KAIST
School of Computing

Web3@KAIST
Building Web3 Apps to Solve Real Problems

Building Web3 & Blockchain Applications
(CS492 Special Topics in Computer Science)
Spring 2023

Building Dapp: The Process

Lecture 9 (2023-04-05)

Jason Han, Ph.D
Adjunct Professor of KAIST School of Computing
Founder of Ground X & Klaytn

web3classdao@gmail.com
http://web3classdao.xyz/kaist/

Today’s Lecture 9 & 10 Overview

- **Lecture Objective**
  - Understanding the entire process of building Dapp
  - Building the ability to develop and deploy Dapp samples
  - Learning various dev tools

- **Lecture will cover**
  - Dapp development process
  - Phased contract deployment
  - Remix, truffle & Ganache, hardhat
  - Ethereum JavaScript library: ethers.js
  - Dapp samples: online voting, token

Note before we get started

Today’s lecture is geared toward beginners.
The goal is to let them try out Dapp samples.

Today's lecture contains a lot of content.
Use it as a resource for self-study later.

I’m not a seasoned Dapp developer.
Don’t hesitate to comment if there are any errors.
Feel free to answer questions if you know.

References for the lecture

- Blockchain in Action (by Bina Ramamurthy) (Online voting example, outdated)
- Hardhat Tutorial (Token example)
- Web3 developer guide and overview from Alchemy
- Web3 tutorial from Alchemy
- ethers.js official documentation
- web3.js official documentation
- solidity official documentation
- MetaMask developer documentation
- OpenZeppelin documentation
- Ethereum development tutorials compiled by Ethereum Founation
- Ultimate Web3, Full Stack Solidity, and Smart Contract Course by Patrick Collins

A simple Ballot Dapp

Example from a book of 'Blockchain in Action'
with some modification

It's simple, and easy to understand
Good for walking through the entire process of Dapp dev

Clone the code here!

```
git clone https://github.com/web3classdao/ballot-truffle.git
```

Use Case: Online Voting

Problem

Online voting is convenient,
but it's also highly susceptible to manipulation.

Solution

Transparent and tamper-proof online voting
can be implemented on the blockchain.

Why Blockchain?

Problems with traditional online voting

1) Manipulation of voting authorities (e.g., Produce 101)
2) Distrust of voting results (e.g., Political elections)
3) Pressure on voting authorities (governance issue)

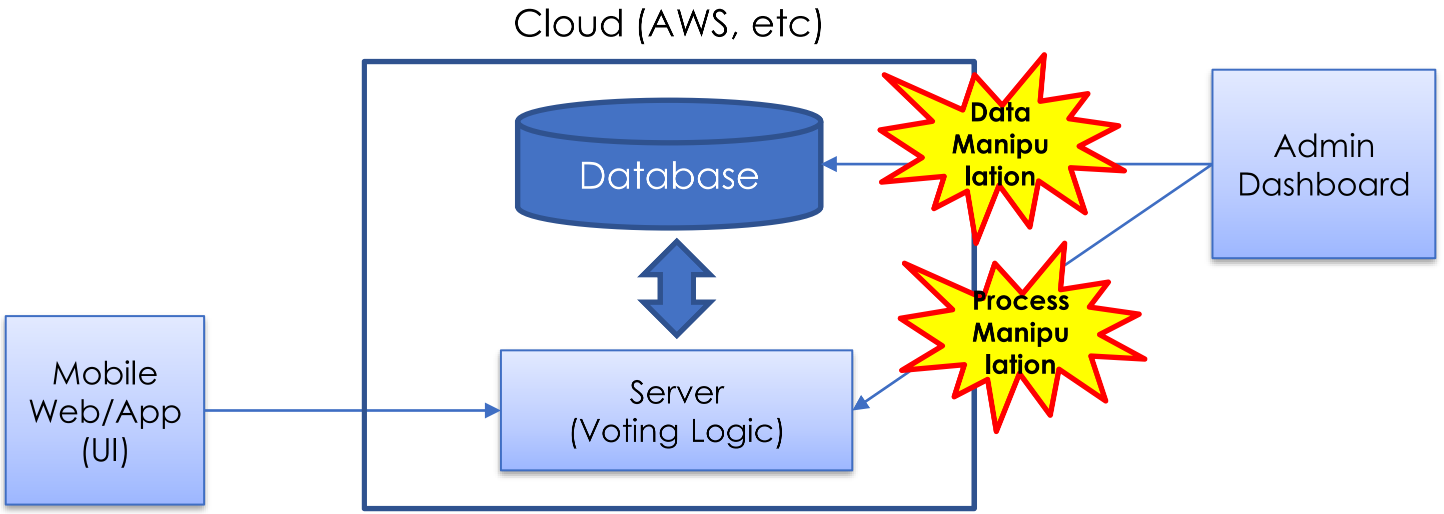
Online voting with smart contracts

- No post-deployment logic changes
- No manipulation of voting data

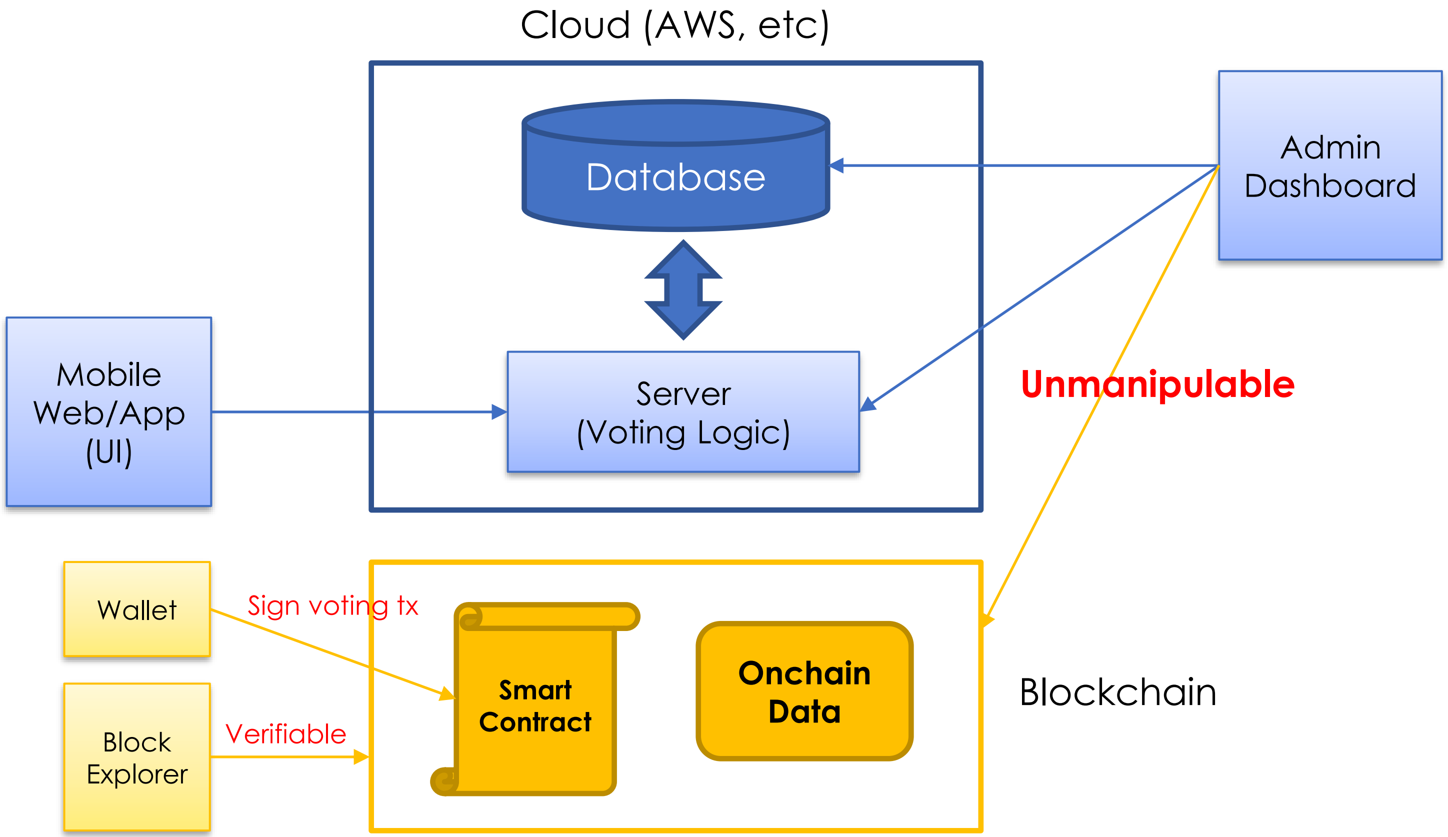
→ Increase trust in online voting
even if you don't trust the voting authority

※ Not decentralized online voting, but using blockchain as a foundation layer to improve trust

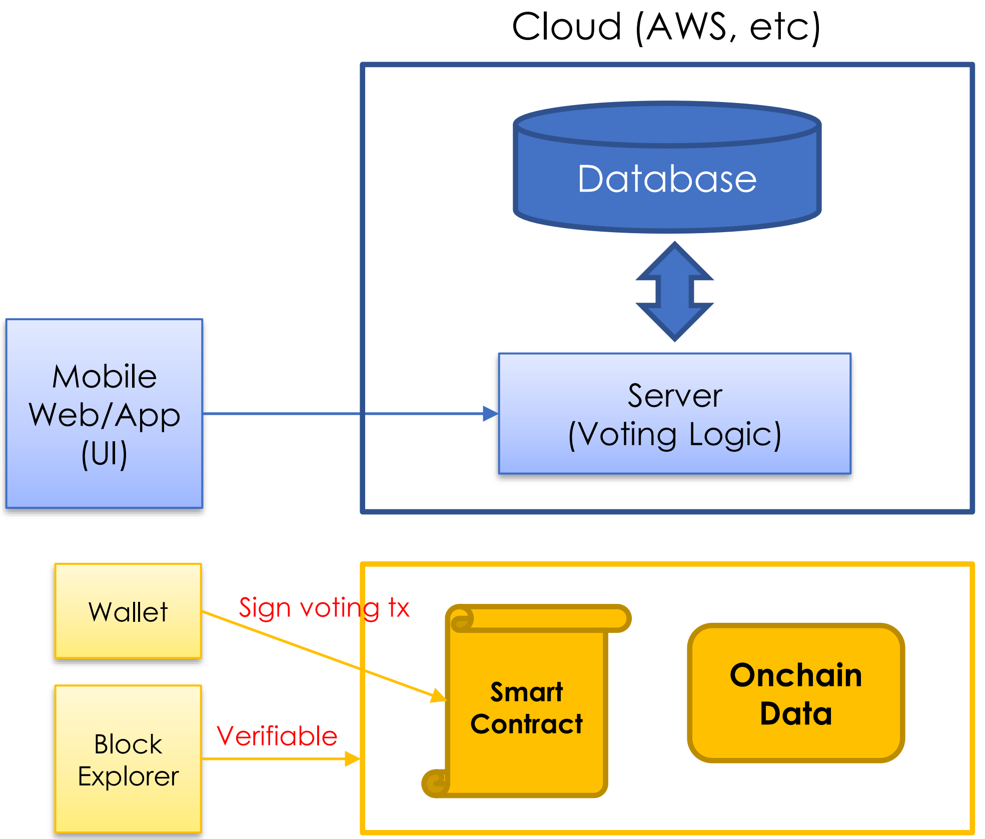
Implementing online voting in the traditional way?



Implementing online voting based on blockchain?



Separating onchain and offchain data



Offchain data
- Voter profile
- Proposal data

Onchain data
- Voting data (address, weight, voting etc)
- Voting result

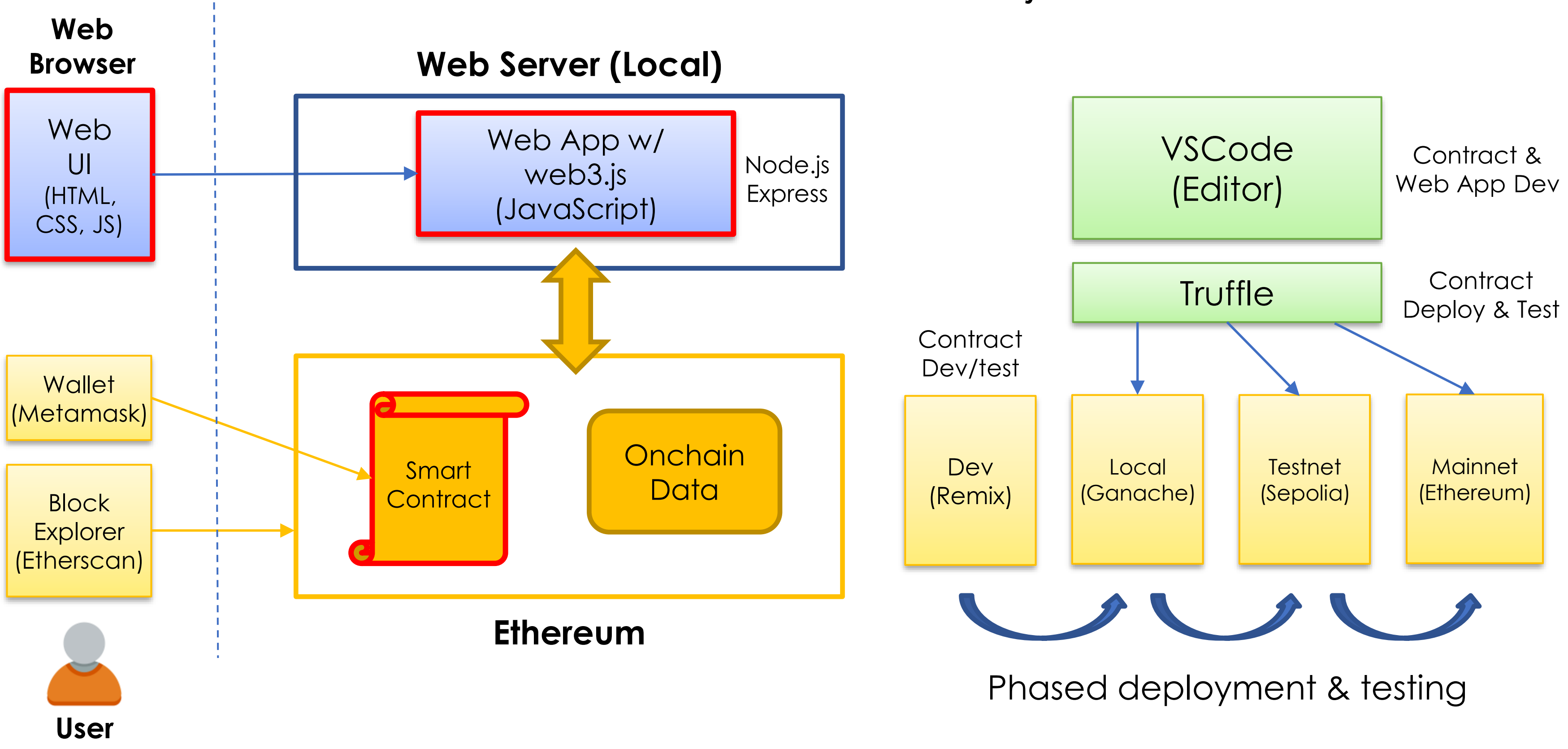
Let's take a look at
the entire process of
developing an online voting Dapp
as an example.

Dapp Development Process

1. Design
2. Develop smart contracts with Remix
3. Deploy & test smart contracts (Local)
4. Develop a web app
5. Deploy & test all (Local)
6. Deploy & test all (Testnet)
7. Deploy & test all (Mainnet)

Dapp Development Environment

based on truffle & web3.js

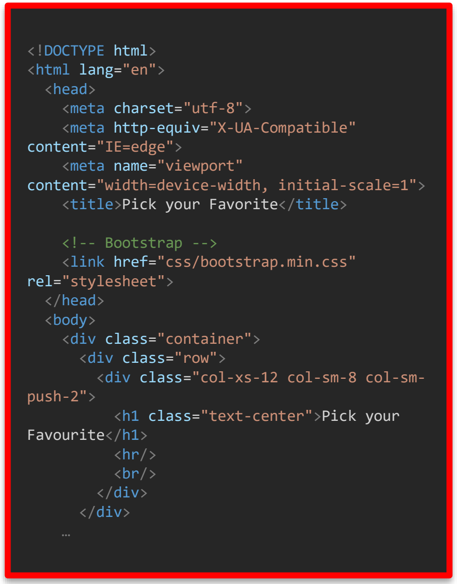


Toolsets that we will use

1. Package manager: npm
2. Web server for the web app: node.js & Express
3. Smart contract IDE: truffle & web3.js
4. Web browser & wallet: Chrome & Metamask
5. Local testnet: Ganache
6. Public testnet: Sepolia
7. Code Editor: VSCode

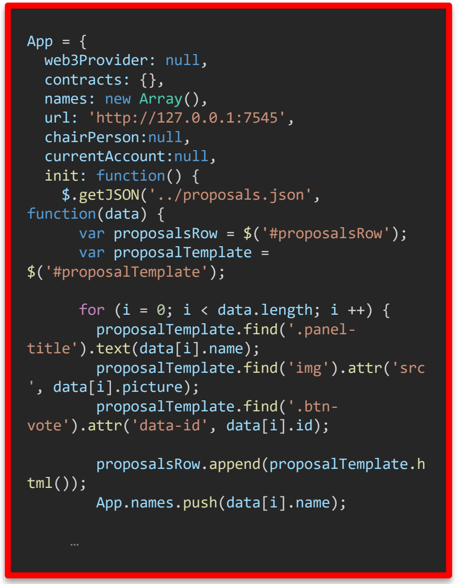
What we develop in this lecture

Web UI
(index.html)



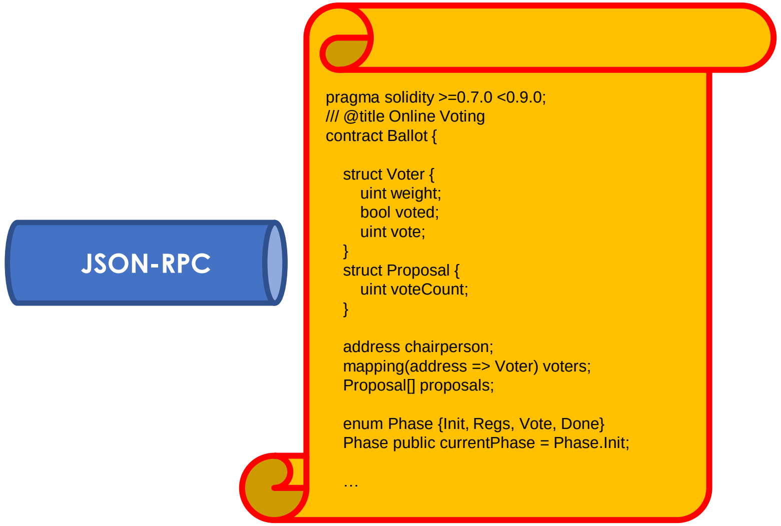
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible"
content="IE=edge">
    <meta name="viewport"
content="width=device-width, initial-scale=1">
    <title>Pick your Favorite</title>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css"
rel="stylesheet">
  </head>
  <body>
    <div class="container">
      <div class="row">
        <div class="col-xs-12 col-sm-8 col-sm-
push-2">
          <h1 class="text-center">Pick your
Favourite</h1>
          <hr/>
          <br/>
        </div>
      </div>
    …
```

Web App
(app.js)



```
App = {
  web3Provider: null,
  contracts: {},
  names: new Array(),
  url: 'http://127.0.0.1:7545',
  chairPerson:null,
  currentAccount:null,
  init: function() {
    $.getJSON('../proposals.json',
function(data) {
      var proposalsRow = $('#proposalsRow');
      var proposalTemplate =
$('#proposalTemplate');

      for (i = 0; i < data.length; i ++) {
        proposalTemplate.find('.panel-
title').text(data[i].name);
        proposalTemplate.find('img').attr('src
', data[i].picture);
        proposalTemplate.find('.btn-
vote').attr('data-id', data[i].id);

        proposalsRow.append(proposalTemplate.h
tml());
        App.names.push(data[i].name);

    …
```

Smart contract
(Ballot.sol)



```
pragma solidity >=0.7.0 <0.9.0;
/// @title Online Voting
contract Ballot {

  struct Voter {
    uint weight;
    bool voted;
    uint vote;
  }
  struct Proposal {
    uint voteCount;
  }

  address chairperson;
  mapping(address => Voter) voters;
  Proposal[] proposals;

  enum Phase {Init, Regs, Vote, Done}
  Phase public currentPhase = Phase.Init;

  …
```

1. Design

Problem Statement: Defining Problem

Online ballot application

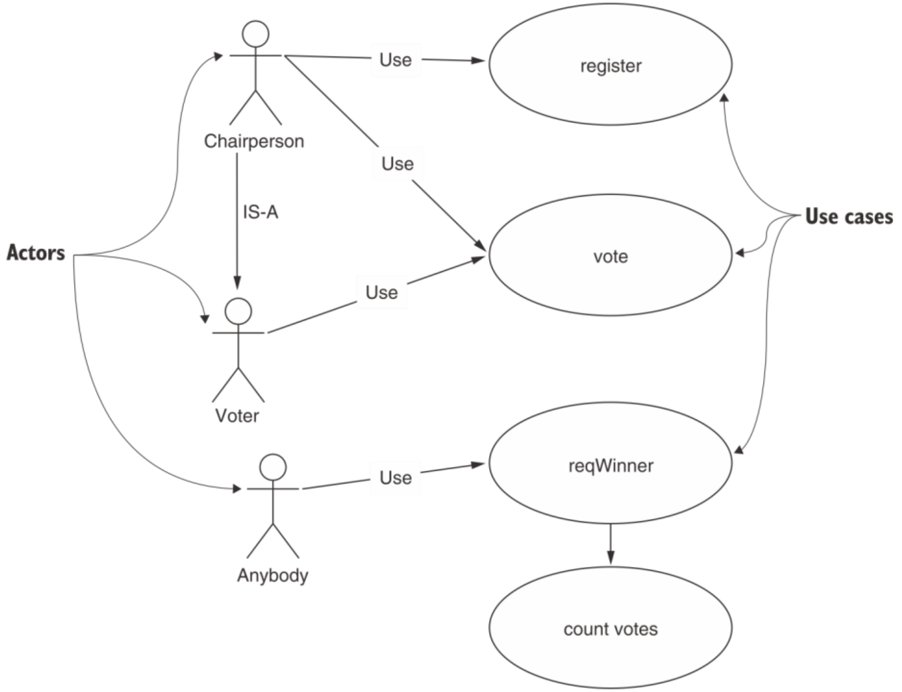
- People vote to choose a proposal from a set of proposals
- A chairperson registers the people who can vote
- Only registered voters can vote (only once) on a proposal of their choice
- The chairperson's vote is weighted twice (x2) as heavily as regular people's votes
- The ballot process goes through four phases (Init, Regs, Vote, Done)
- The respective operations(Initialize, register, vote, count votes) can be performed only in the corresponding phase

Analyzing Problem Statement

- **Roles → Use case diagram**
  - voter
  - chairperson
  - anybody
- **Rules (Constraints) → Modifier**
  - Only chairperson can register voters
  - Only chairperson can change voting phase
  - Only registered voters can vote
  - The respective operations can be performed only in the corresponding phase
- **Assets → Data**
  - voters
  - chairperson
  - proposal
- **States → FSM diagram**
  - Init, Regs, Vote, Done
- **Events → Events**
  - Regs started
  - Vote started
  - Vote done

Use Case Diagram

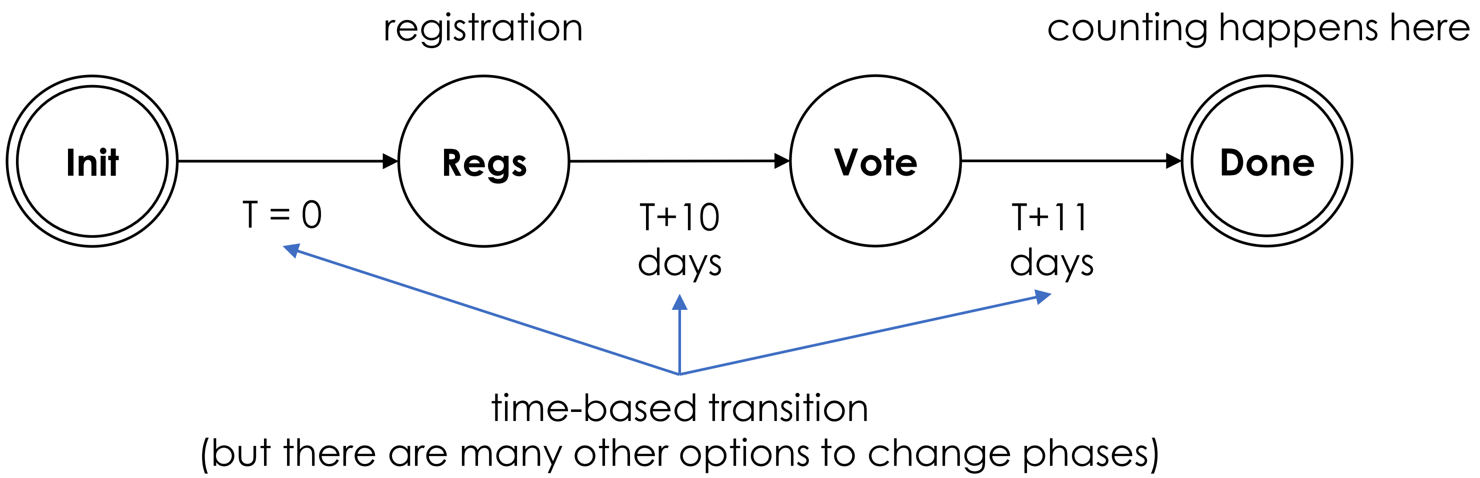
Identifying the users, assets and transactions



Finite State Machine (FSM) Diagram

Representing system dynamics such as state transitions within a smart contract

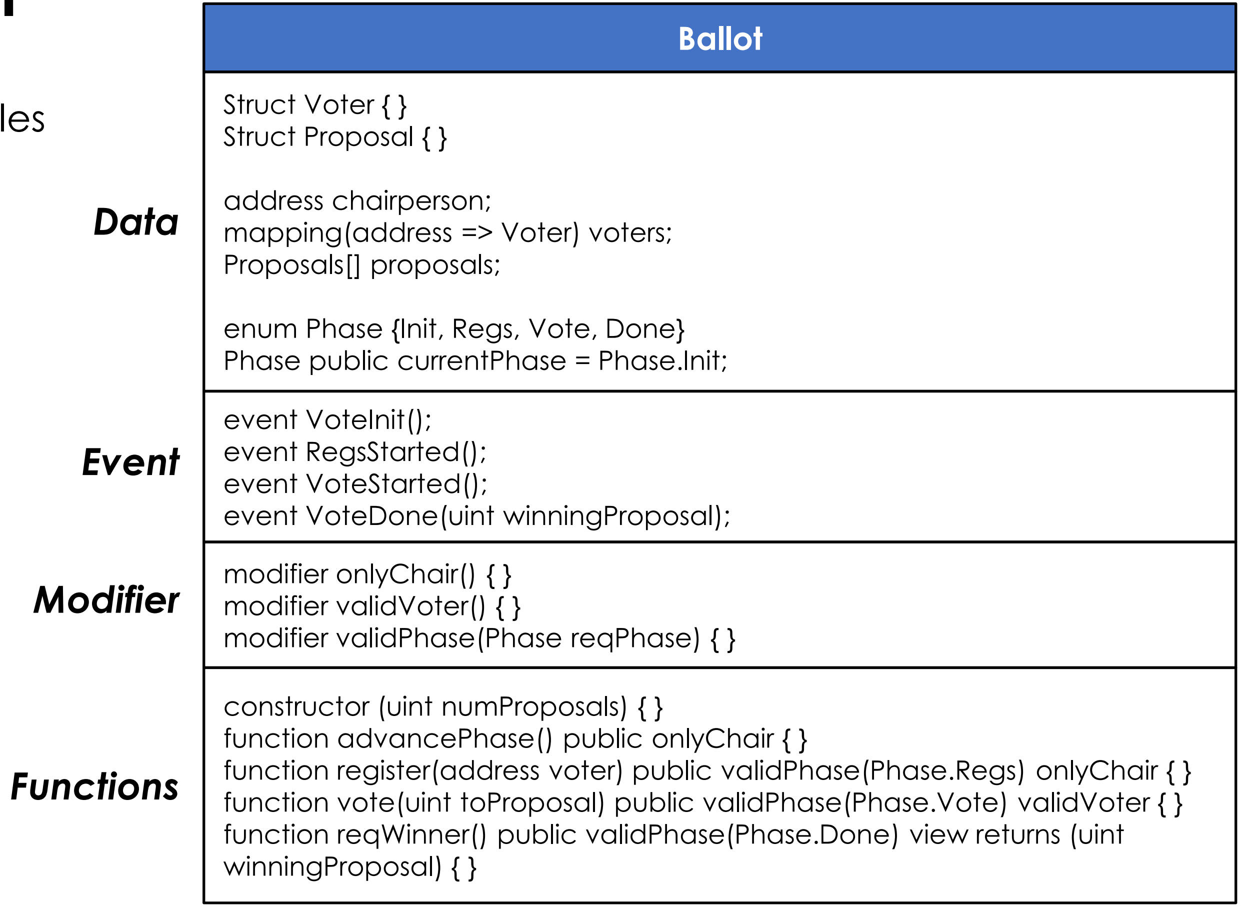
Each phase can specify the allowed operations



Only a chairperson can change phases

Contract Diagram

Specifying the name,
data assets, functions, and rules
for execution of functions
and access to the data



Ballot

Data

```
Struct Voter { }
Struct Proposal { }

address chairperson;
mapping(address => Voter) voters;
Proposals[] proposals;

enum Phase {Init, Regs, Vote, Done}
Phase public currentPhase = Phase.Init;
```

Event

```
event VoteInit();
event RegsStarted();
event VoteStarted();
event VoteDone(uint winningProposal);
```

Modifier

```
modifier onlyChair() { }
modifier validVoter() { }
modifier validPhase(Phase reqPhase) { }
```

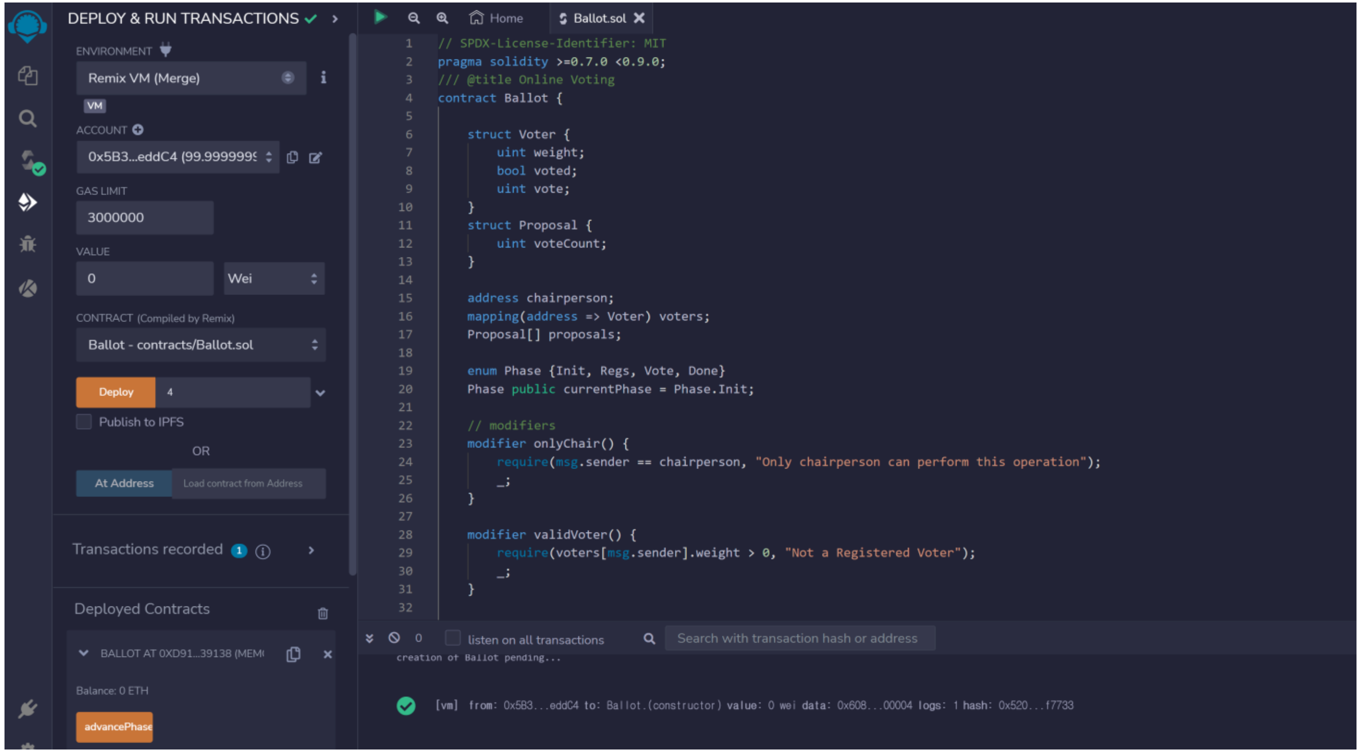
Functions

```
constructor (uint numProposals) { }
function advancePhase() public onlyChair { }
function register(address voter) public validPhase(Phase.Regs) onlyChair { }
function vote(uint toProposal) public validPhase(Phase.Vote) validVoter { }
function reqWinner() public validPhase(Phase.Done) view returns (uint
winningProposal) { }
```

2. Develop smart contract with Remix

Remix: develop and test Solidity codes

Online smart contract development environment
with the simulated Ethereum network


DEPLOY & RUN TRANSACTIONS
Home
Ballot.sol
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded 1
Deployed Contracts
advancePhase
listen on all transactions
Search with transaction hash or address

ENVIRONMENT: Remix VM (Merge)
ACCOUNT: 0x5B3...eddC4 (99.999999
GAS LIMIT: 3000000
VALUE: 0 Wei
CONTRACT (Compiled by Remix): Ballot - contracts/Ballot.sol
Deploy: 4
BALLOT AT 0XD91...39138 (MEM
Balance: 0 ETH

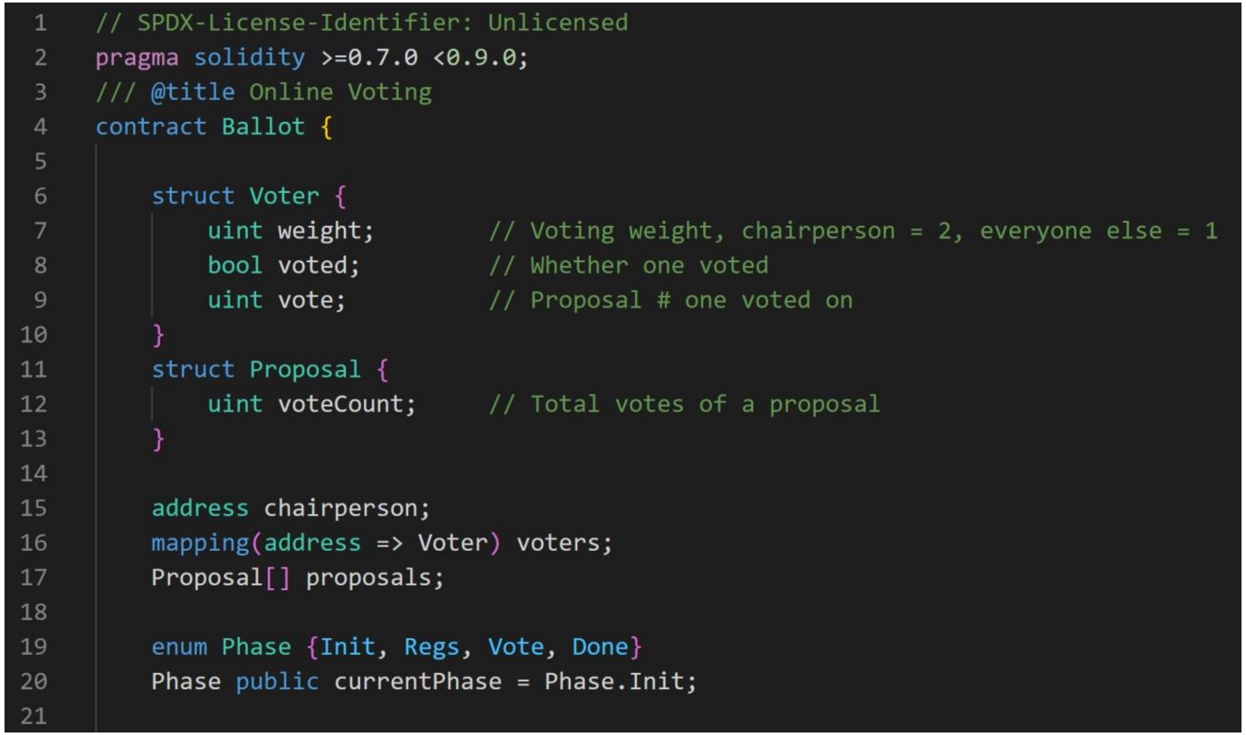
```solidity
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;
/// @title Online Voting
contract Ballot {

    struct Voter {
        uint weight;
        bool voted;
        uint vote;
    }
    struct Proposal {
        uint voteCount;
    }

    address chairperson;
    mapping(address => Voter) voters;
    Proposal[] proposals;

    enum Phase {Init, Regs, Vote, Done}
    Phase public currentPhase = Phase.Init;

    // modifiers
    modifier onlyChair() {
        require(msg.sender == chairperson, "Only chairperson can perform this operation");
        _;
    }

    modifier validVoter() {
        require(voters[msg.sender].weight > 0, "Not a Registered Voter");
        _;
    }

```

```
creation of Ballot pending...

[vm] from: 0x5B3...eddC4 to: Ballot.(constructor) value: 0 wei data: 0x608...00004 logs: 1 hash: 0x520...f7733
```

https://remix.ethereum.org/

Programming Data Items

Identifying the users and data assets

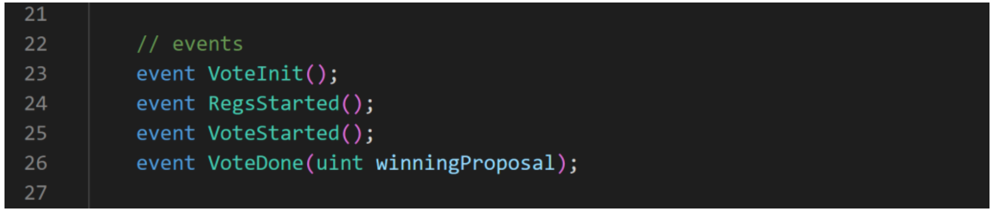


```solidity
// SPDX-License-Identifier: Unlicensed
pragma solidity >=0.7.0 <0.9.0;
/// @title Online Voting
contract Ballot {

    struct Voter {
        uint weight;        // Voting weight, chairperson = 2, everyone else = 1
        bool voted;         // Whether one voted
        uint vote;          // Proposal # one voted on
    }
    struct Proposal {
        uint voteCount;     // Total votes of a proposal
    }

    address chairperson;
    mapping(address => Voter) voters;
    Proposal[] proposals;

    enum Phase {Init, Regs, Vote, Done}
    Phase public currentPhase = Phase.Init;

```

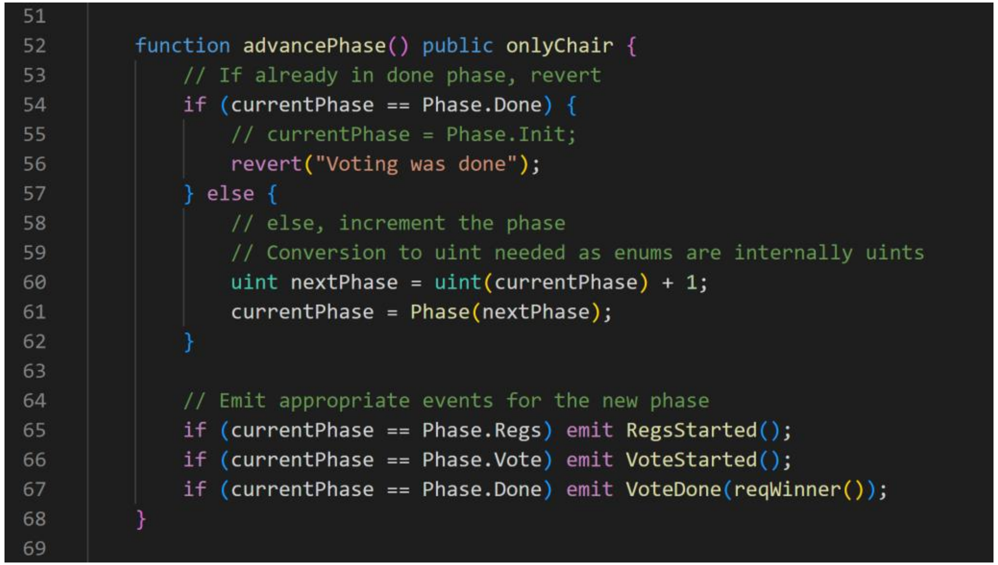
https://docs.soliditylang.org/en/v0.8.17/types.html

Programming State Transitions

Implementing a function and events for state transitions



```

    // events
    event VoteInit();
    event RegsStarted();
    event VoteStarted();
    event VoteDone(uint winningProposal);

```

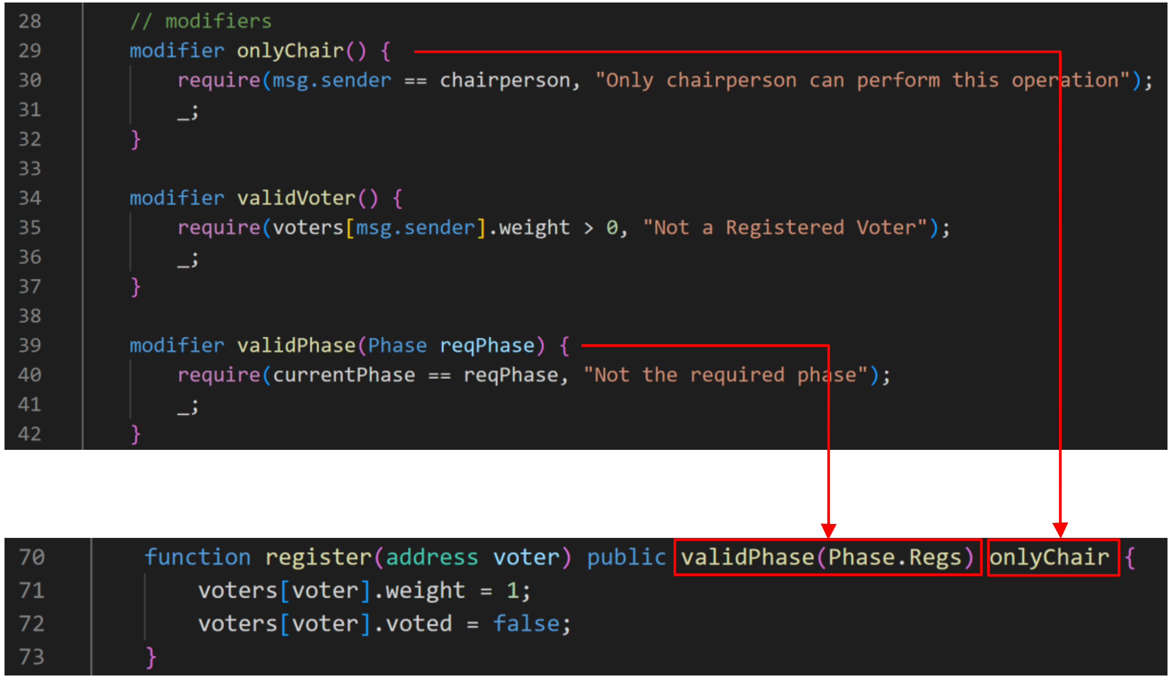


```

    function advancePhase() public onlyChair {
        // If already in done phase, revert
        if (currentPhase == Phase.Done) {
            // currentPhase = Phase.Init;
            revert("Voting was done");
        } else {
            // else, increment the phase
            // Conversion to uint needed as enums are internally uints
            uint nextPhase = uint(currentPhase) + 1;
            currentPhase = Phase(nextPhase);
        }

        // Emit appropriate events for the new phase
        if (currentPhase == Phase.Regs) emit RegsStarted();
        if (currentPhase == Phase.Vote) emit VoteStarted();
        if (currentPhase == Phase.Done) emit VoteDone(reqWinner());
    }

```

https://www.alchemy.com/overviews/solidity-events

Programming Modifiers

A modifier is a special type of Solidity function
that is used to modify the behavior of other functions
Check that a certain condition is met before allowing the function to execute



```
// modifiers
modifier onlyChair() {
    require(msg.sender == chairperson, "Only chairperson can perform this operation");
    _;
}

modifier validVoter() {
    require(voters[msg.sender].weight > 0, "Not a Registered Voter");
    _;
}

modifier validPhase(Phase reqPhase) {
    require(currentPhase == reqPhase, "Not the required phase");
    _;
}
```

```
function register(address voter) public validPhase(Phase.Regs) onlyChair {
    voters[voter].weight = 1;
    voters[voter].voted = false;
}
```

https://www.alchemy.com/overviews/solidity-modifier

Solidity Error Handling

Error handling in Solidity ensures atomicity as a property
When a smart contract call terminates with an error, all the state changes are reverted
Three special functions for error handling: require, assert, revert

require()
- act as a gate check modifier verifying inputs and conditions before execution
- ideal for logic flow gating and validating user inputs on functions
- if failed, the unused gas is returned to the caller and the state is reversed to the original state

revert()
- identical to require() without evaluating any condition
- useful for more complex logic flow gates (i.e., complicated if-then blocks)
- if called, the unused gas is returned and the state reverts to its original state

assert()
- used to check for code that should never be false
- play an important role in preventing impossible scenarios
- don’t return any unused gas and instead, will consume the gas supply

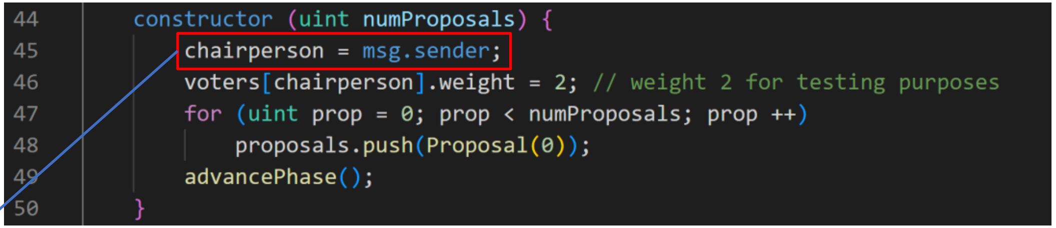
https://www.alchemy.com/overviews/solidity-require

Programming Functions

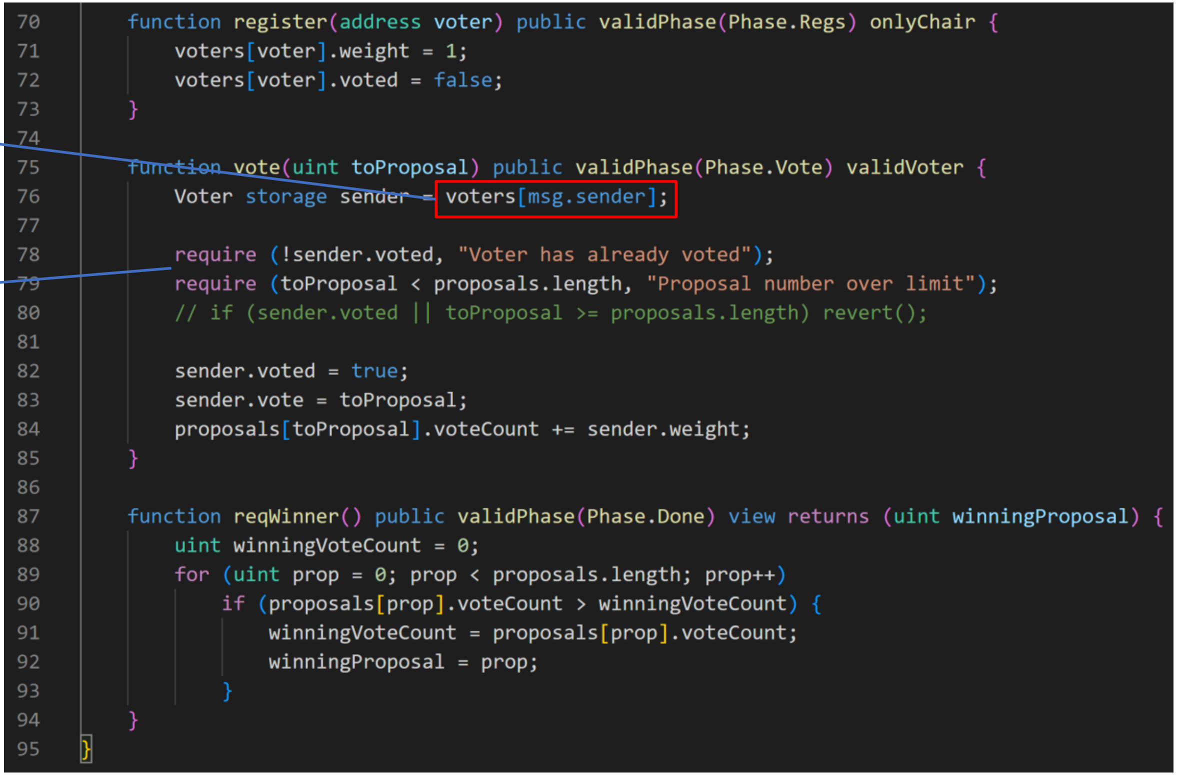
msg.sender
= contractor deployer
= contract owner

msg.sender
the address that has called or initiated a function(vote)

require()
check the condition and inputs



```
    constructor (uint numProposals) {
        chairperson = msg.sender;
        voters[chairperson].weight = 2; // weight 2 for testing purposes
        for (uint prop = 0; prop < numProposals; prop ++)
            proposals.push(Proposal(0));
        advancePhase();
    }
```



```
    function register(address voter) public validPhase(Phase.Regs) onlyChair {
        voters[voter].weight = 1;
        voters[voter].voted = false;
    }

    function vote(uint toProposal) public validPhase(Phase.Vote) validVoter {
        Voter storage sender = voters[msg.sender];

        require (!sender.voted, "Voter has already voted");
        require (toProposal < proposals.length, "Proposal number over limit");
        // if (sender.voted || toProposal >= proposals.length) revert();

        sender.voted = true;
        sender.vote = toProposal;
        proposals[toProposal].voteCount += sender.weight;
    }

    function reqWinner() public validPhase(Phase.Done) view returns (uint winningProposal) {
        uint winningVoteCount = 0;
        for (uint prop = 0; prop < proposals.length; prop++)
            if (proposals[prop].voteCount > winningVoteCount) {
                winningVoteCount = proposals[prop].voteCount;
                winningProposal = prop;
            }
    }
}
```

Testing smart contract in Remix

- **Positive tests: verify that the behavior works as expected given valid input**
  - Chairperson registers three voters
  - Chairperson changes to Vote phase
  - Chairperson votes on a specific proposal
  - The remaining voters (addresses) also vote on a specific proposal
  - Chairperson changes to Done phase
  - Call the voting results at any address to verify that the results are correct
- **Negative testing: check and validate to catch errors and revert functions when given invalid input**
  - Non-chairperson address calls register (onlyChair())
  - Attempt to vote in Regs phase (validPhase())
  - Attempt to vote from an unregistered address (validVoter())
  - Invalid proposal voting attempt (require())

Video-01
Compiling and testing contracts in Remix

3. Deploy & test smart contract (Local)

Truffle: Dapp development framework

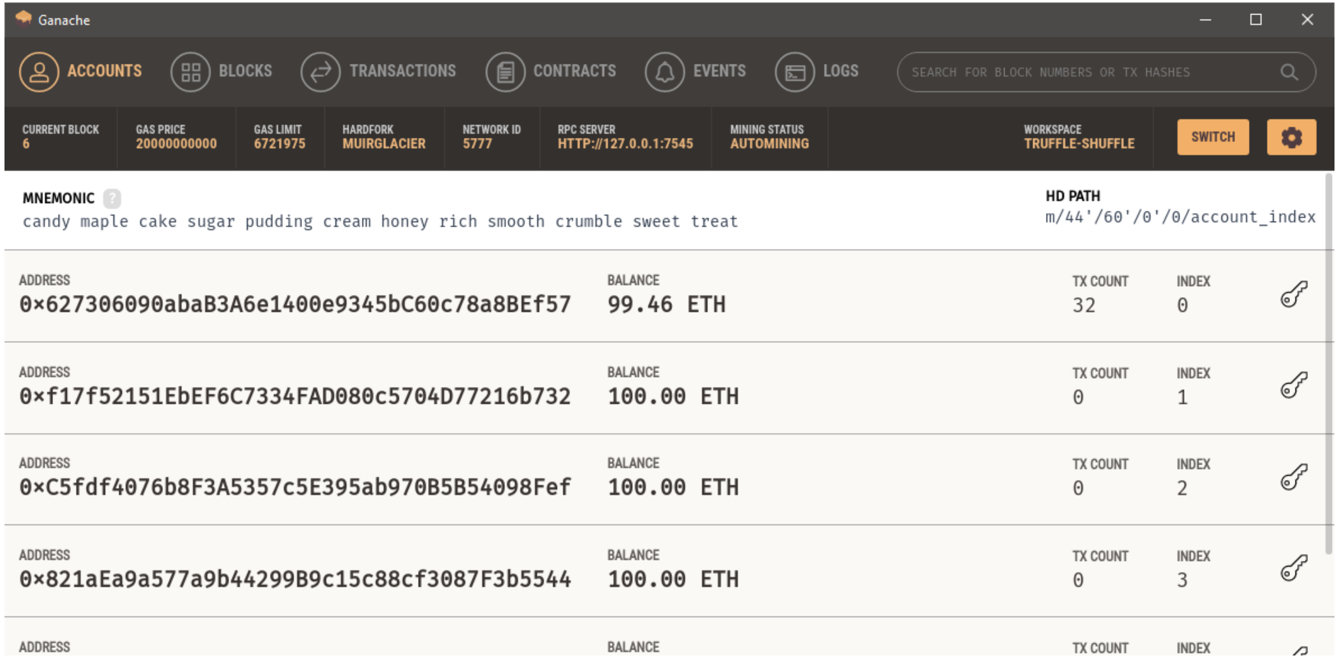
A world class development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM)

- 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.
- Advanced debugging with breakpoints, variable analysis, and step functionality.
- Use console.log in your smart contracts
- Interactive console for direct contract communication.
- External script runner that executes scripts within a Truffle environment.
- Package management with NPM, using the ERC190 standard.
- Configurable build pipeline with support for tight integration.

https://trufflesuite.com/truffle/

Ganache: local test chain of truffle suite

a personal Ethereum blockchain which you can use to run tests, execute commands, and inspect state while controlling how the chain operates



CURRENT BLOCK 6 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MUIRGLACIER | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE TRUFFLE-SHUFFLE

MNEMONIC
candy maple cake sugar pudding cream honey rich smooth crumble sweet treat

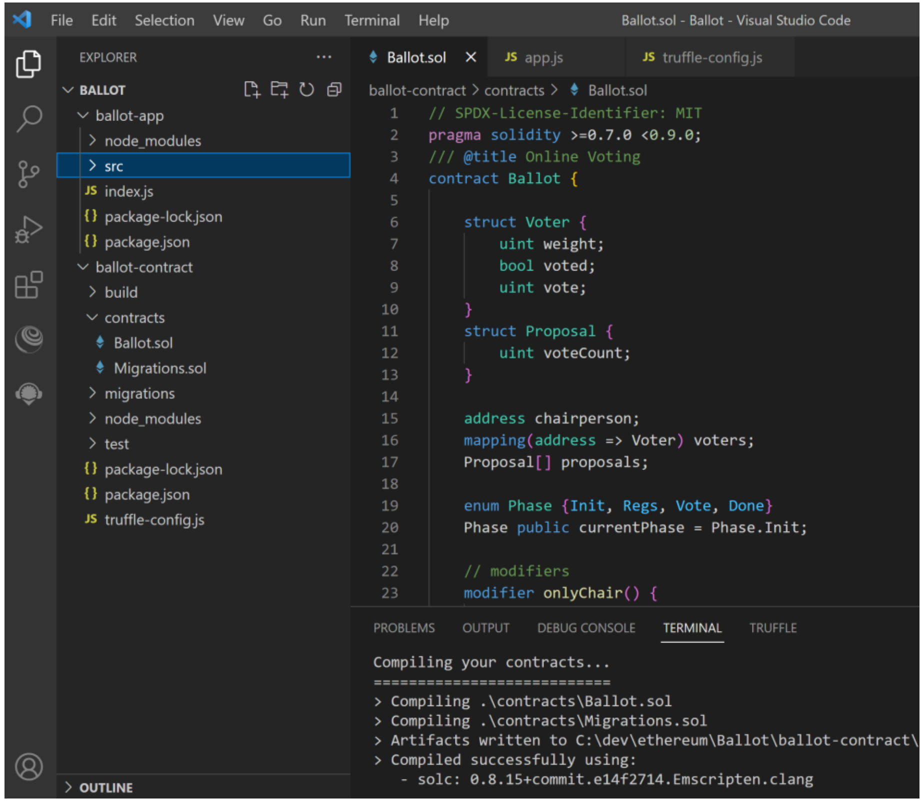
HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0x627306090abaB3A6e1400e9345bC60c78a8BEf57	99.46 ETH	32	0
0xf17f52151EbEF6C7334FAD080c5704D77216b732	100.00 ETH	0	1
0xC5fdf4076b8F3A5357c5E395ab970B5B54098Fef	100.00 ETH	0	2
0x821aEa9a577a9b44299B9c15c88cf3087F3b5544	100.00 ETH	0	3

https://trufflesuite.com/ganache/

VSCode (Visual Studio Code)

a popular code editor

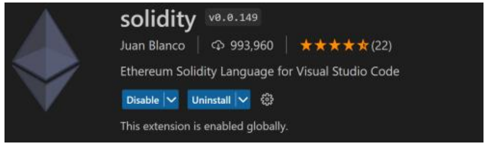


```solidity
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;
/// @title Online Voting
contract Ballot {

    struct Voter {
        uint weight;
        bool voted;
        uint vote;
    }
    struct Proposal {
        uint voteCount;
    }

    address chairperson;
    mapping(address => Voter) voters;
    Proposal[] proposals;

    enum Phase {Init, Regs, Vote, Done}
    Phase public currentPhase = Phase.Init;

    // modifiers
    modifier onlyChair() {
```

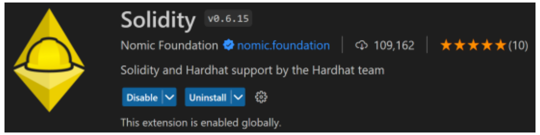
```
Compiling your contracts...
===========================
> Compiling .\contracts\Ballot.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to C:\dev\ethereum\Ballot\ballot-contract\
> Compiled successfully using:
   - solc: 0.8.15+commit.e14f2714.Emscripten.clang
```

https://code.visualstudio.com/

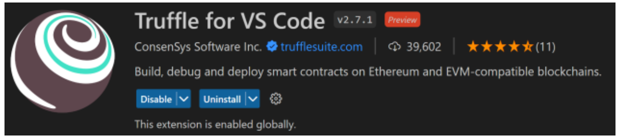
Useful extensions


solidity v0.0.149
Juan Blanco | 993,960 | ★★★★★(22)
Ethereum Solidity Language for Visual Studio Code

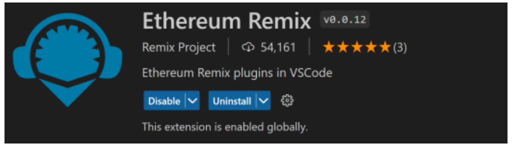
This extension is enabled globally.


Solidity v0.6.15
Nomic Foundation nomic.foundation | 109,162 | ★★★★★(10)
Solidity and Hardhat support by the Hardhat team

This extension is enabled globally.


Truffle for VS Code v2.7.1 Preview
ConsenSys Software Inc. trufflesuite.com | 39,602 | ★★★★★(11)
Build, debug and deploy smart contracts on Ethereum and EVM-compatible blockchains.

This extension is enabled globally.


Ethereum Remix v0.0.12
Remix Project | 54,161 | ★★★★★(3)
Ethereum Remix plugins in VSCode

This extension is enabled globally.

1) Initialize a template directory for contracts

1) install truffle

```
npm install –g truffle
truffle version
```

2) create folders for a project and contracts

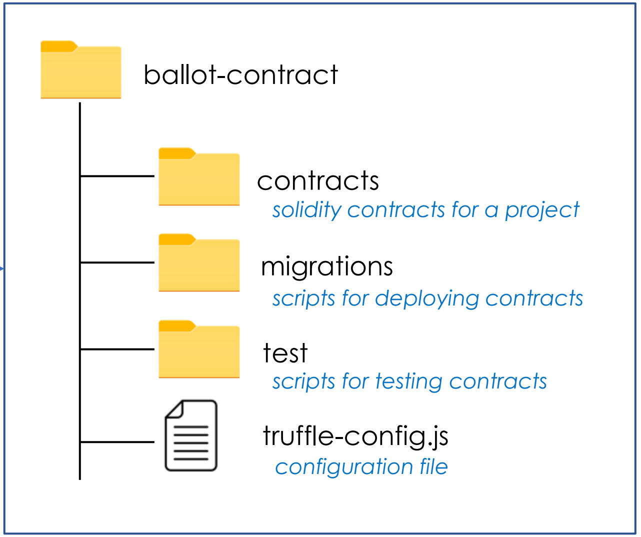
```
mkdir ballot
cd ballot
mkdir ballot-contract
```

3) create a template directory w/ the structure

```
cd ballot-contract
npm install
truffle init
```

4) write contracts and put them into the contracts folder

Generated by truffle init


ballot-contract
contracts
solidity contracts for a project
migrations
scripts for deploying contracts
test
scripts for testing contracts
truffle-config.js
configuration file

*** When you run 'truffle init', don't overwrite existing files**

```
? Overwrite contracts? No
migrations already exists in this directory...
? Overwrite migrations? No
test already exists in this directory...
? Overwrite test? No
truffle-config.js already exists in this directory...
? Overwrite truffle-config.js? No
```

2) Compile and deploy contracts

5) compile contracts

```
cd contracts
truffle compile
```

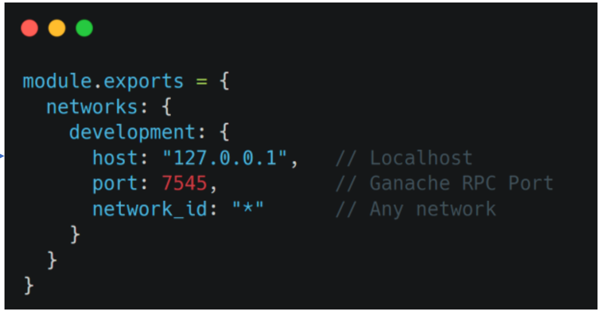
6) write config files for deployment
  6-1) modify truffle-config.js
  6-2) write migrations/2_deploy_contracts.js

7) run Ganache

8) deploy contracts

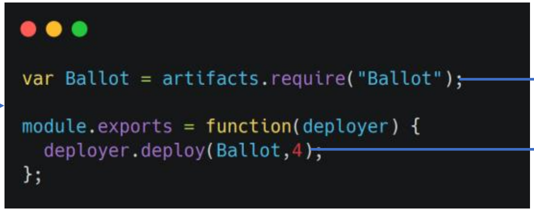
```
truffle migrate --reset
```

truffle-config.js



```
module.exports = {
  networks: {
    development: {
      host: "127.0.0.1",     // Localhost
      port: 7545,            // Ganache RPC Port
      network_id: "*"        // Any network
    }
  }
}
```

migrations/2_deploy_contracts.js



```
var Ballot = artifacts.require("Ballot");

module.exports = function(deployer) {
  deployer.deploy(Ballot,4);
};
```

contract name

numProposals

Systematic testing for smart contracts

Truffle supports an automated testing framework with testing scripts
1) in Javascript (Mocha testing framework & Chai for assertions), 2) in Solidity

- **beforeEach() -** the preconditions for other tests, specifying the code that will be executed before every test defined by it() and describe() test specifications. The beforeEach() function initializes the contract and establishes the base condition for the execution of a test command.

- **it() -** a standalone test of a function as an *independent test* or a unit test.

- **describe() -** This function is a composite test structure, and it specifies a group of related it() tests. Inside the test functions (it, describe, and so on), you'll also use a few other declarations:
    - **async()** - Allows for the asynchronous execution of functions, especially because transactions on a blockchain takes variable run times
    - **await()** - Waits for a callback from the function invoked using async() mode
    - **assert()** - Specifies the condition to assert; typically, it helps match the actual result of a statement execution with expected results. If the match fails, the assertion fails.

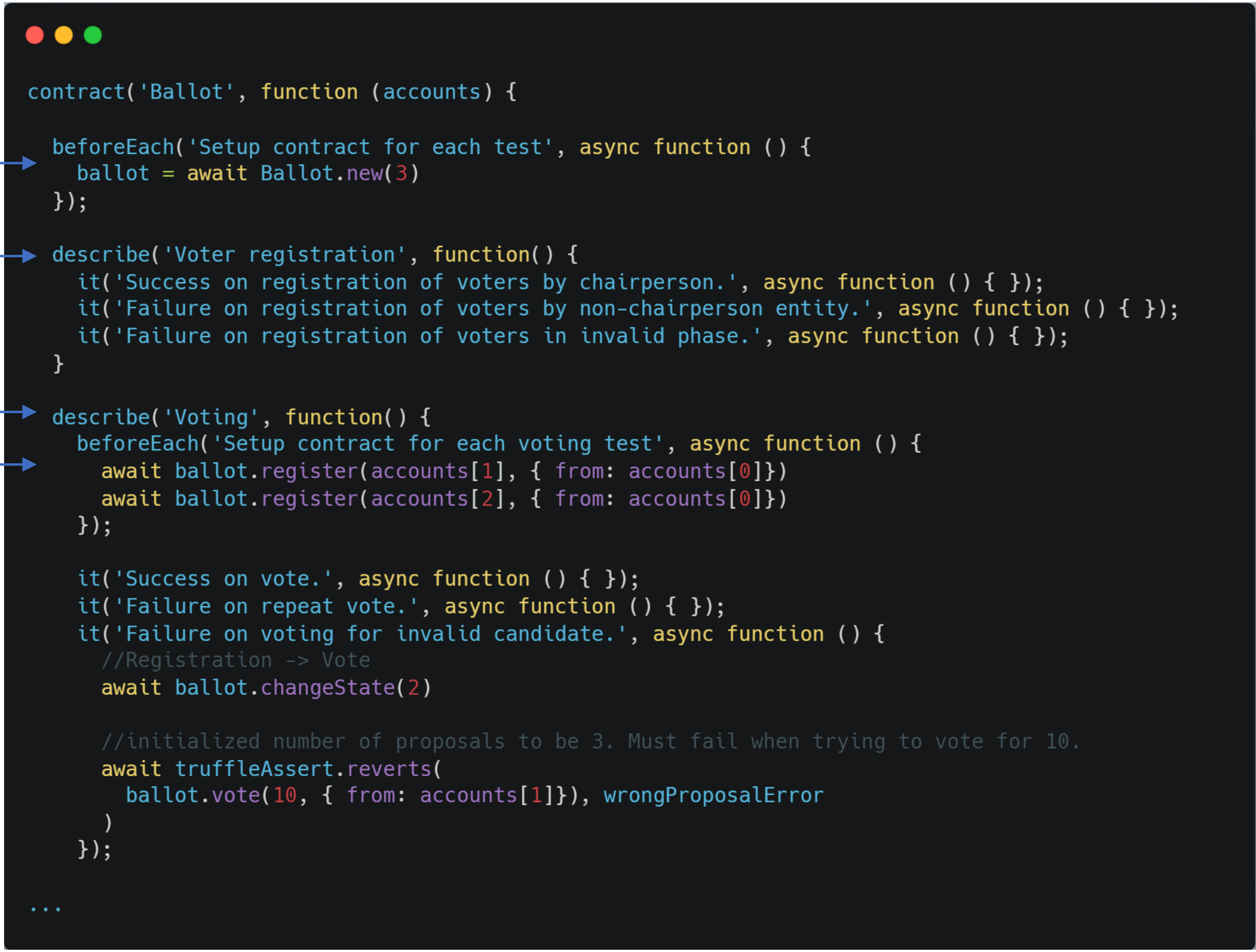
ballotTest.js

Initialize contracts for every unit test

Testing Regs phase

Testing Vote phase

Initialize Vote phase for every unit test(it()) within this describe()



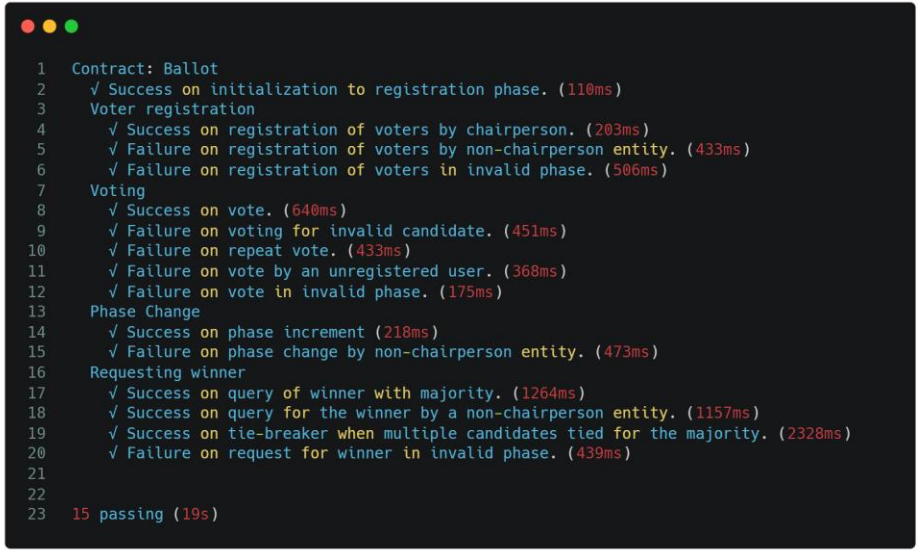
```
contract('Ballot', function (accounts) {

  beforeEach('Setup contract for each test', async function () {
    ballot = await Ballot.new(3)
  });

  describe('Voter registration', function() {
    it('Success on registration of voters by chairperson.', async function () { });
    it('Failure on registration of voters by non-chairperson entity.', async function () { });
    it('Failure on registration of voters in invalid phase.', async function () { });
  }

  describe('Voting', function() {
    beforeEach('Setup contract for each voting test', async function () {
      await ballot.register(accounts[1], { from: accounts[0]})
      await ballot.register(accounts[2], { from: accounts[0]})
    });

    it('Success on vote.', async function () { });
    it('Failure on repeat vote.', async function () { });
    it('Failure on voting for invalid candidate.', async function () {
      //Registration -> Vote
      await ballot.changeState(2)

      //initialized number of proposals to be 3. Must fail when trying to vote for 10.
      await truffleAssert.reverts(
        ballot.vote(10, { from: accounts[1]}), wrongProposalError
      )
    });

...
```

3) Test contracts w/ truffle

9) write ballotTest.js and put it into the test folder

10) test contracts

```
truffle test
```



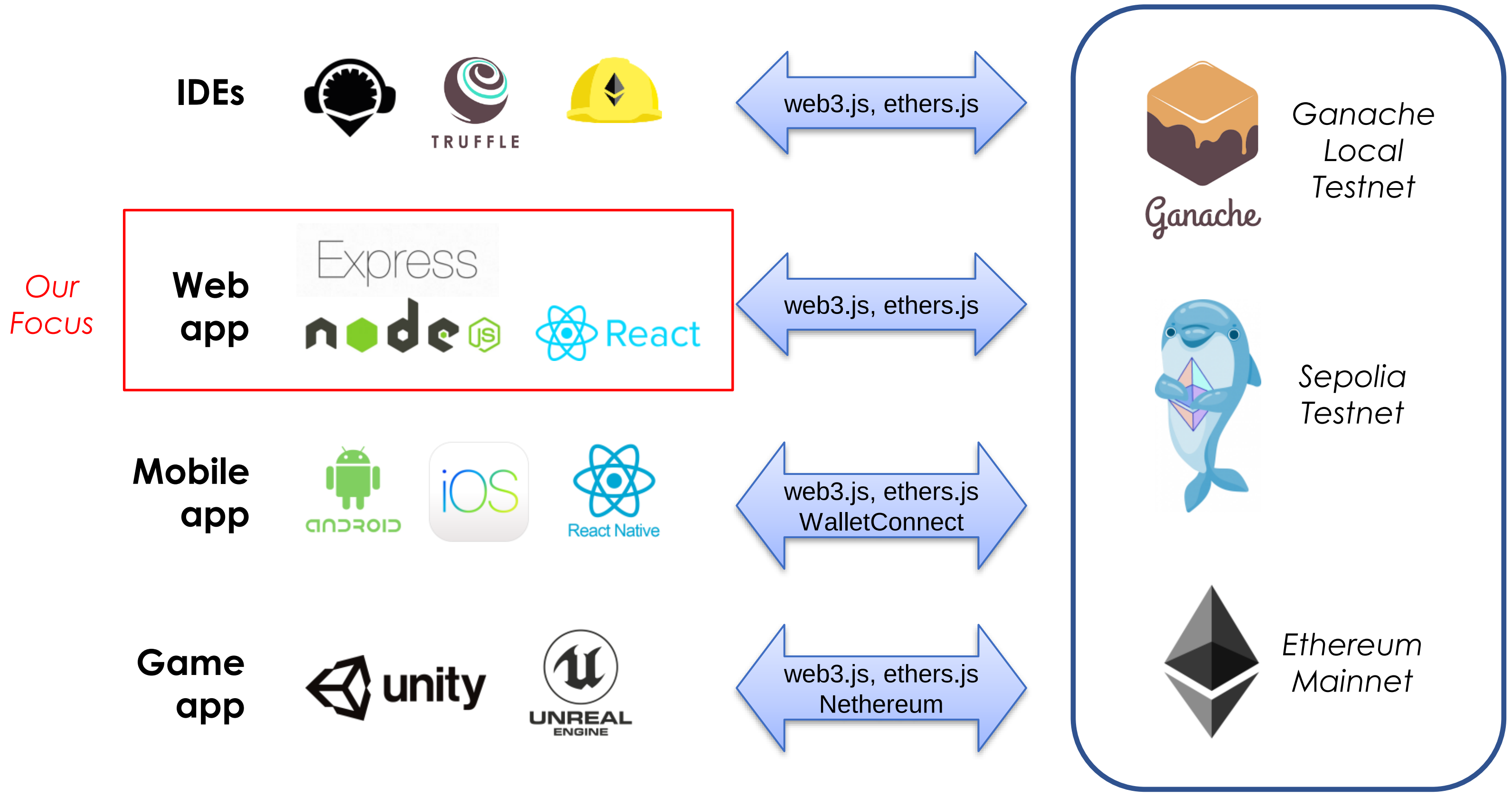
```
Contract: Ballot
  √ Success on initialization to registration phase. (110ms)
  Voter registration
    √ Success on registration of voters by chairperson. (203ms)
    √ Failure on registration of voters by non-chairperson entity. (433ms)
    √ Failure on registration of voters in invalid phase. (506ms)
  Voting
    √ Success on vote. (640ms)
    √ Failure on voting for invalid candidate. (451ms)
    √ Failure on repeat vote. (433ms)
    √ Failure on vote by an unregistered user. (368ms)
    √ Failure on vote in invalid phase. (175ms)
  Phase Change
    √ Success on phase increment (218ms)
    √ Failure on phase change by non-chairperson entity. (473ms)
  Requesting winner
    √ Success on query of winner with majority. (1264ms)
    √ Success on query for the winner by a non-chairperson entity. (1157ms)
    √ Success on tie-breaker when multiple candidates tied for the majority. (2328ms)
    √ Failure on request for winner in invalid phase. (439ms)


15 passing (19s)
```

Test results

Video-02
Deploying and testing contracts with Truffle

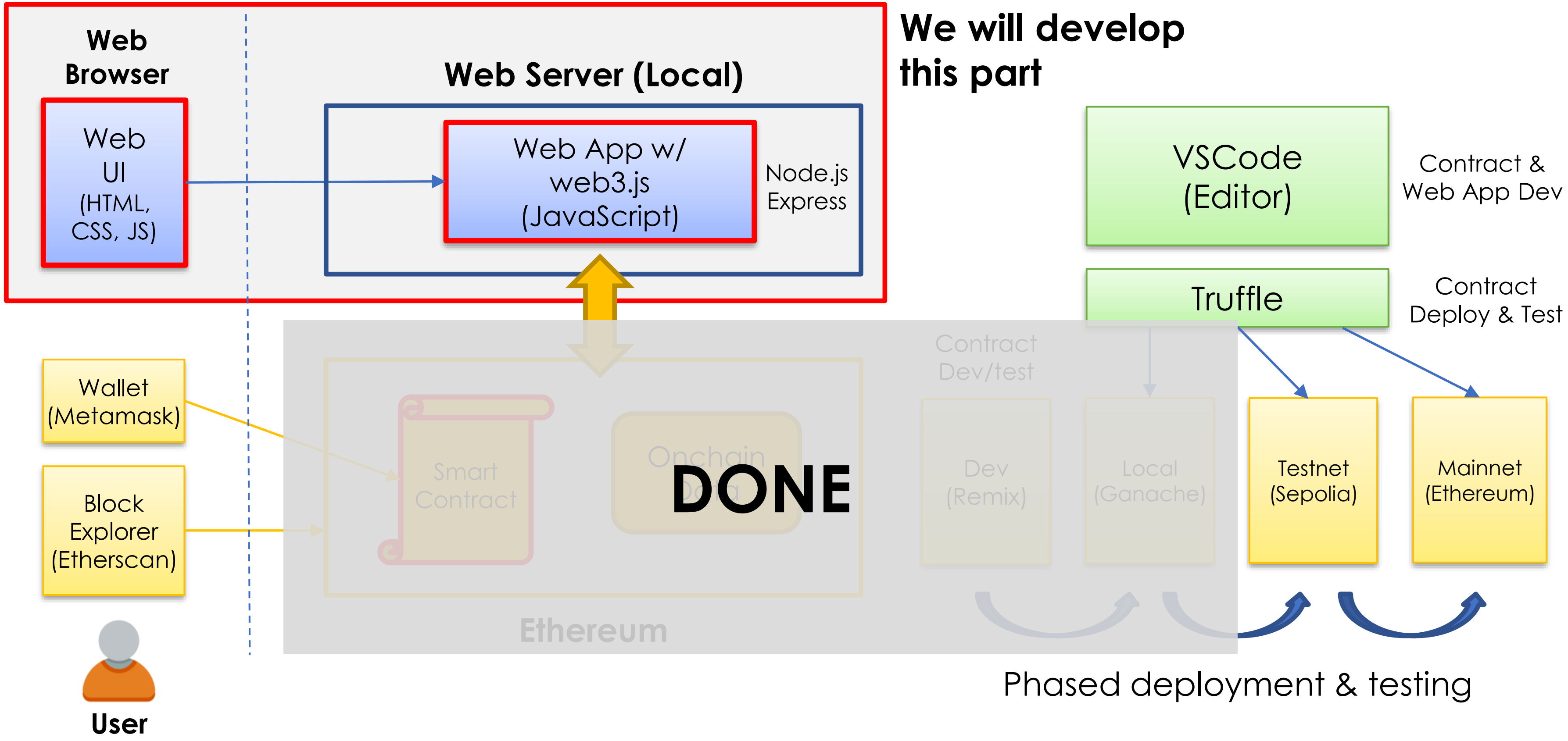
4. Develop Web App

How to call smart contracts in various app types

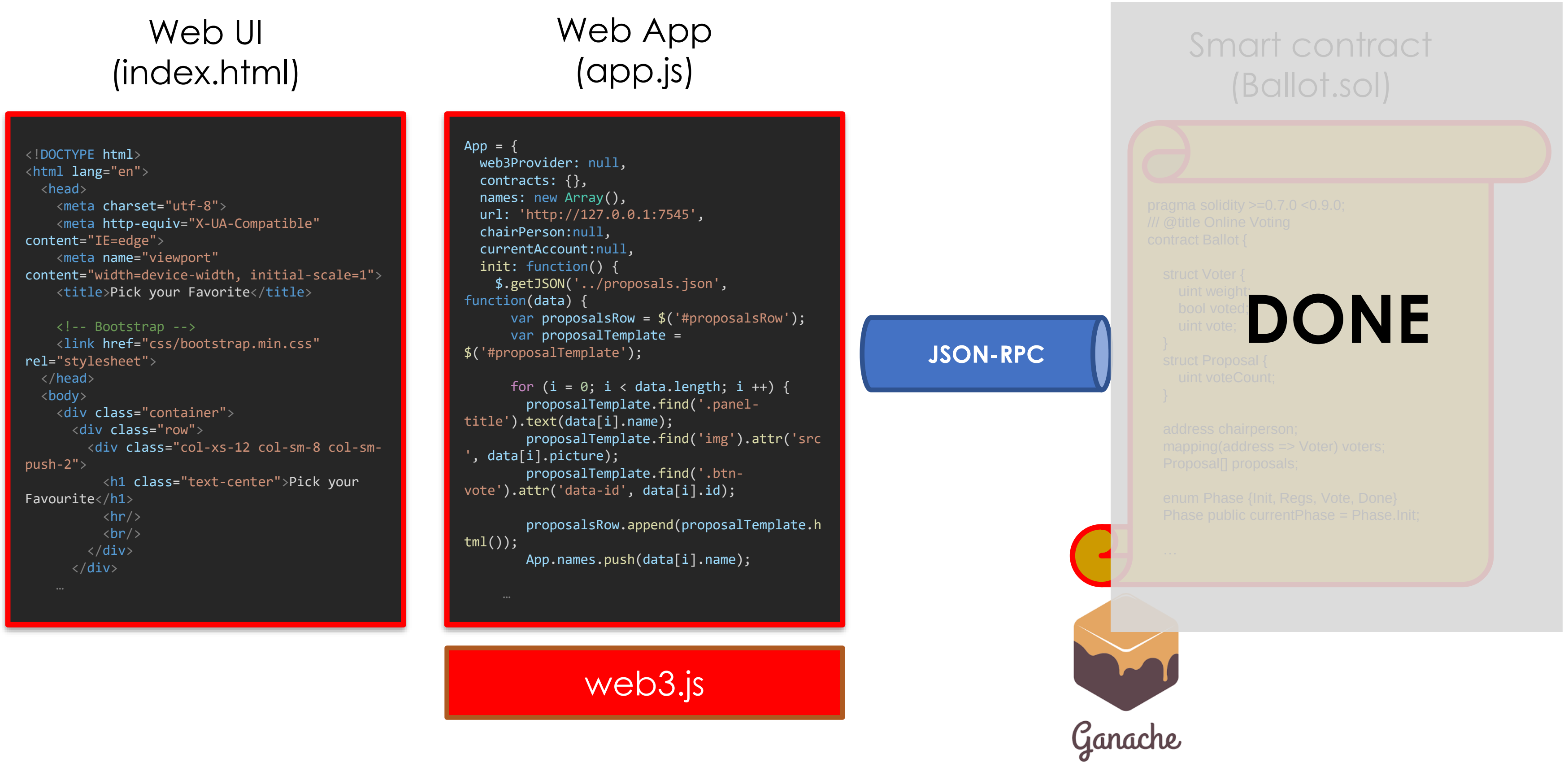


Dapp Development Environment

based on truffle & web3.js



What we develop in this lecture

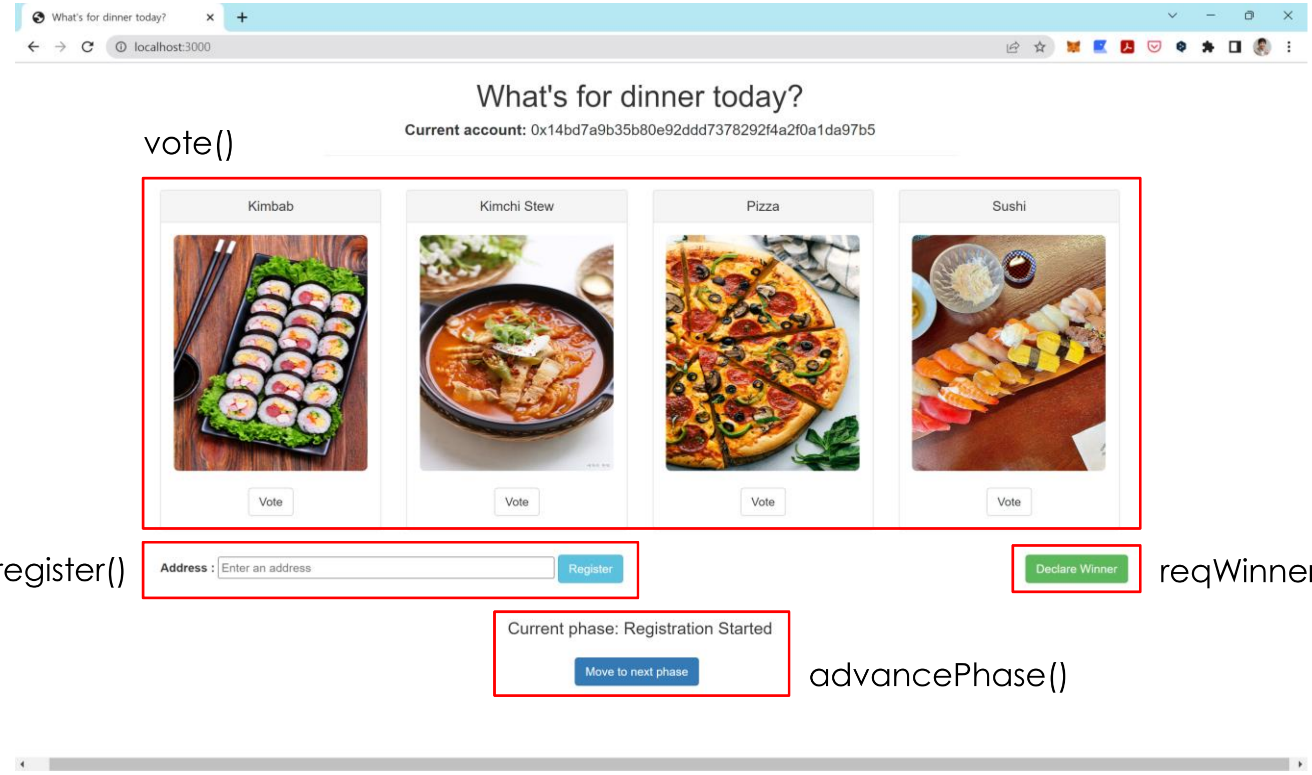


```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible"
content="IE=edge">
    <meta name="viewport"
content="width=device-width, initial-scale=1">
    <title>Pick your Favorite</title>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css"
rel="stylesheet">
  </head>
  <body>
    <div class="container">
      <div class="row">
        <div class="col-xs-12 col-sm-8 col-sm-
push-2">
          <h1 class="text-center">Pick your
Favourite</h1>
          <hr/>
          <br/>
        </div>
      </div>
    …
```

```javascript
App = {
  web3Provider: null,
  contracts: {},
  names: new Array(),
  url: 'http://127.0.0.1:7545',
  chairPerson:null,
  currentAccount:null,
  init: function() {
    $.getJSON('../proposals.json',
function(data) {
      var proposalsRow = $('#proposalsRow');
      var proposalTemplate =
$('#proposalTemplate');

      for (i = 0; i < data.length; i ++) {
        proposalTemplate.find('.panel-
title').text(data[i].name);
        proposalTemplate.find('img').attr('src
', data[i].picture);
        proposalTemplate.find('.btn-
vote').attr('data-id', data[i].id);

        proposalsRow.append(proposalTemplate.h
tml());
        App.names.push(data[i].name);

    …
```

```solidity
pragma solidity >=0.7.0 <0.9.0;
/// @title Online Voting
contract Ballot {

  struct Voter {
    uint weight;
    bool voted;
    uint vote;
  }
  struct Proposal {
    uint voteCount;
  }

  address chairperson;
  mapping(address => Voter) voters;
  Proposal[] proposals;

  enum Phase {Init, Regs, Vote, Done}
  Phase public currentPhase = Phase.Init;

  …
```

Web UI (frontend) for Online Voting



What's for dinner today?

Current account: 0x14bd7a9b35b80e92ddd7378292f4a2f0a1da97b5

Address : Enter an address

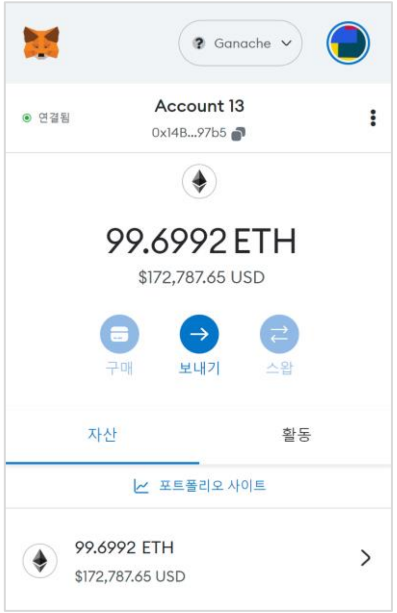
Current phase: Registration Started

vote()

register()

reqWinner()

advancePhase()



Develop Web App

1) create a template directory for a web app

```
cd ballot-app
npm init   // you don't need to run this
```

2) modify package.json

3) write express-based page (index.js)

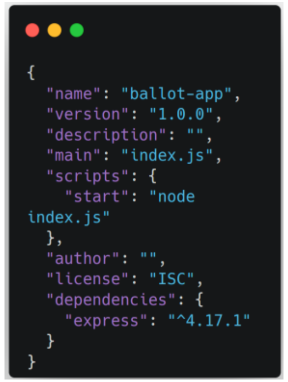
4) write web UI (index.html)

5) write web app (app.js)

6) install modules and start Node.js

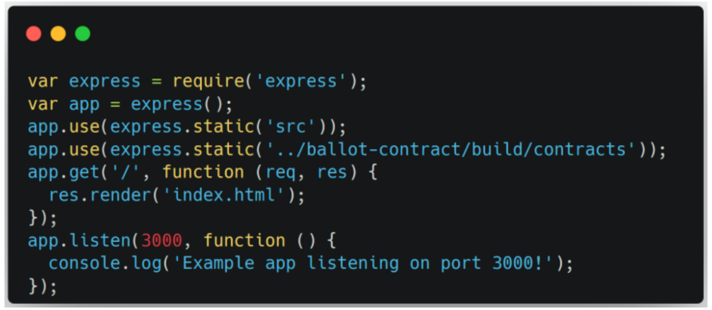
```
npm install
npm start
```

package.json



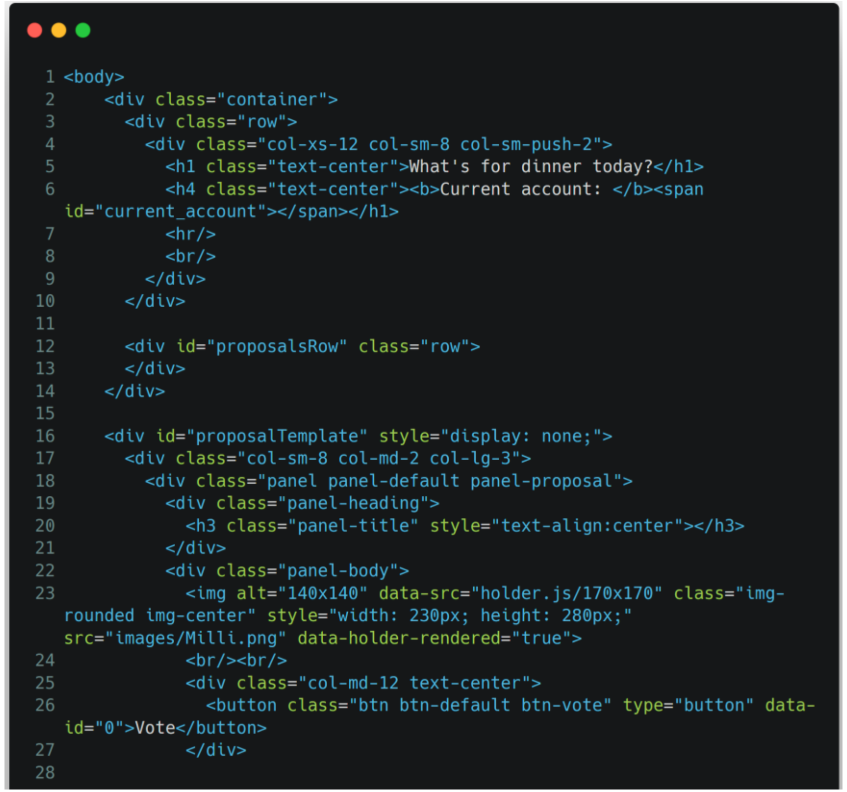
```
{
  "name": "ballot-app",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node
index.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

index.js

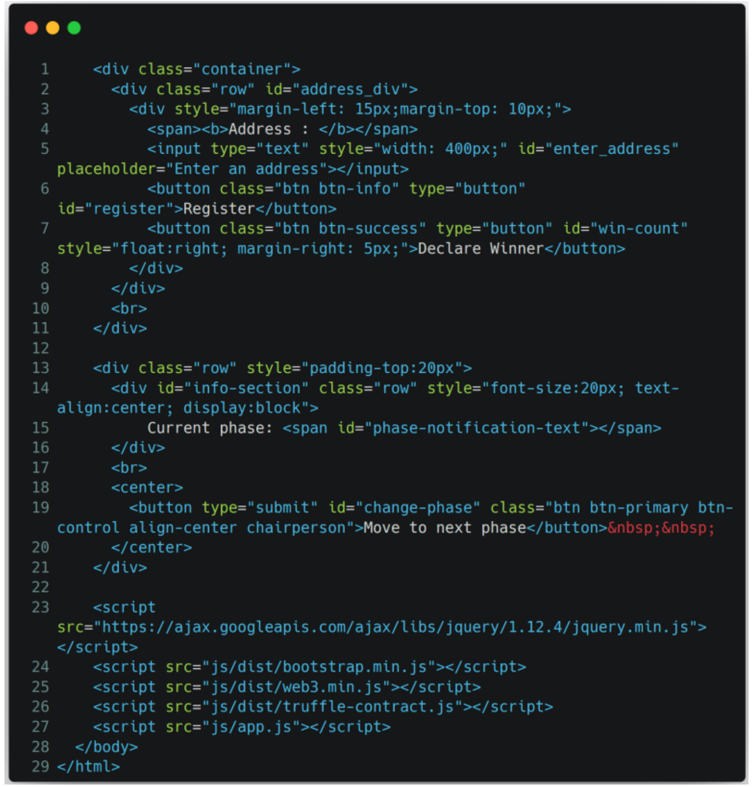


```
var express = require('express');
var app = express();
app.use(express.static('src'));
app.use(express.static('../ballot-contract/build/contracts'));
app.get('/', function (req, res) {
  res.render('index.html');
});
app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

Web UI (index.html)

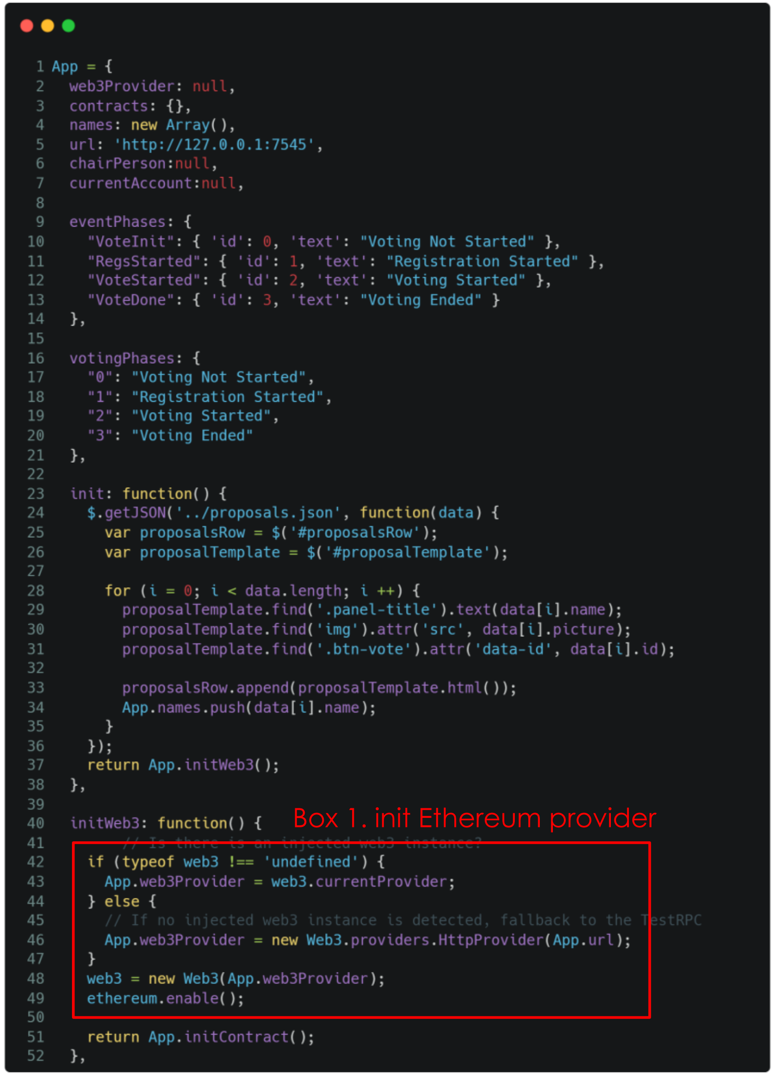


```html
<body>
  <div class="container">
   <div class="row">
    <div class="col-xs-12 col-sm-8 col-sm-push-2">
     <h1 class="text-center">What's for dinner today?</h1>
     <h4 class="text-center"><b>Current account: </b><span id="current_account"></span></h1>
     <hr/>
     <br/>
    </div>
   </div>

   <div id="proposalsRow" class="row">
   </div>
  </div>

  <div id="proposalTemplate" style="display: none;">
   <div class="col-sm-8 col-md-2 col-lg-3">
    <div class="panel panel-default panel-proposal">
     <div class="panel-heading">
      <h3 class="panel-title" style="text-align:center"></h3>
     </div>
     <div class="panel-body">
      <img alt="140x140" data-src="holder.js/170x170" class="img-rounded img-center" style="width: 230px; height: 280px;" src="images/Milli.png" data-holder-rendered="true">
      <br/><br/>
      <div class="col-md-12 text-center">
       <button class="btn btn-default btn-vote" type="button" data-id="0">Vote</button>
      </div>

```

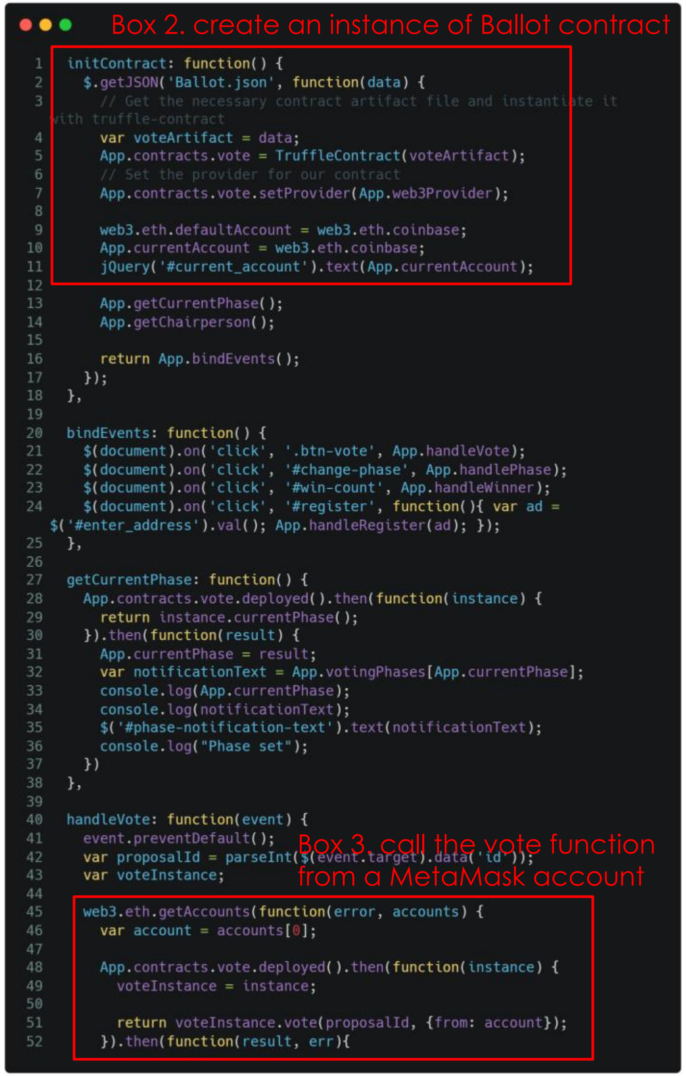


```html
  <div class="container">
   <div class="row" id="address_div">
    <div style="margin-left: 15px;margin-top: 10px;">
     <span><b>Address : </b></span>
     <input type="text" style="width: 400px;" id="enter_address" placeholder="Enter an address"></input>
     <button class="btn btn-info" type="button" id="register">Register</button>
     <button class="btn btn-success" type="button" id="win-count" style="float:right; margin-right: 5px;">Declare Winner</button>
    </div>
   </div>
   <br>
  </div>

  <div class="row" style="padding-top:20px">
   <div id="info-section" class="row" style="font-size:20px; text-align:center; display:block">
     Current phase: <span id="phase-notification-text"></span>
   </div>
   <br>
   <center>
    <button type="submit" id="change-phase" class="btn btn-primary btn-control align-center chairperson">Move to next phase</button>&nbsp;&nbsp;
   </center>
  </div>

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
  <script src="js/dist/bootstrap.min.js"></script>
  <script src="js/dist/web3.min.js"></script>
  <script src="js/dist/truffle-contract.js"></script>
  <script src="js/app.js"></script>
 </body>
</html>
```

Web App (app.js)



```javascript
App = {
  web3Provider: null,
  contracts: {},
  names: new Array(),
  url: 'http://127.0.0.1:7545',
  chairPerson:null,
  currentAccount:null,

  eventPhases: {
    "VoteInit": { 'id': 0, 'text': "Voting Not Started" },
    "RegsStarted": { 'id': 1, 'text': "Registration Started" },
    "VoteStarted": { 'id': 2, 'text': "Voting Started" },
    "VoteDone": { 'id': 3, 'text': "Voting Ended" }
  },

  votingPhases: {
    "0": "Voting Not Started",
    "1": "Registration Started",
    "2": "Voting Started",
    "3": "Voting Ended"
  },

  init: function() {
    $.getJSON('../proposals.json', function(data) {
      var proposalsRow = $('#proposalsRow');
      var proposalTemplate = $('#proposalTemplate');

      for (i = 0; i < data.length; i ++) {
        proposalTemplate.find('.panel-title').text(data[i].name);
        proposalTemplate.find('img').attr('src', data[i].picture);
        proposalTemplate.find('.btn-vote').attr('data-id', data[i].id);

        proposalsRow.append(proposalTemplate.html());
        App.names.push(data[i].name);
      }
    });
    return App.initWeb3();
  },

  initWeb3: function() {
        // Is there is an injected web3 instance?
    if (typeof web3 !== 'undefined') {
      App.web3Provider = web3.currentProvider;
    } else {
      // If no injected web3 instance is detected, fallback to the TestRPC
      App.web3Provider = new Web3.providers.HttpProvider(App.url);
    }
    web3 = new Web3(App.web3Provider);
    ethereum.enable();

    return App.initContract();
  },
```

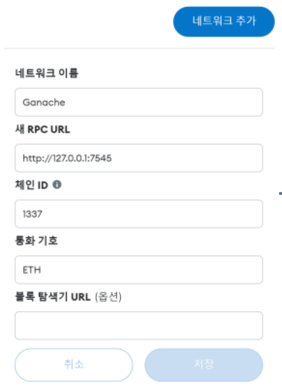

Box 2. create an instance of Ballot contract
Box 3. call the vote function from a MetaMask account

```javascript
  initContract: function() {
    $.getJSON('Ballot.json', function(data) {
      // Get the necessary contract artifact file and instantiate it
with truffle-contract
      var voteArtifact = data;
      App.contracts.vote = TruffleContract(voteArtifact);
      // Set the provider for our contract
      App.contracts.vote.setProvider(App.web3Provider);

      web3.eth.defaultAccount = web3.eth.coinbase;
      App.currentAccount = web3.eth.coinbase;
      jQuery('#current_account').text(App.currentAccount);

      App.getCurrentPhase();
      App.getChairperson();

      return App.bindEvents();
    });
  },

  bindEvents: function() {
    $(document).on('click', '.btn-vote', App.handleVote);
    $(document).on('click', '#change-phase', App.handlePhase);
    $(document).on('click', '#win-count', App.handleWinner);
    $(document).on('click', '#register', function(){ var ad =
$('#enter_address').val(); App.handleRegister(ad); });
  },

  getCurrentPhase: function() {
    App.contracts.vote.deployed().then(function(instance) {
      return instance.currentPhase();
    }).then(function(result) {
      App.currentPhase = result;
      var notificationText = App.votingPhases[App.currentPhase];
      console.log(App.currentPhase);
      console.log(notificationText);
      $('#phase-notification-text').text(notificationText);
      console.log("Phase set");
    })
  },

  handleVote: function(event) {
    event.preventDefault();
    var proposalId = parseInt($(event.target).data('id'));
    var voteInstance;

    web3.eth.getAccounts(function(error, accounts) {
      var account = accounts[0];

      App.contracts.vote.deployed().then(function(instance) {
        voteInstance = instance;

        return voteInstance.vote(proposalId, {from: account});
      }).then(function(result, err){
```

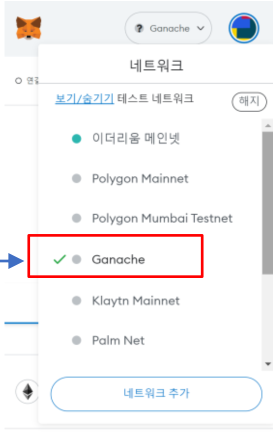
5. Deploy & test all (Local)

Setting up test accounts in Metamask

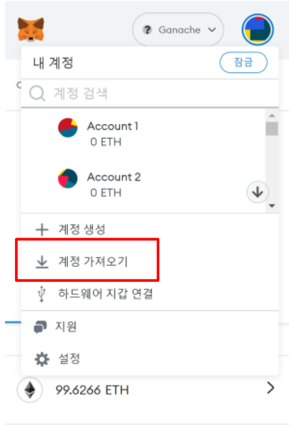
1) add local network (Ganache)

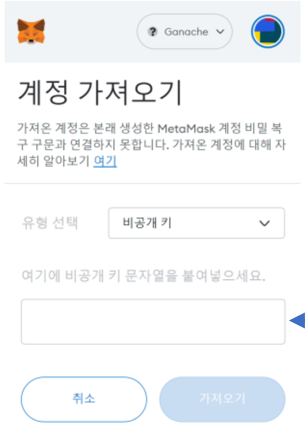


네트워크 이름: Ganache
새 RPC URL: http://127.0.0.1:7545
체인 ID: 1337
통화 기호: ETH
블록 탐색기 URL (옵션):



2) import accounts to Metamask



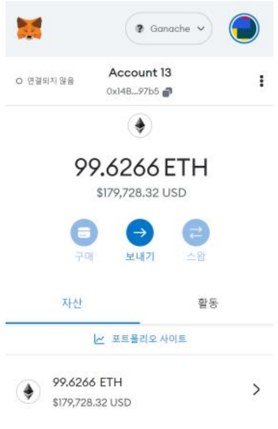


계정 가져오기

가져온 계정은 본래 생성한 MetaMask 계정 비밀 복구 구문과 연결하지 못합니다. 가져온 계정에 대해 자세히 알아보기 여기

유형 선택: 비공개 키

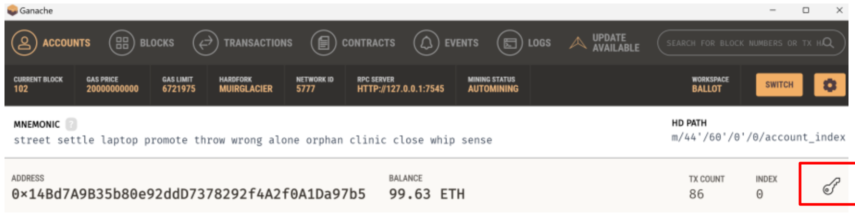
여기에 비공개 키 문자열을 붙여넣으세요.



Account 13
0x14B...97b5

99.6266 ETH
$179,728.32 USD

99.6266 ETH
$179,728.32 USD

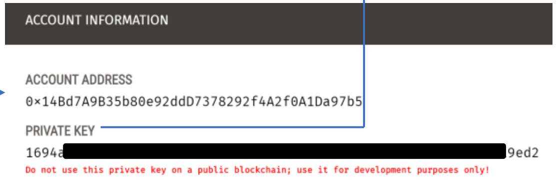


CURRENT BLOCK 102 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MUIRGLACIER | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE BALLOT

MNEMONIC
street settle laptop promote throw wrong alone orphan clinic close whip sense

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

ADDRESS	BALANCE	TX COUNT	INDEX
0×14Bd7A9B35b80e92ddD7378292f4A2f0A1Da97b5	99.63 ETH	86	0


ACCOUNT INFORMATION

ACCOUNT ADDRESS
0×14Bd7A9B35b80e92ddD7378292f4A2f0A1Da97b5

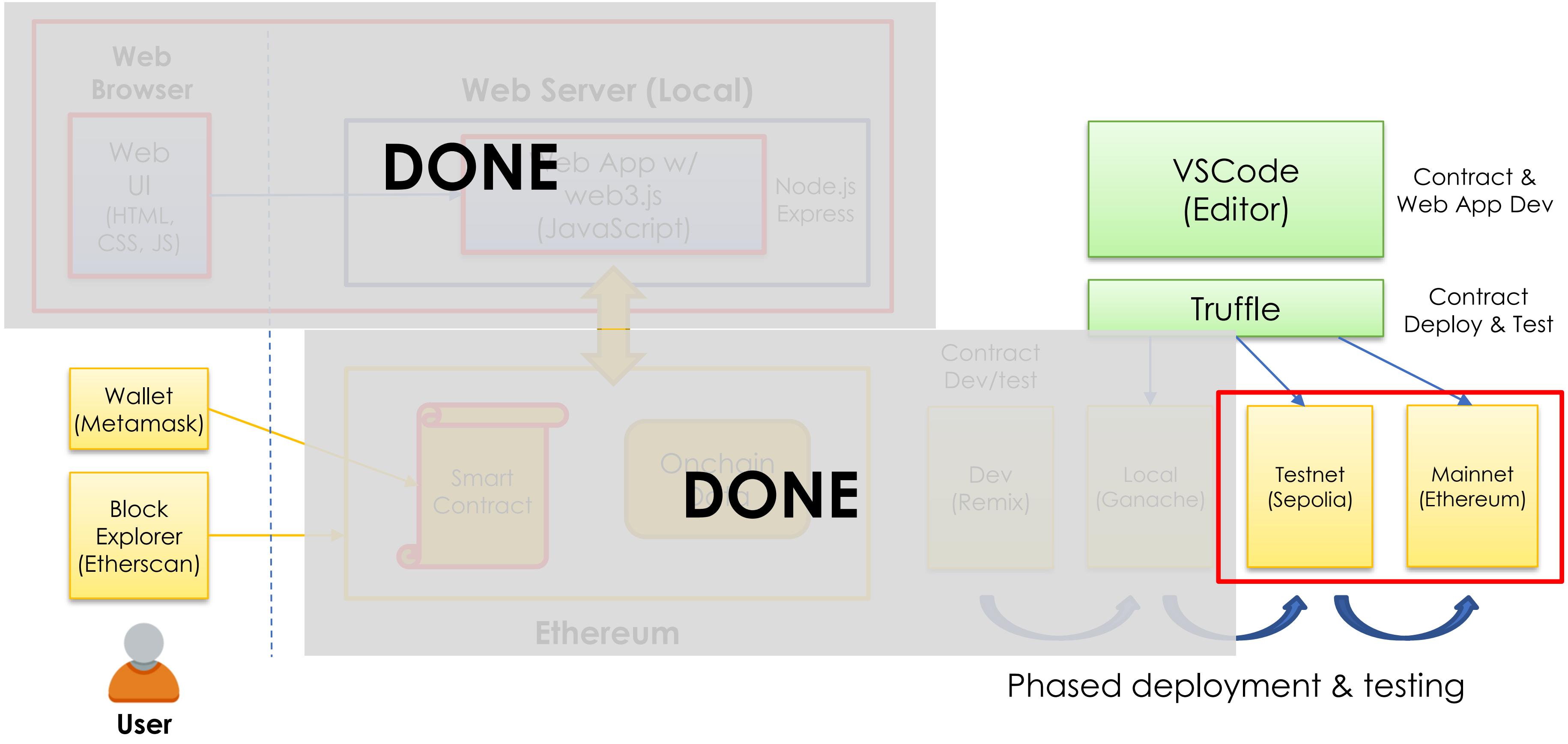
PRIVATE KEY
1694a[redacted]9ed2
Do not use this private key on a public blockchain; use it for development purposes only!

Video-03
Deploying & testing integrations locally

6. Deploy & test all (Testnet)

Dapp Development Environment

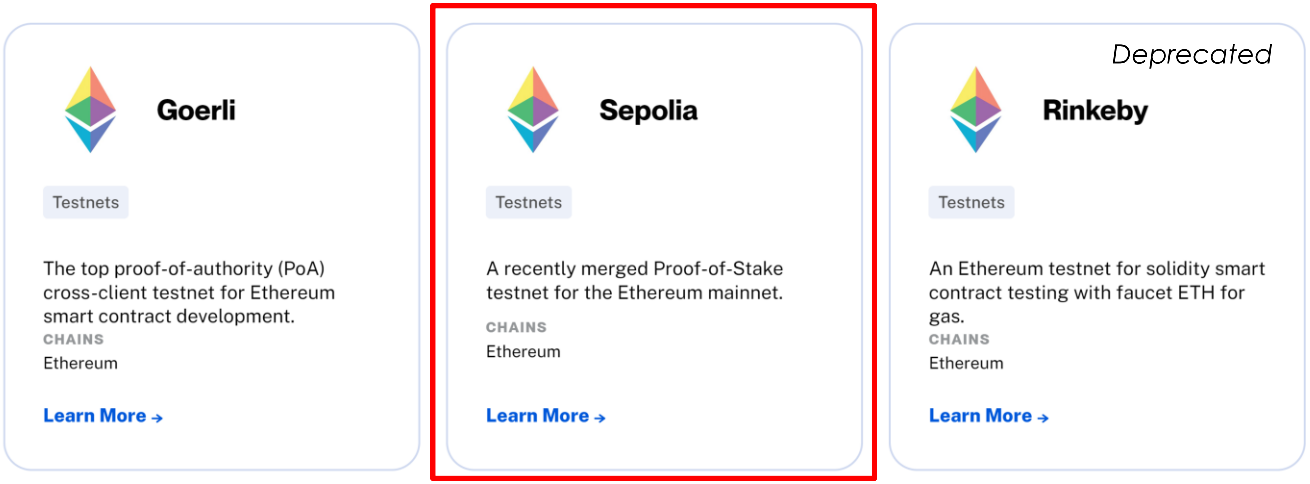
based on truffle & web3.js



Ethereum Testnets

- *Mimic a Mainnet but exist on a separate ledger*
- *Help developers test their applications and smart contracts in a risk-free way*

Our choice



Goerli

Testnets

The top proof-of-authority (PoA) cross-client testnet for Ethereum smart contract development.

CHAINS

Ethereum

Learn More →

Sepolia

Testnets

A recently merged Proof-of-Stake testnet for the Ethereum mainnet.

CHAINS

Ethereum

Learn More →

Deprecated

Rinkeby

Testnets

An Ethereum testnet for solidity smart contract testing with faucet ETH for gas.

CHAINS

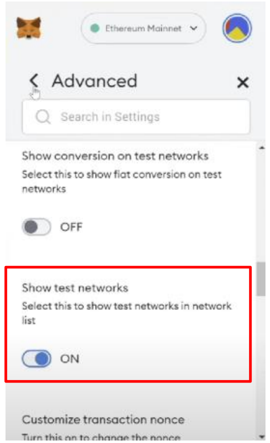
Ethereum

Learn More →

https://www.alchemy.com/list-of/testnets-on-ethereum
https://www.alchemy.com/overviews/goerli-vs-sepolia
https://ethereum.org/en/developers/docs/networks/

1. Add Sepolia to Metamask

1) add Sepolia testnet



Ethereum Mainnet

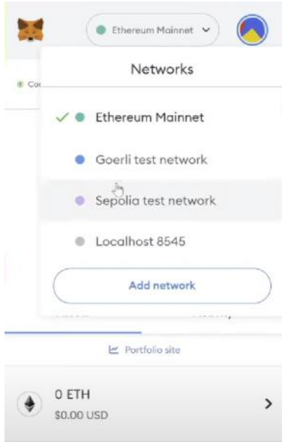
Advanced

Search in Settings

Show conversion on test networks
Select this to show fiat conversion on test networks
OFF

Show test networks
Select this to show test networks in network list
ON

Customize transaction nonce
Turn this on to change the nonce



Ethereum Mainnet

Networks

- ✓ Ethereum Mainnet
- Goerli test network
- Sepolia test network
- Localhost 8545

Add network

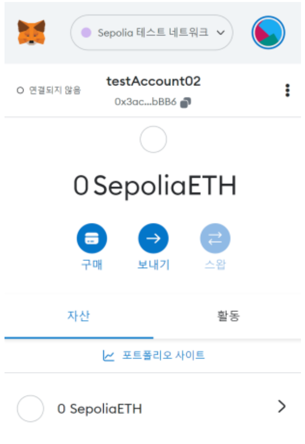
Portfolio site

0 ETH
$0.00 USD

If you want to add it manually,

- **Network Name** - Sepolia Test Netwok
- **RPC URL** - [get URL from RPC node proviers]
- **Chain ID** - 11155111
- **Currency Symbol** - SepoliaETH
- **Block Explorer URL** - https://sepolia.etherscan.io/

2) get free SepoloaETH for gas fee



Sepolia 테스트 네트워크

연결되지 않음

testAccount02
0x3ac...bBB6

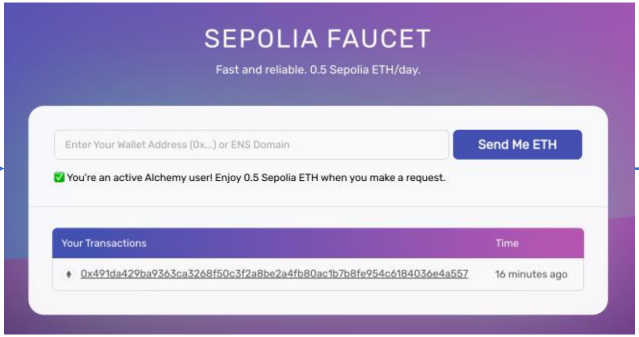
0 SepoliaETH

구매 보내기 스왑

자산 활동

포트폴리오 사이트

0 SepoliaETH



SEPOLIA FAUCET

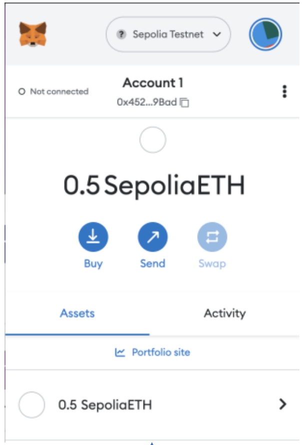
Fast and reliable. 0.5 Sepolia ETH/day.

Enter Your Wallet Address (0x...) or ENS Domain

Send Me ETH

✅ You're an active Alchemy user! Enjoy 0.5 Sepolia ETH when you make a request.

Your Transactions	Time
0x491da429ba9363ca3268f50c3f2a8be2a4fb80ac1b7b8fe954c6184036e4a557	16 minutes ago



Sepolia Testnet

Not connected

Account 1
0x452...9Bad

0.5 SepoliaETH

Buy Send Swap

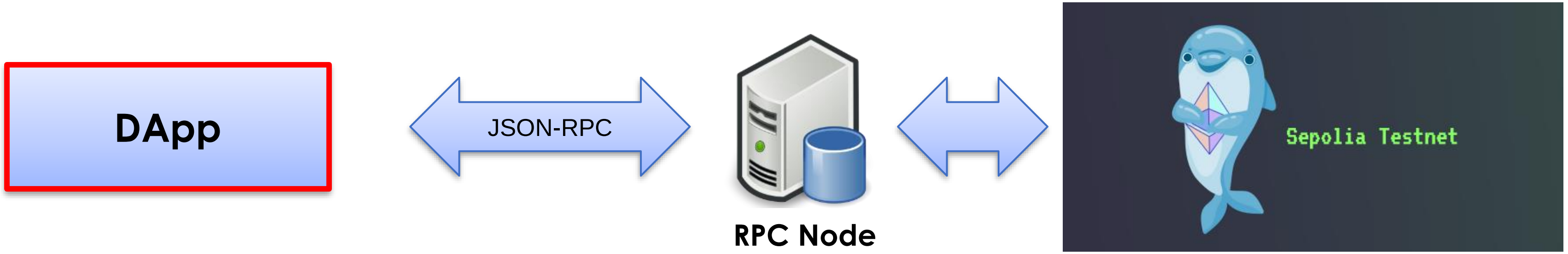
Assets Activity

Portfolio site

0.5 SepoliaETH

Faucets
https://sepoliafaucet.com/
https://www.infura.io/faucet

2. Prepare an RPC node



- Connect dapps to the blockchain through RPC (Remote Procedure Call)
- A computer running blockchain client software

Options to access an RPC node

1. Use an RPC node provider
2. Use a public RPC node
3. Run your own RPC node

https://www.alchemy.com/overviews/rpc-node

alchemy

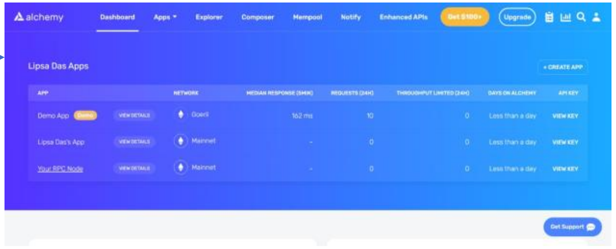
https://www.alchemy.com/

INFURA

https://www.infura.io/

All That Node

https://www.allthatnode.com/



Lipsa Das Apps

APP		NETWORK	MEDIAN RESPONSE (5MIN)	REQUESTS (24H)	THROUGHPUT LIMITED (24H)	DAYS ON ALCHEMY	API KEY
Demo App Demo	VIEW DETAILS	Goerli	162 ms	10	0	Less than a day	VIEW KEY
Lipsa Das's App	VIEW DETAILS	Mainnet	-	0	0	Less than a day	VIEW KEY
Your RPC Node	VIEW DETAILS	Mainnet	-	0	0	Less than a day	VIEW KEY

3. Deploy contracts to Sepolia

1) install required packages

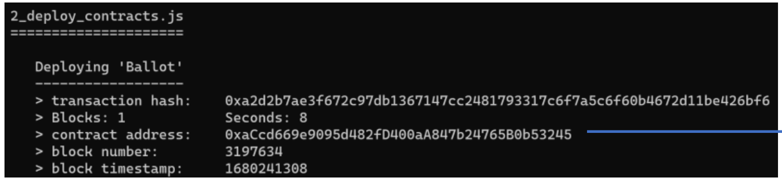
```
npm install @truffle/hdwallet-provider dotenv
```

2) write .env file to ballot-contract folder
PRIVATE_KEY=your_private_key
ALCHEMY_API_KEY=your_alchemy_api_key

3) add Sepolia conf to truffle-config.js

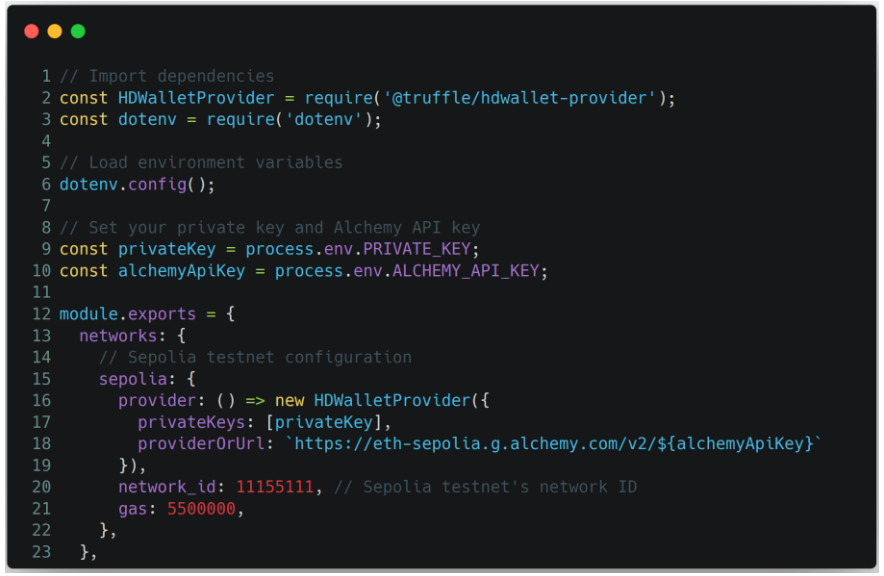
4) deploy contracts to Sepolia

```
truffle migrate --network sepolia
```



```
2_deploy_contracts.js
=====================

   Deploying 'Ballot'
   ------------------
   > transaction hash:    0xa2d2b7ae3f672c97db1367147cc2481793317c6f7a5c6f60b4672d11be426bf6
   > Blocks: 1            Seconds: 8
   > contract address:    0xaCcd669e9095d482fD400aA847b24765B0b53245
   > block number:        3197634
   > block timestamp:     1680241308
```

truffle-config.js

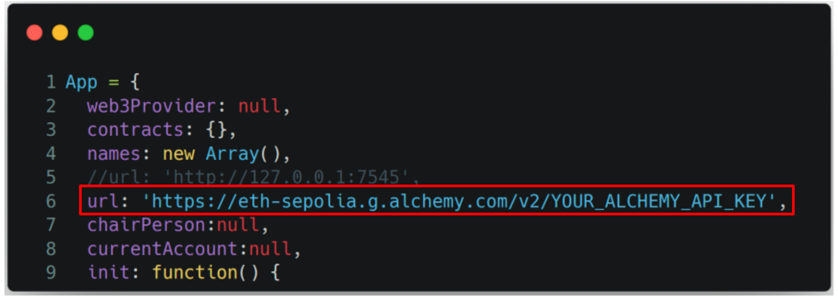


```
// Import dependencies
const HDWalletProvider = require('@truffle/hdwallet-provider');
const dotenv = require('dotenv');

// Load environment variables
dotenv.config();

// Set your private key and Alchemy API key
const privateKey = process.env.PRIVATE_KEY;
const alchemyApiKey = process.env.ALCHEMY_API_KEY;

module.exports = {
  networks: {
    // Sepolia testnet configuration
    sepolia: {
      provider: () => new HDWalletProvider({
        privateKeys: [privateKey],
        providerOrUrl: `https://eth-sepolia.g.alchemy.com/v2/${alchemyApiKey}`
      }),
      network_id: 11155111, // Sepolia testnet's network ID
      gas: 5500000,
    },
  },
```

Check the contract in Etherscan for Sepolia
https://sepolia.etherscan.io/

4. Run web app with contracts in Sepolia

1) modify app.js with an RPC node



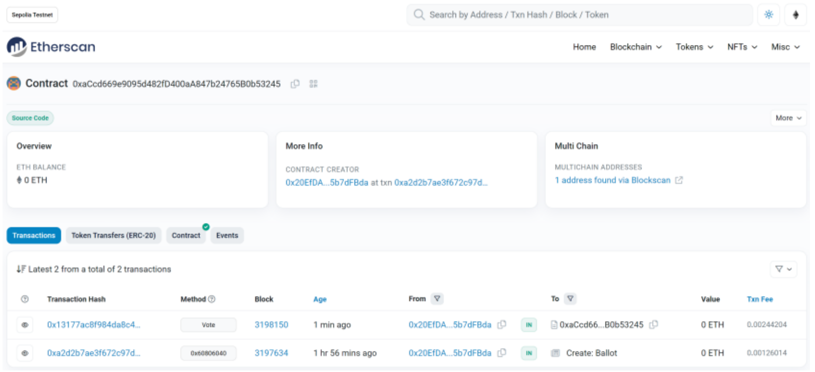
```
App = {
  web3Provider: null,
  contracts: {},
  names: new Array(),
  //url: 'http://127.0.0.1:7545',
  url: 'https://eth-sepolia.g.alchemy.com/v2/YOUR_ALCHEMY_API_KEY',
  chairPerson:null,
  currentAccount:null,
  init: function() {
```

2) run node.js

```
npm start
```

3) vote in the web page

4) check the tx on Etherscan



Contract 0xaCcd669e9095d482fD400aA847b24765B0b53245

Overview

ETH BALANCE
0 ETH

More Info

CONTRACT CREATOR
0x20EfDA...5b7dFBda at txn 0xa2d2b7ae3f672c97d...

Multi Chain

MULTICHAIN ADDRESSES
1 address found via Blockscan

Latest 2 from a total of 2 transactions

	Transaction Hash	Method	Block	Age	From		To	Value	Txn Fee
	0x13177ac8f984da8c4...	Vote	3198150	1 min ago	0x20EfDA...5b7dFBda	IN	0xaCcd66...B0b53245	0 ETH	0.00244204
	0xa2d2b7ae3f672c97d...	0x60806040	3197634	1 hr 56 mins ago	0x20EfDA...5b7dFBda	IN	Create: Ballot	0 ETH	0.00126014

Video-04
Deploying & testing integrations on Sepolia testnet

7. Deploy & test all (Mainnet)

It’s exactly the same as the testnet process, except for the RPC node and accounts

		alchemy	Etherscan
Sepolia Testnet	Sepolia account SepoliaETH	https://eth-sepolia.g.alchemy.com/v2/ YOUR_ALCHEMY_API_KEY	https://etherscan.io/
Ethereum Mainnet	Ethereum account ETH	https://eth-mainnet.g.alchemy.com/v2/ YOUR_ALCHEMY_API_KEY	https://sepolia.etherscan.io/

Wrap-up

We Learned

The entire process of building Dapp with Online voting Dapp

1. Design
2. Develop smart contracts with Remix
3. Deploy & test smart contracts (Local)
4. Develop a web app
5. Deploy & test all (Local)
6. Deploy & test all (Testnet)
7. Deploy & test all (Mainnet)

Note.

Smart contract development should be **a rigorous process**, because smart contracts are closely tied to financial assets, and a small mistake may lead to big losses and disaster.