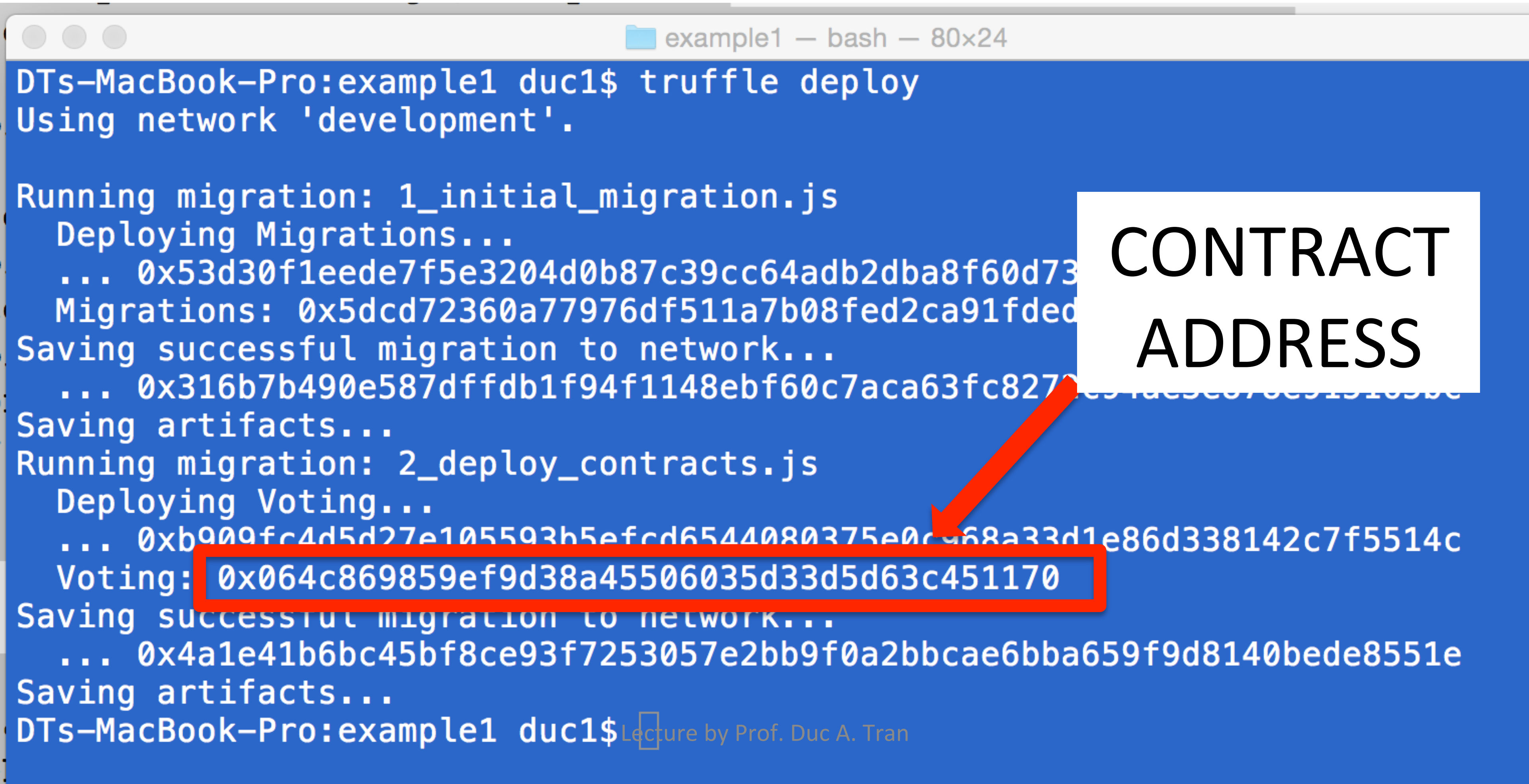
```
DTs-MacBook-Pro:migrations duc1$ cat 2_deploy_contracts.js
var Voting = artifacts.require("./Voting.sol");
module.exports = function(deployer) {
  deployer.deploy(Voting, ['Rama','Nick','Jose']);
};
DTs-MacBook-Pro:migrations duc1$ 
```



```
// We have to specify what version of compiler this code will compile with
pragma solidity ^0.4.18;

contract Voting {
  /* mapping field below is equivalent to an associative array or hash.
  The key of the mapping is candidate name stored as type bytes32 and value is
  an unsigned integer to store the vote count */
  mapping (bytes32 => uint8) public votesReceived;

  /* Solidity doesn't let you pass in an array of strings in the constructor (yet).
  We will use an array of bytes32 instead to store the list of candidates */
  bytes32[] public candidateList;

  /* This is the constructor which will be called once when you
  deploy the contract to the blockchain. When we deploy the contract,
  we will pass an array of candidates who will be contesting in the election */
  function Voting(bytes32[] candidateNames) public {
    candidateList = candidateNames;
  }
```

truffle deploy

- Make sure you run a node first, *“ganache-cli”*
- Then, run *“truffle deploy”*



```
DTs-MacBook-Pro:example1 duc1$ truffle deploy
Using network 'development'.

Running migration: 1_initial_migration.js
  Deploying Migrations...
  ... 0x53d30f1eede7f5e3204d0b87c39cc64adb2dba8f60d73
  Migrations: 0x5dcd72360a77976df511a7b08fed2ca91fded
Saving successful migration to network...
  ... 0x316b7b490e587dffdb1f94f1148ebf60c7aca63fc827
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Deploying Voting...
  ... 0xb909fc4d5d27e105593b5efcd6544080375e0c968a33d1e86d338142c7f5514c
  Voting: 0x064c869859ef9d38a45506035d33d5d63c451170
Saving successful migration to network...
  ... 0x4a1e41b6bc45bf8ce93f7253057e2bb9f0a2bbcae6bba659f9d8140bede8551e
Saving artifacts...
DTs-MacBook-Pro:example1 duc1$
```

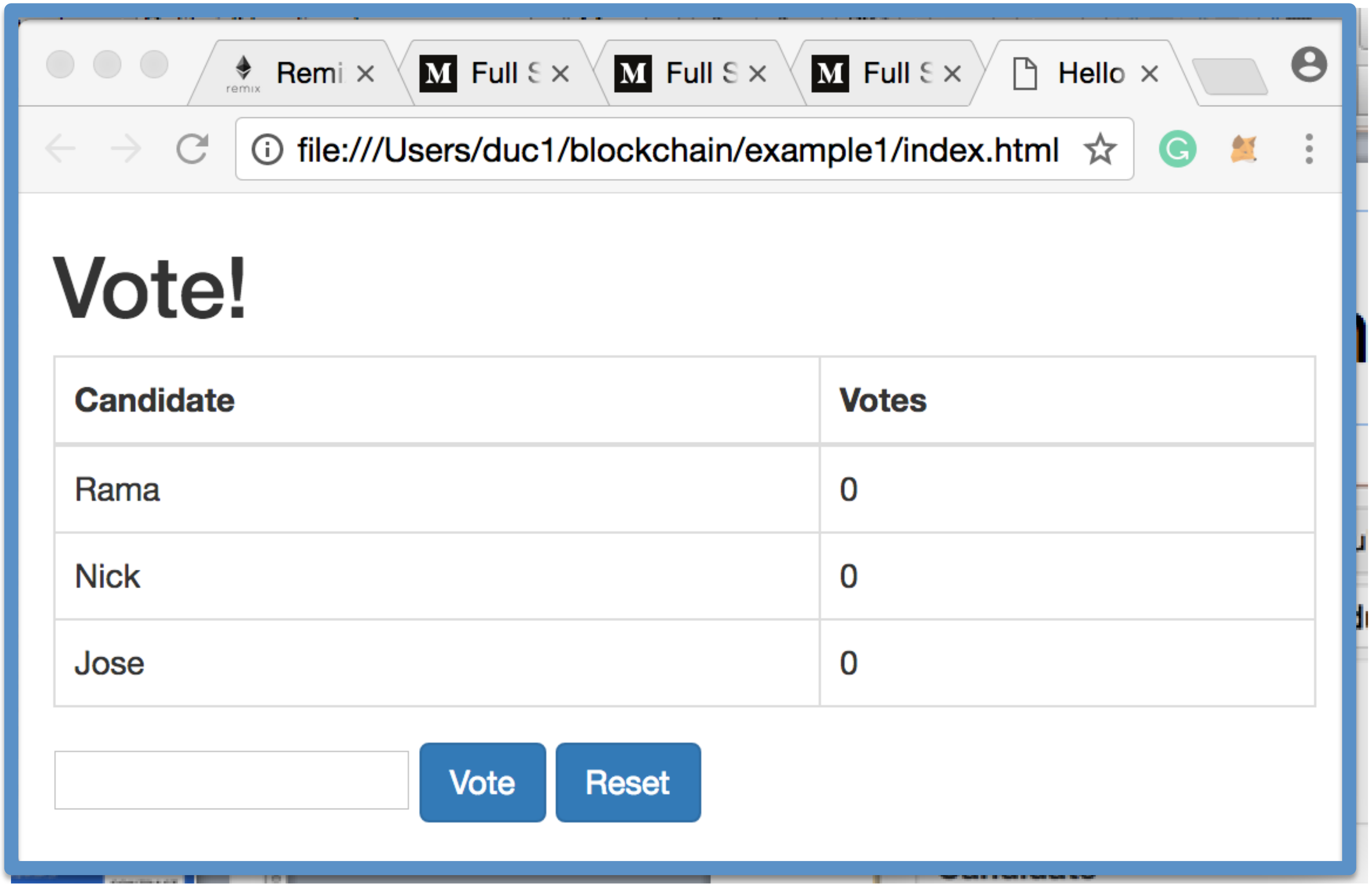
Ready to test?

- We will write a webpage javascript to allow user to interact with the contract (i.e., to vote)

index.html (user-interface webpage)

index.js (called when user inputs)

User-Interface Webpage



Vote!

Candidate	Votes
Rama	0
Nick	0
Jose	0

index.html (you can copy & paste)

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello World DApp</title>
  <link href='https://fonts.googleapis.com/css?family=Open+Sans:400,700' rel='stylesheet' type='text/css'>
  <link href='https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css' rel='stylesheet' type='text/css'>
</head>
<body class="container">
  <h1>A Simple Hello World Voting Application</h1>
  <div class="table-responsive">
    <table class="table table-bordered">
      <thead>
        <tr>
          <th>Candidate</th>
          <th>Votes</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>Rama</td>
          <td id="candidate-1"></td>
        </tr>
        <tr>
          <td>Nick</td>
          <td id="candidate-2"></td>
        </tr>
        <tr>
          <td>Jose</td>
          <td id="candidate-3"></td>
        </tr>
      </tbody>
    </table>
  </div>
  <input type="text" id="candidate" />
  <a href="#" onclick="voteForCandidate()" class="btn btn-primary">Vote</a>
</body>
<script src="https://cdn.rawgit.com/ethereum/web3.js/develop/dist/web3.js"></script>
<script src="https://code.jquery.com/jquery-3.1.1.slim.min.js"></script>
<script src="./index.js"></script>
</html>
```

index.js (you can copy & paste)

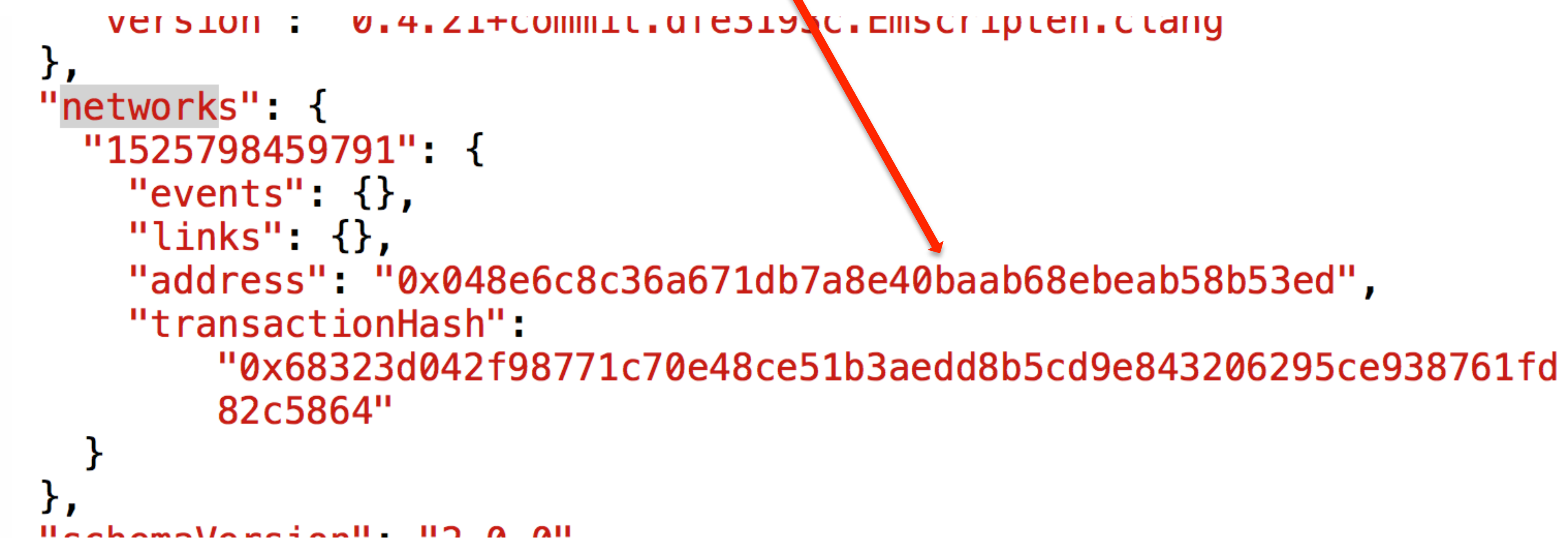
```
web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));
abi = JSON.parse('[{"constant":false,"inputs":[{"name":"candidate","type":"bytes32"}],"name":"totalVotesFor","outputs":
[{"name":"","type":"uint8"}],"payable":false,"type":"function"},{"constant":false,"inputs":
[{"name":"candidate","type":"bytes32"}],"name":"validCandidate","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},{"constant":true,"inputs":
[{"name":"","type":"bytes32"}],"name":"votesReceived","outputs":[{"name":"","type":"uint8"}],"payable":false,"type":"function"},{"constant":true,"inputs":
[{"name":"x","type":"bytes32"}],"name":"bytes32ToString","outputs":[{"name":"","type":"string"}],"payable":false,"type":"function"},{"constant":true,"inputs":
[{"name":"","type":"uint256"}],"name":"candidateList","outputs":[{"name":"","type":"bytes32"}],"payable":false,"type":"function"},{"constant":false,"inputs":
[{"name":"candidate","type":"bytes32"}],"name":"voteForCandidate","outputs":[],"payable":false,"type":"function"},{"constant":true,"inputs":
[],"name":"contractOwner","outputs":[{"name":"","type":"address"}],"payable":false,"type":"function"},{"inputs":
[{"name":"candidateNames","type":"bytes32[]"}],"payable":false,"type":"constructor"}]')
VotingContract = web3.eth.contract(abi);
// In your nodejs console, execute contractInstance.address to get the address at which the contract is deployed and change the line below to use your deployed
address
contractInstance = VotingContract.at('0x064c869859ef9d38a45506035d33d5d63c451[illegible]70');
candidates = {"Rama": "candidate-1", "Nick": "candidate-2", "Jose": "candidate-3"}

function voteForCandidate() {
  candidateName = $("#candidate").val();
  contractInstance.voteForCandidate(candidateName, {from: web3.eth.accounts[0]}, function() {
    let div_id = candidates[candidateName];
    $("#" + div_id).html(contractInstance.totalVotesFor.call(candidateName).toString());
  });
}

$(document).ready(function() {
  candidateNames = Object.keys(candidates);
  for (var i = 0; i < candidateNames.length; i++) {
    let name = candidateNames[i];
    let val = contractInstance.totalVotesFor.call(name).toString()
    $("#" + candidates[name]).html(val);
  }
});
```

You can copy the json string from *Voting.json* in *"./build/contracts/"*

Make sure the contract address is correct

Need Contract Address and ABI?

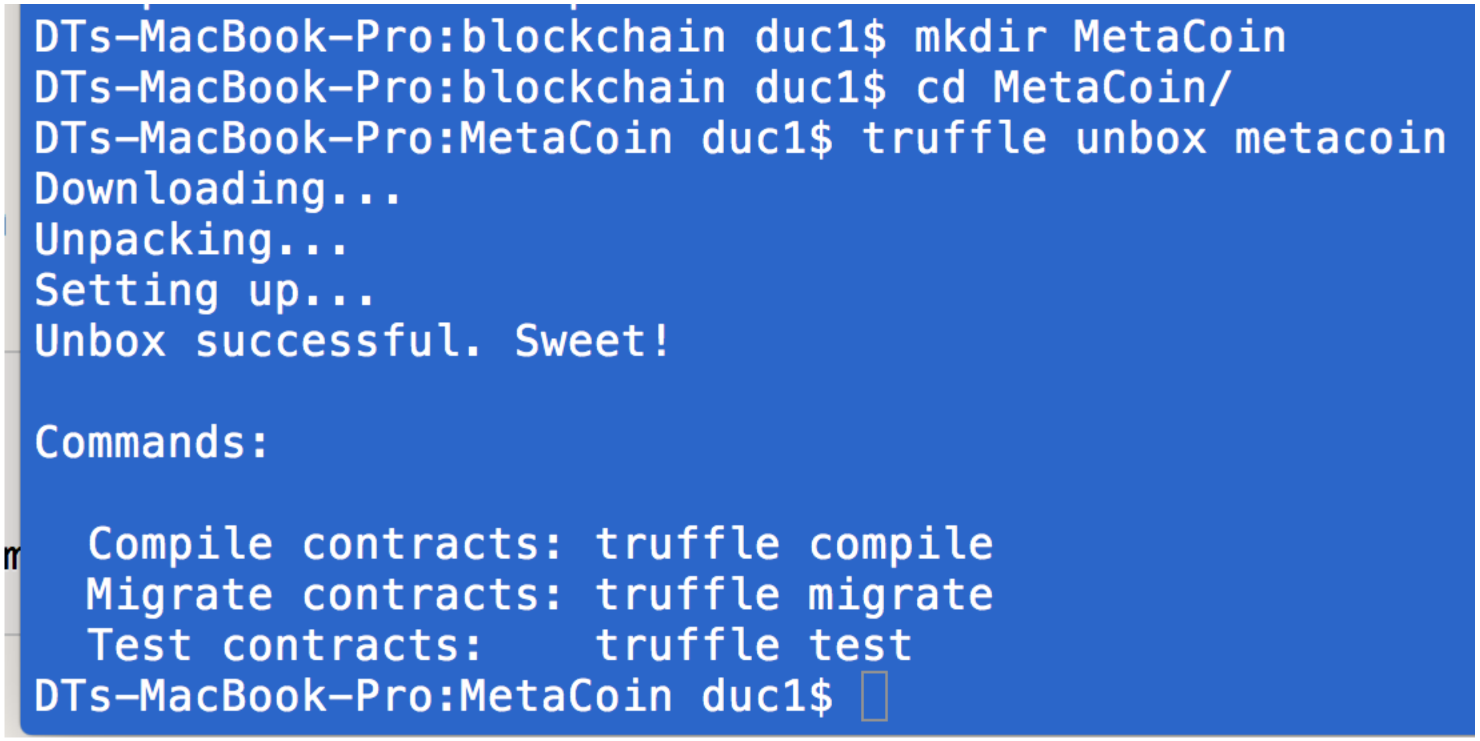
- Look inside the json file in folder *"./build/ contracts/"*



```
    "version": "0.4.21+commit.dfe3193c.Emscripten.clang"
},
"networks": {
  "1525798459791": {
    "events": {},
    "links": {},
    "address": "0x048e6c8c36a671db7a8e40baab68ebeab58b53ed",
    "transactionHash":
        "0x68323d042f98771c70e48ce51b3aedd8b5cd9e843206295ce938761fd
        82c5864"
  }
},
"schemaVersion": "2.0.0"
```

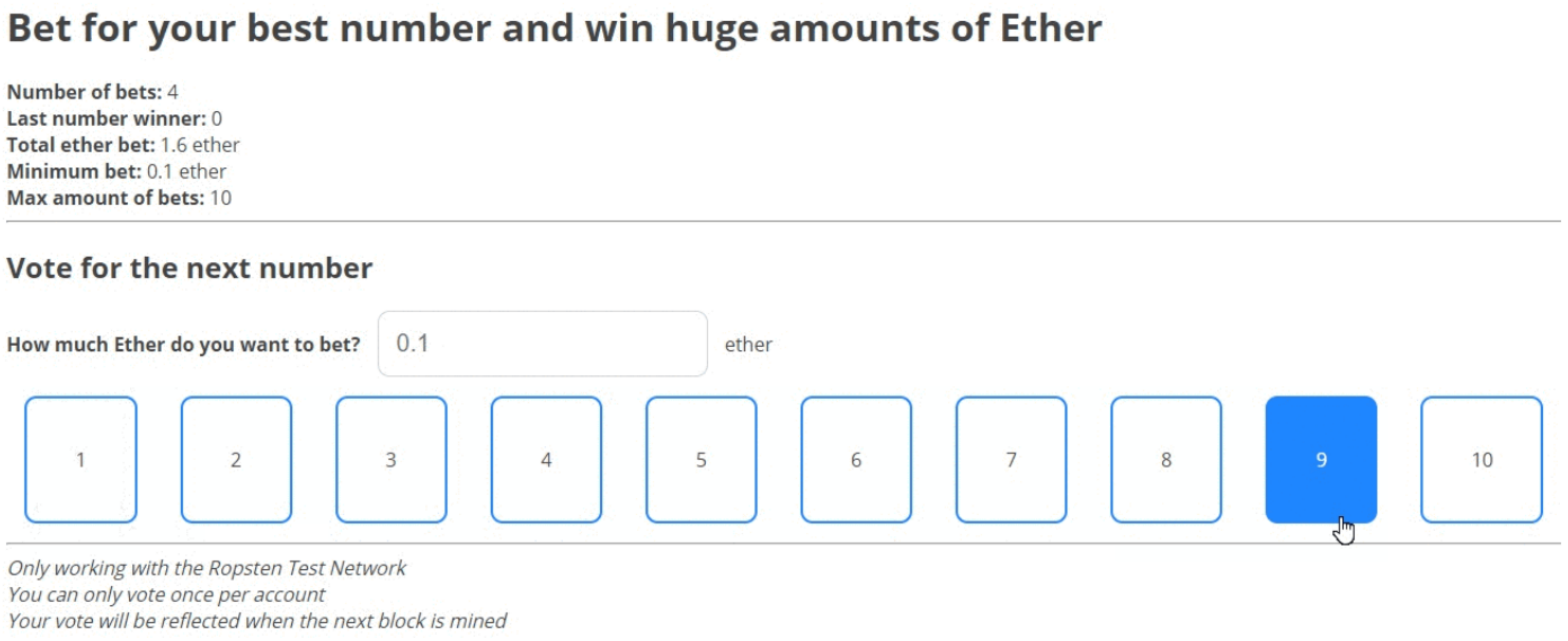
Another Example?

- **MetaCoin**: an example provided by Truffle. To get it to your computer, do the following



```
DTs-MacBook-Pro:blockchain duc1$ mkdir MetaCoin
DTs-MacBook-Pro:blockchain duc1$ cd MetaCoin/
DTs-MacBook-Pro:MetaCoin duc1$ truffle unbox metacoin
Downloading...
Unpacking...
Setting up...
Unbox successful. Sweet!

Commands:

  Compile contracts: truffle compile
  Migrate contracts: truffle migrate
  Test contracts:    truffle test
DTs-MacBook-Pro:MetaCoin duc1$ 
```

Another: Number Betting


Bet for your best number and win huge amounts of Ether

Number of bets: 4
Last number winner: 0
Total ether bet: 1.6 ether
Minimum bet: 0.1 ether
Max amount of bets: 10

Vote for the next number

How much Ether do you want to bet? 0.1 ether

Only working with the Ropsten Test Network
You can only vote once per account
Your vote will be reflected when the next block is mined

https://medium.com/@merunasgrincalaitis/the-ultimate-end-to-end-tutorial-to-create-and-deploy-a-fully-descentralized-dapp-in-ethereum-18f0cf6d7e0e

Do the following in order

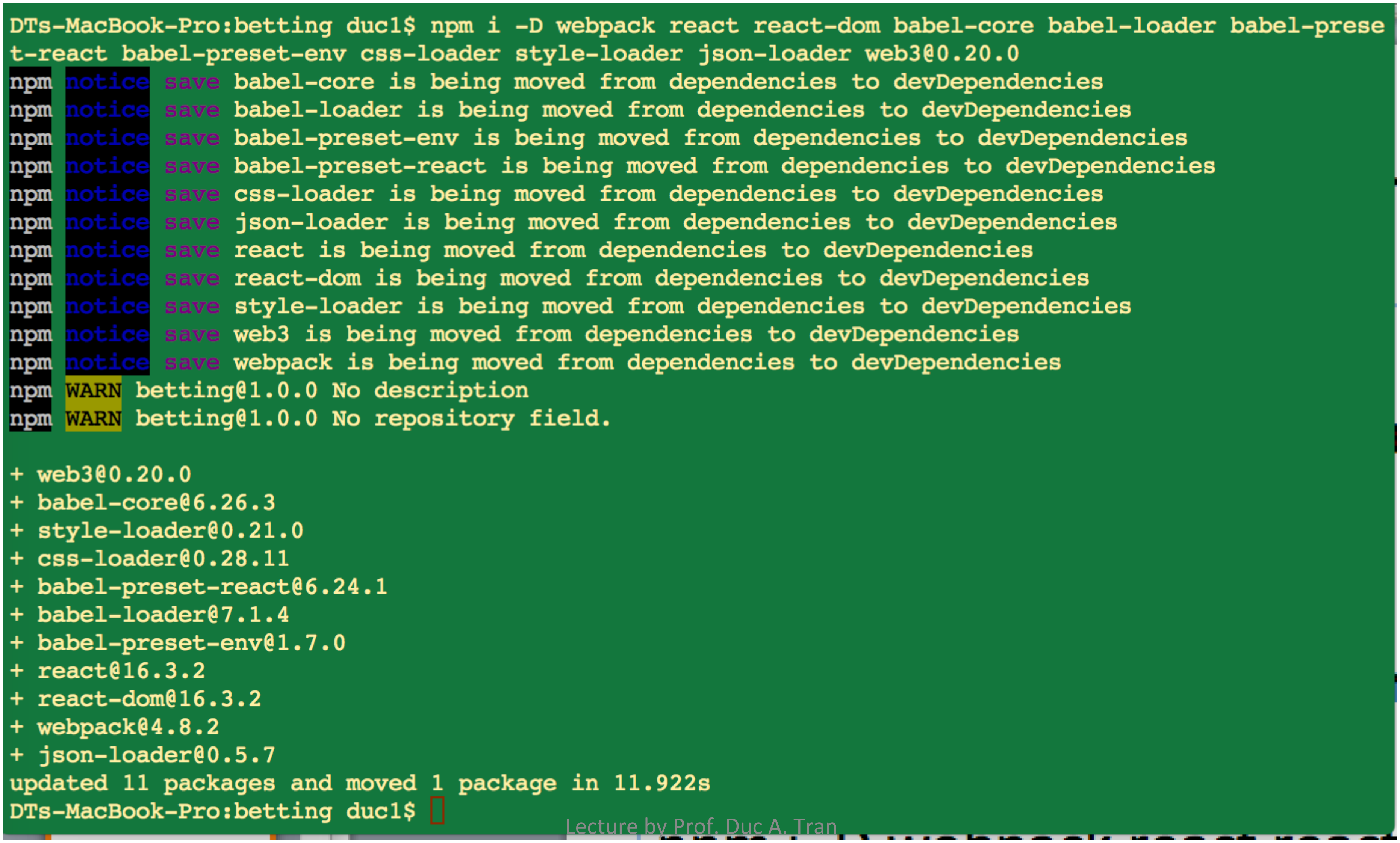
1. Create a new folder, say *“betting”*
2. Go to this folder
3. Type *“truffle init”* (to initialize truffle)
4. Type *“npm init –y”* (to create package.json)

Front-End Preparation

- Need <u>webpack, react, babel and web3.</u> Type the following:

```
npm i -D webpack react react-dom babel-core babel-loader babel-preset-react babel-preset-env css-loader style-loader json-loader web3@0.20.0
```

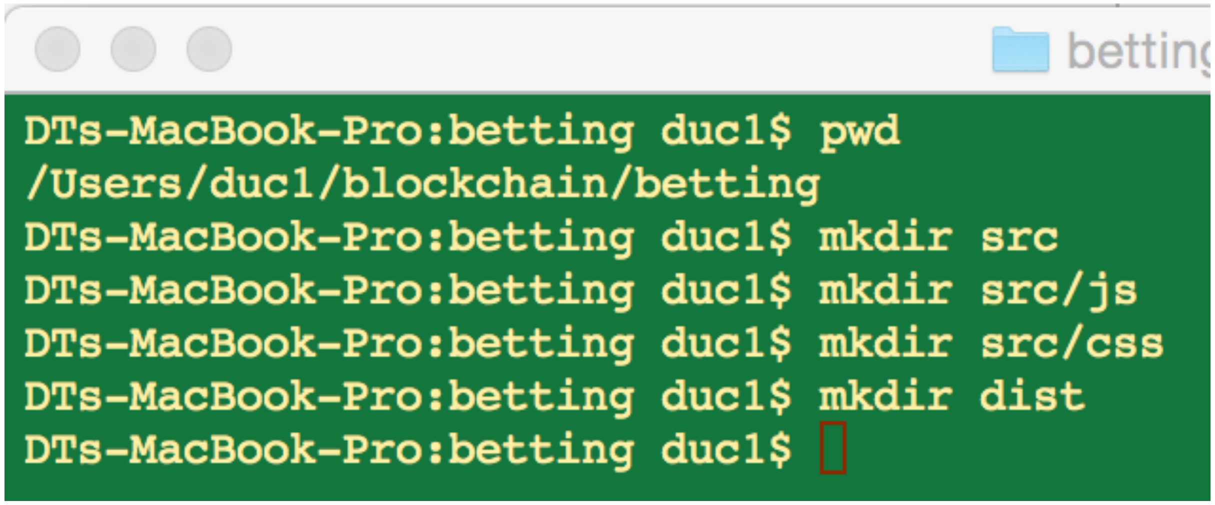
Need webpack, react, babel and web3



```
DTs-MacBook-Pro:betting duc1$ npm i -D webpack react react-dom babel-core babel-loader babel-prese
t-react babel-preset-env css-loader style-loader json-loader web3@0.20.0
npm notice save babel-core is being moved from dependencies to devDependencies
npm notice save babel-loader is being moved from dependencies to devDependencies
npm notice save babel-preset-env is being moved from dependencies to devDependencies
npm notice save babel-preset-react is being moved from dependencies to devDependencies
npm notice save css-loader is being moved from dependencies to devDependencies
npm notice save json-loader is being moved from dependencies to devDependencies
npm notice save react is being moved from dependencies to devDependencies
npm notice save react-dom is being moved from dependencies to devDependencies
npm notice save style-loader is being moved from dependencies to devDependencies
npm notice save web3 is being moved from dependencies to devDependencies
npm notice save webpack is being moved from dependencies to devDependencies
npm WARN betting@1.0.0 No description
npm WARN betting@1.0.0 No repository field.

+ web3@0.20.0
+ babel-core@6.26.3
+ style-loader@0.21.0
+ css-loader@0.28.11
+ babel-preset-react@6.24.1
+ babel-loader@7.1.4
+ babel-preset-env@1.7.0
+ react@16.3.2
+ react-dom@16.3.2
+ webpack@4.8.2
+ json-loader@0.5.7
updated 11 packages and moved 1 package in 11.922s
DTs-MacBook-Pro:betting duc1$
```

Front-End File Organization

- Create folders for source files (index.js, index.css) and output-distribution files (index.html)



```
DTs-MacBook-Pro:betting duc1$ pwd
/Users/duc1/blockchain/betting
DTs-MacBook-Pro:betting duc1$ mkdir src
DTs-MacBook-Pro:betting duc1$ mkdir src/js
DTs-MacBook-Pro:betting duc1$ mkdir src/css
DTs-MacBook-Pro:betting duc1$ mkdir dist
DTs-MacBook-Pro:betting duc1$ 
```

```
contracts/
-- Migrations.sol
migrations/
node_modules/
test/
src/
-- css/index.css
-- js/index.js
dist/
-- index.html
package.json
truffle-config.js
truffle.js
webpack.config.js
```

webpack.config.js

```
const path = require('path')
module.exports = {
  entry: path.join(__dirname, 'src/js', 'index.js'), // Our frontend will be inside the src folder
  output: {
    path: path.join(__dirname, 'dist'),
    filename: 'build.js' // The final file will be created in dist/build.js
  },
  module: {
    loaders: [{
      test: /\.css$/, // To load the css in react
      use: ['style-loader', 'css-loader'],
      include: /src/
    }, {
      test: /\.jsx?$/, // To load the js and jsx files
      loader: 'babel-loader',
      exclude: /node_modules/,
      query: {
        presets: ['es2015', 'react', 'stage-2']
      }
    }, {
      test: /\.json$/, // To load the json files
      loader: 'json-loader'
    }]
  }
}
```

webpack will read this file to generate a single file called ***"build.js"*** combining all the js and css files, to be compatible with new and old browsers.

Create dist/index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href='https://fonts.googleapis.com/css?family=Open+Sans:400,700'
rel='stylesheet' type='text/css'>
  <title>Casino Ethereum Dapp</title>
</head>
<body>
  <div id="root"></div>
  <script src="build.js"></script>
</body>
</html>
```

react code will be inserted here: <div id="root"></div>

Contract: *“Casino.sol”*

```
pragma solidity ^0.4.20;
contract Casino {
	address public owner;
	uint256 public minimumBet;
	uint256 public totalBet;
	uint256 public numberOfBets;
	uint256 public maxAmountOfBets = 100;
	address[] public players;
	struct Player {
		uint256 amountBet;
		uint256 numberSelected;
	}
	// The address of the player and => the user info
	mapping(address => Player) public playerInfo;

	…
}
```

```
function() public payable {}

function Casino(uint256 _minimumBet) public {
    owner = msg.sender;
    if(_minimumBet != 0 ) minimumBet = _minimumBet;
}

function kill() public {
    if(msg.sender == owner) selfdestruct(owner);
}

function checkPlayerExists(address player) public constant
returns(bool) {
    for(uint256 i = 0; i < players.length; i++){
            if(players[i] == player) return true;
    }
    return false;
}
```

```
// Generates a number between 1 and 10 that will be the winner

function generateNumberWinner() public {
    uint256 numberGenerated = block.number % 10 + 1; // This isn't secure
    distributePrizes(numberGenerated);
  }
```

```
// Sends the corresponding ether to each winner depending on the total bets

function distributePrizes(uint256 numberWinner) public {
        address[100] memory winners;
        uint256 count = 0; // This counts the number of winners

        for(uint256 i = 0; i < players.length; i++){
                address playerAddress = players[i];
                if(playerInfo[playerAddress].numberSelected == numberWinner){
                        winners[count] = playerAddress;
                        count++;
                }
                delete playerInfo[playerAddress]; // Delete all the players
        }

        players.length = 0; // Delete all the players array
        uint256 winnerEtherAmount = totalBet / winners.length; // How much each winner gets
        for(uint256 j = 0; j < count; j++){
                if(winners[j] != address(0)) // Check that the address in this fixed array is not empty
                        winners[j].transfer(winnerEtherAmount);
         }
}
```

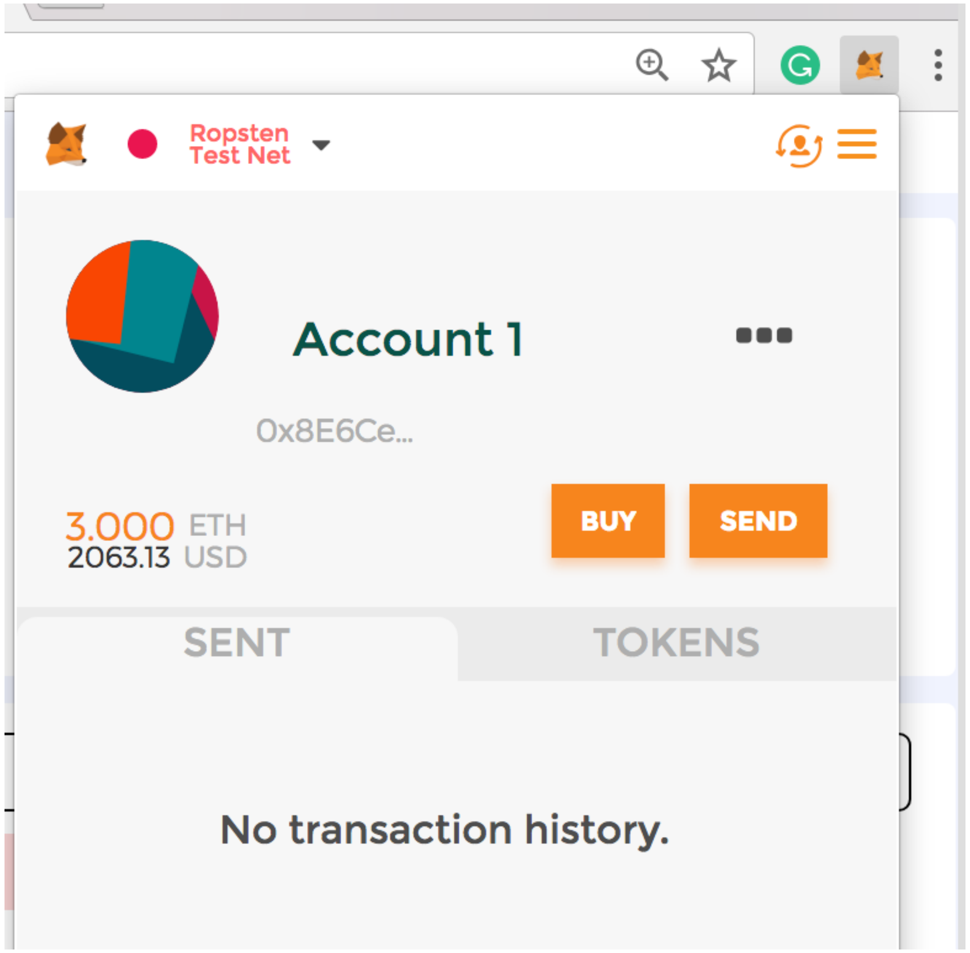
```
// To bet for a number between 1 and 10 both inclusive
function bet(uint256 numChosen) public payable {
    require(!checkPlayerExists(msg.sender));
    require(numChosen >= 1 && numChosen <= 10);
    require(msg.value >= minimumBet);

    playerInfo[msg.sender].amountBet = msg.value;
    playerInfo[msg.sender].numberSelected= numChosen ;

    numberOfBets++;
    players.push(msg.sender);
    totalBet += msg.value;
}
```

Test on a Real Blockchain

- So far we have tested on our local blockchain (running on local host).
- Now, lets test on a real blockchain
- **Testnet**:
  - Ropsten, Rinkeby, Kovan, etc.
  - Used for testing purposes only, with fake Ether.
- **Mainnet** (also called Homestead):
  - The real blockchain used by the entire world with real Ether.

Deploy Contract on Testnet

- Create an account with MetaMask and log in
- Set MetaMask to connect to Ropsten Blockchain Testnet (for testing purposes)
- Need Ether to test. Get some for free at https://faucet.metamask.io/ (wait some time to see this money in MetaMask)

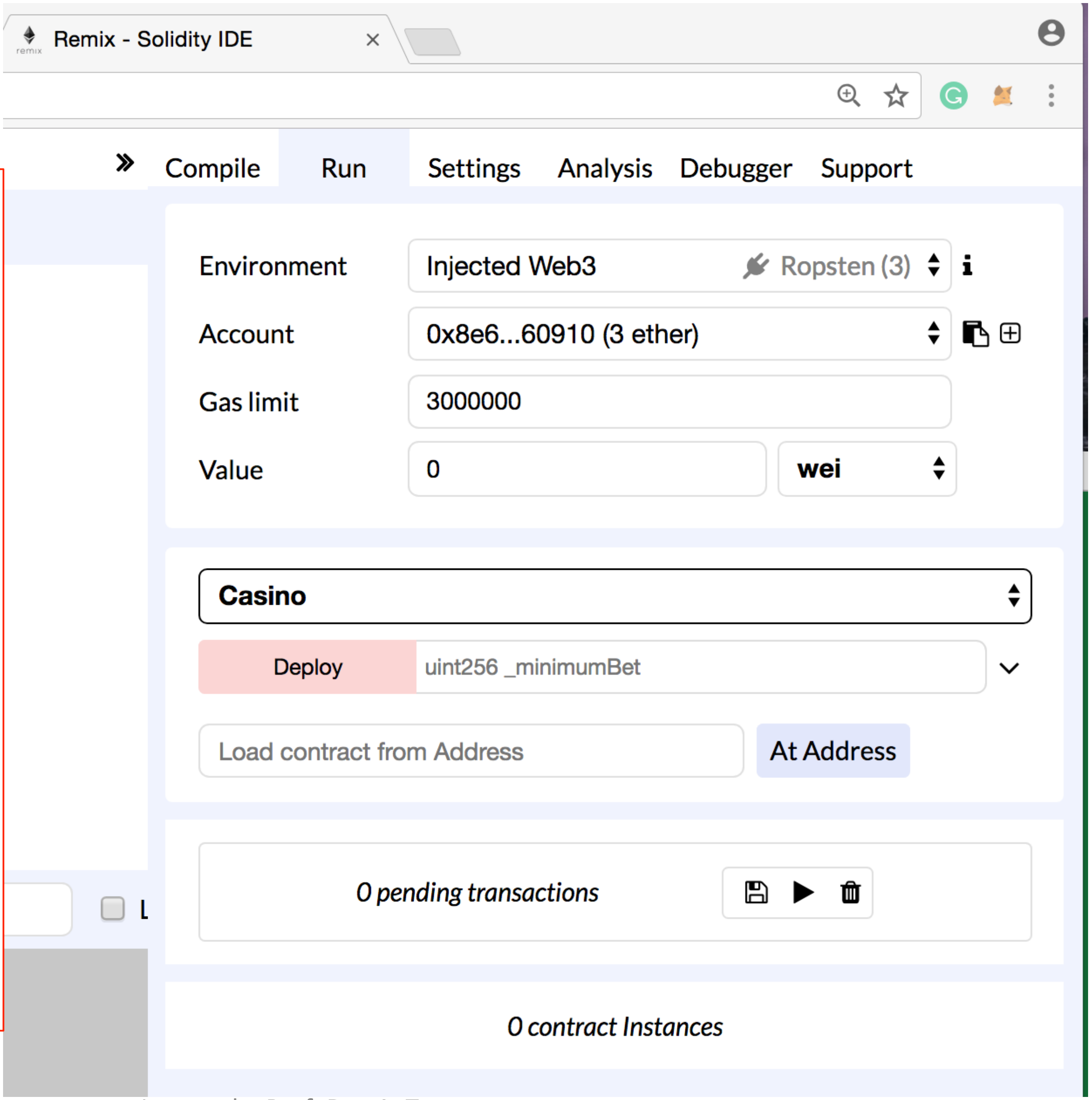


REMIX

remix.ethereum.org

Go to "Run" tab

You will see these boxes populated automatically as a result of logging in MetaMask



Environment: Injected Web3 — Ropsten (3)
Account: 0x8e6...60910 (3 ether)
Gas limit: 3000000
Value: 0 wei

Casino
Deploy — uint256 _minimumBet
Load contract from Address — At Address

0 pending transactions

0 contract Instances

Testnets

- Rinkeby (Geth only)
  - Proof of Authority (PoA), recommended for development (quick mining, consistent)
- Kovan (Parity only)
  - Proof of Authority (PoA), recommended for development (quick mining, consistent)
- Ropsten (Geth and Parity)
  - closest to the Mainnet, uses Proof of Work (PoW) consensus, has been subject to attacks in the past, more problematic for developers
- You can get free ETH to test on these test blockchain networks

Install *geth*

- Remember, ganache-cli is for development
- To run a real node that you own, need *geth*
  - Need to run *geth* to sync with the blockchain network
  - First time running: will take long time, so be patient!

```
$ brew update
$ brew upgrade
$ brew tap ethereum/ethereum
$ brew install ethereum
```

Calling a Contract inside a Contract (1)

- Say, we want to interact with a contract deployed at addr 0x692a70d2e424a56d2c6c27aa97d1a86395877b3a

```
pragma solidity ^0.4.18;
contract SimpleStorage {
    uint public value = 1;
    function set(uint a) public { value = a; }
    function get() view public returns (uint) {
        return value;
    }
}
```

Calling a Contract inside a Contract (2)

- Now, the current contract
- Set _address to 0x692a70d2e424a56d2c6c27aa97d1a86395877b3a to pull out the SimpleStorage contract from the blockchain

```
// need this prototype
contract SimpleStorage {
        function set(uint) public;
        function get() view public returns (uint);
}
contract CurrentContract {
        SimpleStorage deployed_contract;
        constructor(address _address) public {
                deployed_contract = SimpleStorage(_address);
        }
        function get1() public view returns (uint result) {
                return deployed_contract.get();
        }
        function set1(uint _val) public {
                deployed_contract.set(_val);
        }
}
```