

**Environment**

R&D - VoteChain Team

April 4, 2018

**Project Coordinator:** Petteri Kaskenpalo

Lam Quang Vu

**Client:** Hoang Le Minh

**Supervisor:** Le Ngoc Son

**Team Members:** Nguyen Trong Tri

Pham Huynh Tri Minh

Contents

[Frontend 2](#_Toc510623749)

[NodeJS 2](#_Toc510623750)

[Angular 5 4](#_Toc510623751)

[Truffle 6](#_Toc510623752)

[Web3JS 9](#_Toc510623753)

[Backend 10](#_Toc510623754)

[Geth 10](#_Toc510623755)

[Solidity 13](#_Toc510623756)

# Frontend

## NodeJS

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js = Runtime Environment + JavaScript Library

**Concepts:**



**Features:**

* Asynchronous and Event Driven.
* Single Threaded but Highly Scalable.
* No Buffering.
* Very Fast.
* License.

**Where to use Node.js?**

* I/O bound Applications.
* Data Streaming Applications.
* Data Intensive Real-time Applications (DIRT).
* JSON APIs based Applications.
* Single Page Applications.

**Useful Commands:**

* Install plug-in, library, module:

npm install library-name

--global tells NodeJS that install library can use everywhere

--save tells NodeJS that only install library on current folder/project

* Check outdated plug-in, library, module:

npm outdated

* Update plug-in, library, module:

npm update

**Library use in this Project:**  
 If you have cloned our project, run command below on that directory to install all missing library.

npm install

Libraries have been used:

bootstrap web3 ethereum/web3.js truffle-contract truffle @types/node jquery popper.js babel-register @angular/material @angular/cdk ng2-scroll-to @angular/animations

**Link to download:** <https://nodejs.org/en/>

## Angular 5

Angular is a platform that makes it easy to build applications with the web. Angular combines declarative templates, dependency injection, end to end tooling, and integrated best practices to solve development challenges. Angular empowers developers to build applications that live on the web, mobile, or the desktop.

In order to be easy to interact with the Angular, we use Angular CLI. The Angular CLI is a command line interface tool that can create a project, add files, and perform a variety of ongoing development tasks such as testing, bundling, and deployment.

**Start with Angular 5:**

1. Install the Node.js above.
2. Check the version by running **node -v** and **npm -v** in a terminal/console window, the angular require at least node 6.9.x and npm 3.x.x
3. Install Angular CLI

npm install -g @angular/cli

1. Create a new project

ng new Project01

1. Serve the application

cd Project01

ng serve --open

1. Result  
   

**Structure:**

* Component:

src/app/navbar/navbar.component.html //Template

src/app/navbar/navbar.component.spec.ts //Check file component.ts

src/app/navbar/navbar.component.ts //Define function of the component

src/app/navbar/navbar.component.css //Design the template

* Services:

src/app/connect.service.spec.ts //Check file services.ts

src/app/connect.service.ts //Define the services

* Module:

src/app/app-routing.module.ts //Define the services

**Useful Commands:**

* Create Component:   
   Components are the fundamental building blocks of Angular applications. They display data on the screen, listen for user input, and take action based on that input.

ng generate component navbar

* Create Services:   
   Components shouldn't fetch or save data directly and they certainly shouldn't knowingly present fake data. They should focus on presenting data and delegate data access to a service.

ng generate service connect --module=app

--module=app tells the CLI to register it in the imports array of the AppModule.

* Create Module Routing:  
   An Angular best practice is to load and configure the router in a separate, top-level module that is dedicated to routing and imported by the root AppModule.

ng generate module app-routing --flat --module=app

--flat puts the file in src/app instead of its own folder.

**For more information:** [**https://angular.io/docs**](https://angular.io/docs)

## Truffle

Truffle is written in JavaScript in a completely modular fashion, allowing you to pick and choose the functionality you’d like to use. Truffle is also a world-class development environment, testing framework and asset pipeline for Ethereum, aiming to make life as an Ethereum developer easier.

**Features:**

With Truffle, you get:

* Built-in smart contract compilation, linking, deployment and binary management.
* Automated contract testing for rapid development.
* Scriptable, extensible deployment & migrations framework.
* Network management for deploying to any number of public & private networks.
* Package management with EthPM & NPM, using the ERC190 standard.
* Interactive console for direct contract communication.
* Configurable build pipeline with support for tight integration.
* External script runner that executes scripts within a Truffle environment.

**Install Truffle inside Angular 5:**

1. Check the version by running **node -v** and **npm -v** in a terminal/console window, the angular require at least node 5.x.x and npm 3.x.x. Truffle also requires that you have a running Ethereum client that supports the standard JSON RPC API (will describe below in section Backend).
2. Already install NodeJS, Angular CLI and locating at Angular Project directory.

npm install truffle-contract truffle --save

**Structure:**

* Folder:

contracts //Store all contracts .sol

build //Store all compiled contracts .json

migrations //Store all files define how to deploy contracts

* 1\_initial\_migration.js:

var Migrations = artifacts.require("./Migrations.sol");

module.exports = function(deployer) {

deployer.deploy(Migrations, {gas: 4700000});

};

//Migrate the default contract of Truffle

* 2\_deploy\_contracts.js:

var Voting = artifacts.require("./Voting.sol");

var ManageAccount = artifacts.require("./ManageAccount.sol");

module.exports = function(deployer) {

deployer.deploy(ManageAccount, {gas: 4700000});

deployer.link(ManageAccount, Voting);

deployer.deploy(Voting, {gas: 4700000});

};

// Deploy library ManageAccount, then link ManageAccount to Voting, then Voting.

// Define the gas need to deploy each of them

**Useful Commands:**

* Compile Solidity Language:   
   Solidity language is a programming language that fully support to write a Smart Contract. In order to push the Contract to the Network, it need to be compile into bytecodes and return an ABI (json) file to interact with the frontend.

truffle compile

* Migrate Smart Contract:  
   After convert from Solidity to Bytecode, the next step is migrate the bytecode into the EVM (Ethereum Virtual Machine) on the Network. When the contract is in the EVM, who have the ABI file can access and use the contract.

truffle migrate

--reset in order to migrate the new contract and replace the old ABI file.

**Caution:** - The Ethereum client must be running when truffle compile and migrate.  
 - Unlock account and have enough ETH to migrate.  
 - Miners are working to bring the contract into the EVM.

**How to use**

Define in file Connect Services

const contract = require('truffle-contract');

const VotingABI = require('../../build/contracts/Voting.json');

//Read the ABI from json file

VotingContract = contract(VotingABI);

//Add all ABI into VotingContract in order to call as Function

constructor() {

this.VotingContract.setProvider(this.web3.currentProvider);

}

//Set the Provider to start interact with the Contract

**For more information:** <http://truffleframework.com/docs/getting_started/installation>

## Web3JS

The web3.js library is a collection of modules that contain specific functionality for the ethereum ecosystem.

**Features:**

* web3-eth is for the ethereum blockchain and smart contracts.
* web3-shh is for the whisper protocol to communicate p2p and broadcast.
* web3-bzz is for the swarm protocol, the decentralized file storage.
* web3-utils contains useful helper functions for Dapp developers.

**Install Web3JS inside Angular 5:**

Already install NodeJS, Angular CLI and locating at Angular Project directory.

npm install web3 ethereum/web3.js --save

**How to use**

Define in file Connect Services

const Web3API = require('web3');

//Read the API from web3

web3: any; //Store the protocol to connect with Ethereum Network

constructor() {

if (typeof window.web3 !== 'undefined') {

this.web3 = new Web3API(window.web3.currentProvider);

//Check if using Mist or Metamask

}

else {

this.web3 = new Web3API(new Web3API.providers.HttpProvider('http://localhost:8545'));

//Using localhost

}

}

**For more information:** [**http://web3js.readthedocs.io/en/1.0/getting-started.html**](http://web3js.readthedocs.io/en/1.0/getting-started.html)

# Backend

## Geth

Go Ethereum is one of the three original implementations (along with C++ and Python) of the Ethereum protocol. It is written in Go, fully open source and licensed under the GNU LGPL v3.

Go Ethereum is available either as a standalone client called Geth that you can install on pretty much any operating system, or as a library that you can embed in your Go, Android or iOS projects.

**Install Geth:**

Geth are built and available for download at <https://geth.ethereum.org/downloads/>

1. Download zip file
2. Extract geth.exe from zip
3. Open geth.exe

**Structure:**

* Folder:

ChainData

|--------geth //Store all chain data, transactions, nodes info

|--------keystore //Store all created account in local.

* Genesis File:   
  A genesis block is the first block of a block chain. The genesis block is usually hardcoded into the software of the applications that utilize its block chain. It is a special case in that it does not reference a previous block.

//Example file CustomGenesis.json

{

"config": {

"chainId": 1994,

"homesteadBlock": 0,

"eip155Block": 0,

"eip158Block": 0,

"byzantiumBlock": 0

},

"difficulty": "400",

"gasLimit": "4700000",

"alloc": {

        "7881a23c2eaa4e0ef9bae2b8e6abc56213187d0b": { "balance": "1000000000000000000000000000000" },

        "afd3c28630a3bdc8777f674d1022d118d34242b0": { "balance": "1000000000000000000000000000000" }

}

}

**Useful Commands:**

* Create Genesis Block and Network:

geth –datadir=DataChain init CustomGenesis.json

* Launch the Network:

geth --datadir=DataChain --rpc --rpcapi db,net,web3,personal,eth --rpccorsdomain="\*" --networkid=1994

--rpc //Using protocol RPC

--rpcapi //Using API listed

--rpccoordomain //Connect to the url

--networkid //Network ID

* Access Geth Console:

geth attach

* In Geth Console
  + Create Account:

personal.newAccount(“password”)

* + Set Miner:

miner.setEtherbase(“address”)

* + Start/Stop Miner:

miner.start()

miner.stop()

* + List Account:

eth.accounts

* + Unlock Account:

personal.unlockAccount(“address”,”password”,timeout)

* + Node Information:

admin.nodeInfo

* + Peer Information:

admin.peers

* + Send Transaction:

eth.sendTransactoin({from:sender, to:receiver, value: amount})

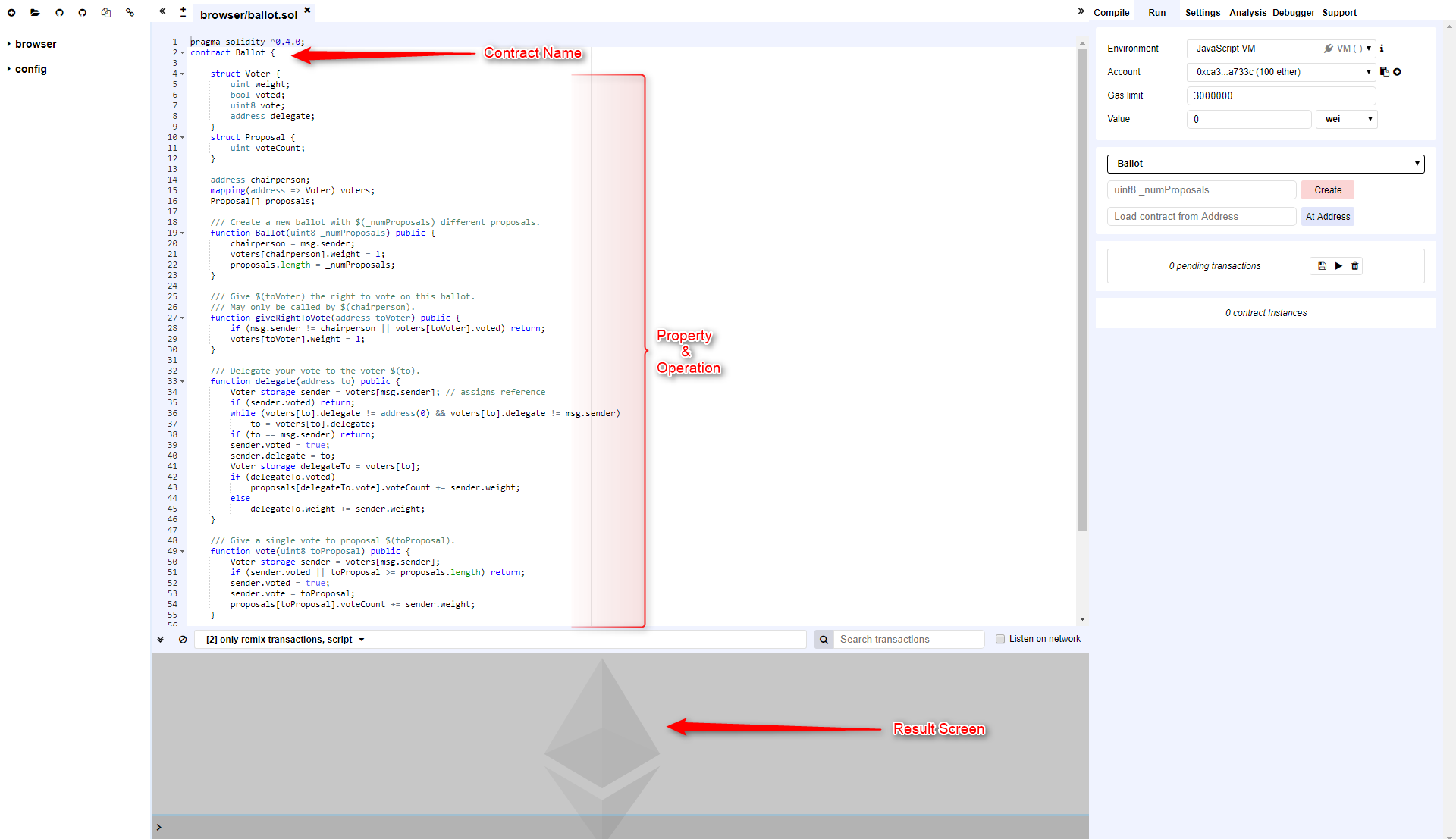
**For more information:** <https://github.com/ethereum/go-ethereum>

## Solidity

Solidity is a contract-oriented, high-level language for implementing smart contracts. It was influenced by C++, Python and JavaScript and is designed to target the Ethereum Virtual Machine (EVM).

Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features.

**IDE:**

****

**For more information:**

[**https://remix.ethereum.org**](https://remix.ethereum.org)

[**http://solidity.readthedocs.io/en/v0.4.21/index.html**](http://solidity.readthedocs.io/en/v0.4.21/index.html)

[**https://www.youtube.com/channel/UCaWes1eWQ9TbzA695gl\_PtA/videos**](https://www.youtube.com/channel/UCaWes1eWQ9TbzA695gl_PtA/videos)