

Be as proud of Sogang as Sogang is proud of you

스마트컨트랙트 Dapp




서강대학교
SOGANG UNIVERSITY

- 전자투표 Dapp
 - Problem Statement
 - 스마트컨트랙트 개발

■ 웹UI


Pick your Favourite

Milli




Vote

Murphy




Vote

Radar



Vote

Riley



Vote

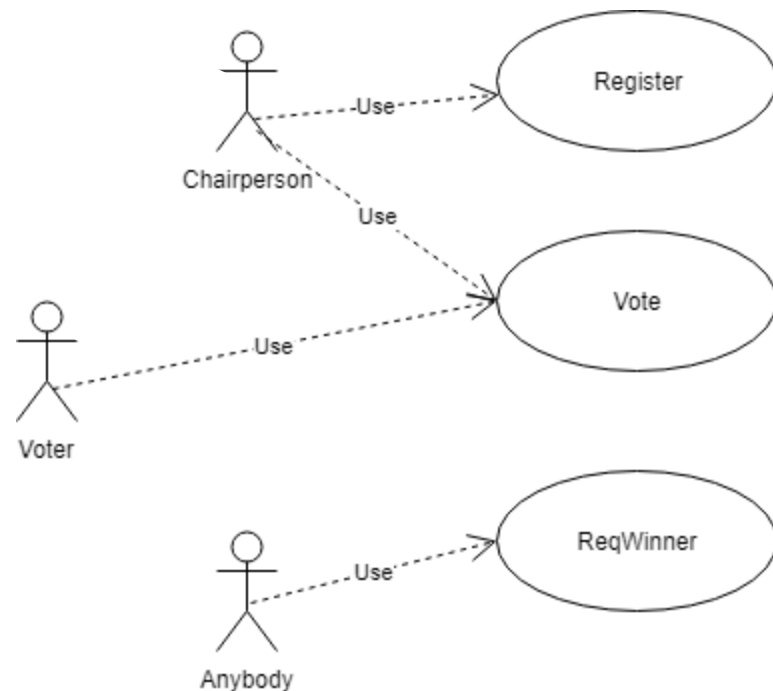
Address : ▼

[Register](#)

[Declare Winner](#)

■ Problem Statement

- 다수의 제안 가운데 하나에만 투표한다
- 의장은 투표할 수 있는 사람을 등록하고(한 계정은 한번만 등록)
- 등록된 사람만이 제안된 선택지 중의 하나에 오직 한번만 투표한다
- 의장의 표는 가중치를 두어 두 표로 계산한다
- 누구나 투표 결과를 확인할 수 있다



■ 데이터

```
struct Voter {  
    uint weight;  
    bool voted;  
    uint vote;  
}
```

투표자 상세 정보
(가중치,
투표 여부,
투표값)

```
struct Proposal {  
    uint voteCount;  
}
```

제안 상세 정보(투표수)

```
address chairperson;
```

```
mapping(address => Voter) public voters;
```

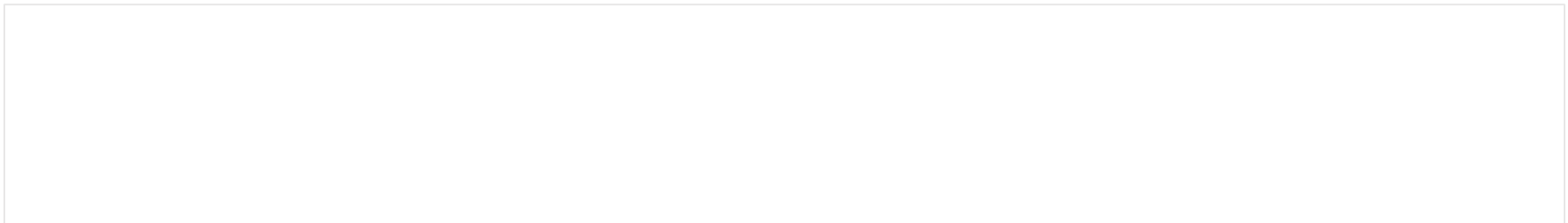
투표자 주소를 투표자
상세 정보로 매핑

```
Proposal[] public proposals;
```

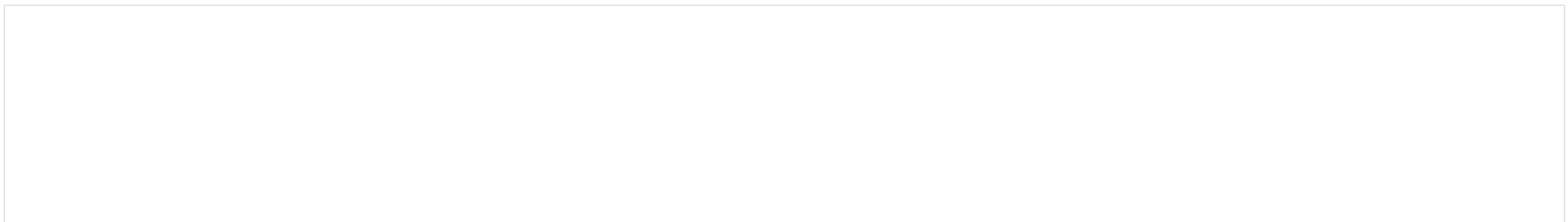
제안들을 담는 배열

- **modifier**

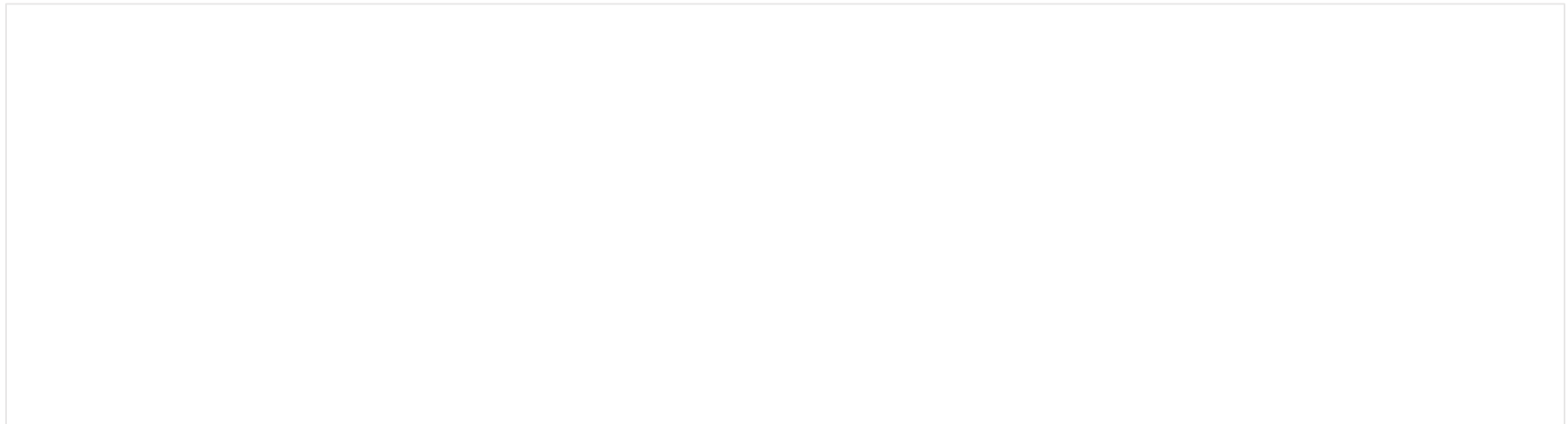
- onlyChair: 의장만 호출하도록 제한



- validVoter: 등록된 투표자만 호출하도록 제한
 - voters 매핑의 weight 정보로 확인

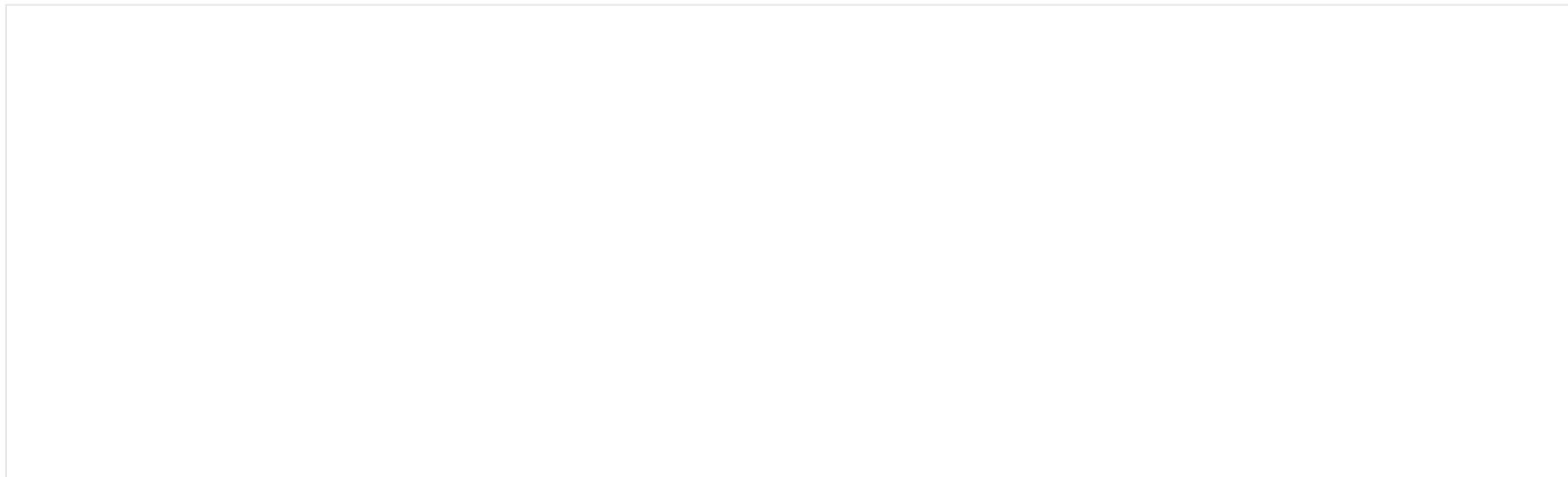


- 생성자
 - 매개변수: numProposals(제안 즉 후보자의 개수)
 - 제안 개수만큼 proposals 배열 초기화



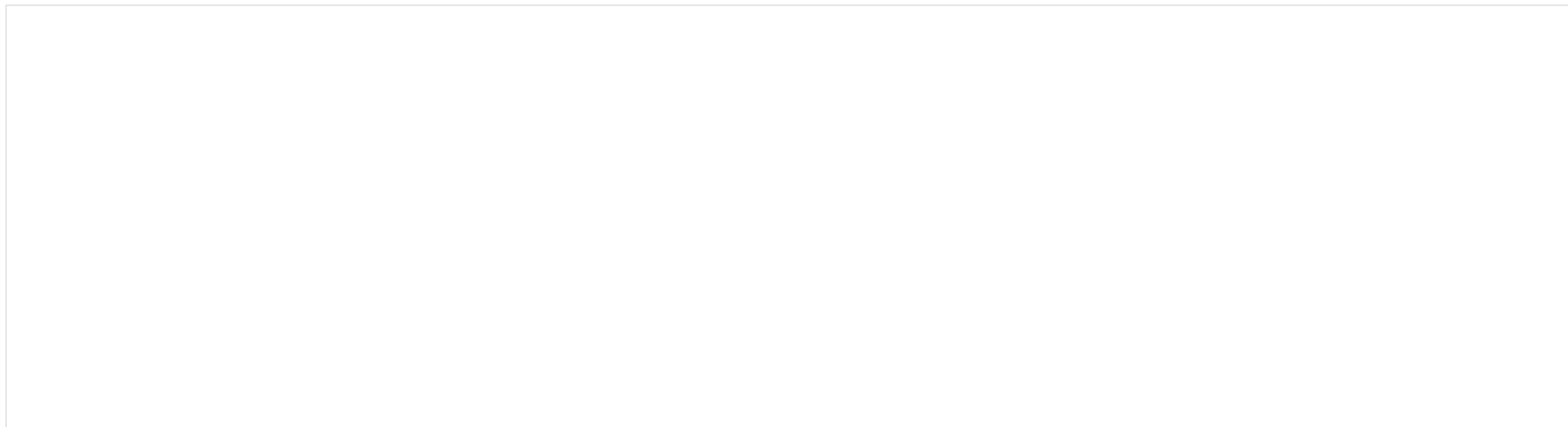
■ 함수

- register(매개변수: 등록할 투표자 주소)
 - chairperson만 실행 가능하게 제한
 - chairperson은 가중치 2로 설정, 나머지는 1
 - voters 매핑에 투표자 정보 추가



■ 함수

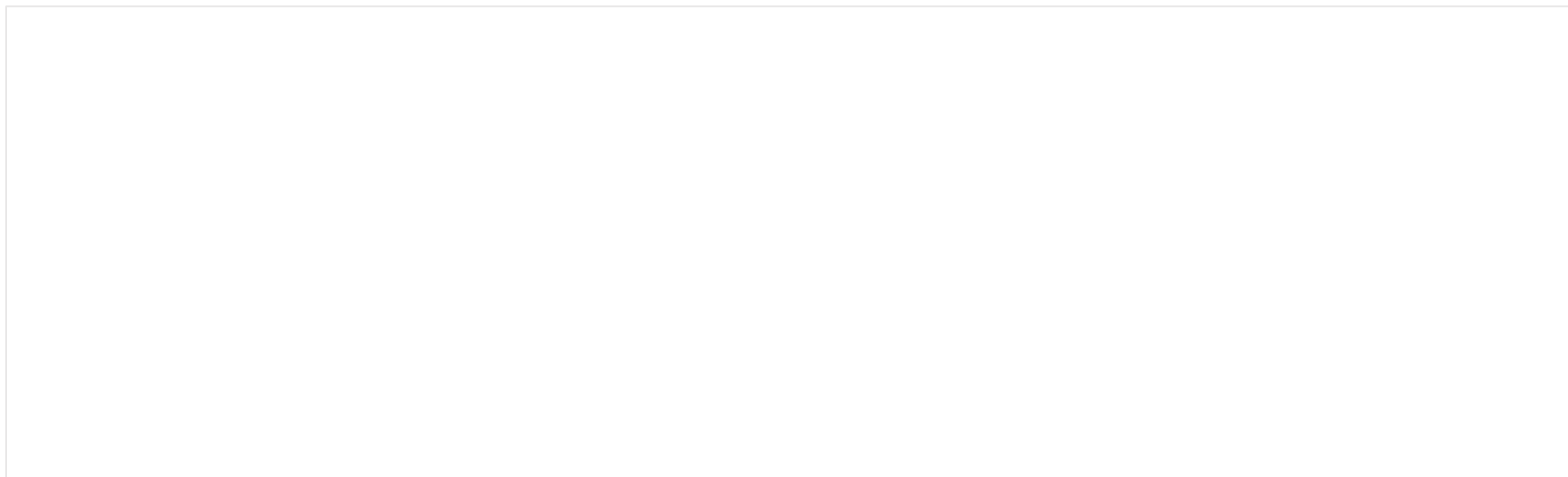
- vote(매개변수: uint자료형의 투표한 제안)
 - 등록된 투표자만 호출하도록 제한
 - 이미 투표한 투표자의 경우는 트랜잭션 실패
 - 제안 내에서 투표했는지 확인, 아니면 트랜잭션 실패
 - 투표자 상세 정보와 제안 상세 정보 업데이트



■ 함수

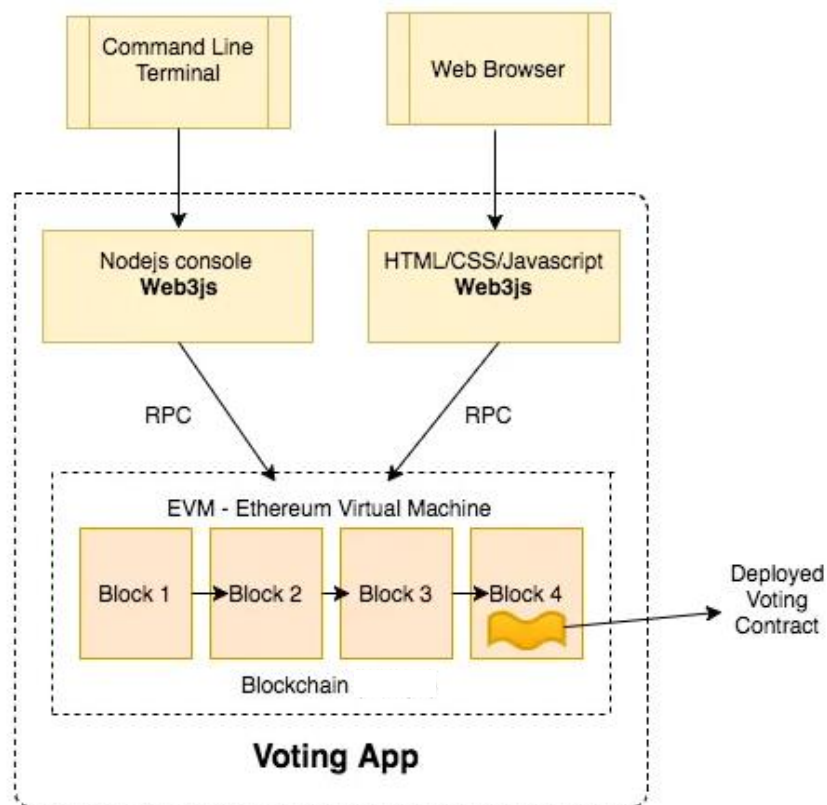
■ reqWinner: 승자 확인

- 반환값: 승리한 제안의 index
- 제안을 담은 배열에서 각 제안의 투표수를 모두 비교하여 가장 큰 값을 갖는 index



■ Dapp(Decentralized application)

- 분산형 애플리케이션은 일반적으로 스마트 계약을 사용하여 분산형 컴퓨팅, 블록체인 또는 기타 분산 원장 시스템에서 실행되는 자율적으로 작동할 수 있는 애플리케이션
- 탈중앙화 분산 어플리케이션
- 블록체인을 기반으로 한 앱



■ Dapp(Decentralized application)

■ Dapp 구성 요소

- 스마트 계약: Dapp의 핵심 로직은 스마트 계약으로 구현되며, 이 계약은 블록체인에 영구적으로 저장되어 누구나 검증하고 호출할 수 있음
- 프론트엔드: 사용자 인터페이스(UI)로 웹 앱 형태로 제공되며, 메타마스크 같은 지갑과 통신하여 스마트 계약을 호출

■ Dapp의 주요 특징

- 탈중앙화
- 스마트 계약
- 투명성: 모든 트랜잭션과 데이터가 블록체인에 기록, 코드와 데이터의 변조 불가
- 신뢰성: 중앙 서버가 없으므로, 서버 다운이나 해킹에 강함

■ HTML/CSS/JavaScript

■ HTML(Hyper Text Markup Language)

- 웹 페이지 표시를 위해 개발된 지배적인 마크업 언어
- 제목, 단락, 목록 등과 같은 본문을 위한 구조적 의미를 나타내는 것뿐만 아니라 링크, 인용과 그 밖의 항목으로 구조적 문서를 만들 수 있는 방법을 제공

■ CSS(Cascading Style Sheet)

- HTML로 구조화된 웹 페이지의 스타일(색상, 글꼴, 여백 등)을 정의하는 언어
- 마크업 언어(ex: HTML)가 웹사이트의 몸체를 담당한다면 CSS는 옷과 액세서리처럼 꾸미는 역할을 담당

■ JavaScript

- 웹문서의 기능적 요소나 동적 요소 제어
- 사용자의 입력이나 이벤트에 반응하는 기능

■ Web3.js

- 이더리움 블록체인과 상호 작용하기 위한 JavaScript 라이브러리
- 이더리움 블록체인에 연결하고 스마트 컨트랙트와 상호 작용하며, 블록체인 기반 응용 프로그램을 개발할 때 사용
- <https://web3js.readthedocs.io/en/v1.10.0/>

■ RPC(remote procedure call, 원격 프로시저 호출)

- 네트워크를 통해 원격 시스템에서 함수를 호출하고 결과를 받게 해주는 프로토콜
- 별도의 원격 제어를 위한 코딩 없이 다른 주소 공간에서 함수나 프로시저를 실행할 수 있게 하는 프로세스 간 통신 기술
- 원격 프로시저 호출을 이용하면 프로그래머는 함수가 실행 프로그램에 로컬 위치에 있든 원격 위치에 있든 동일한 코드를 이용

■ 개발 환경

- **Node.js**: 클라이언트 인터페이스를 위한 웹서버
 - <https://nodejs.org/en>
- **npm**: 노드 패키지 매니저
- **트러플**: 이더리움 스마트 컨트랙트 개발 및 배포를 위한 프레임워크
 - <https://trufflesuite.com/truffle/>
- **가나쉬**: 개발 및 테스트 목적으로 사용할 수 있는 개발용 이더리움 블록체인
 - <https://trufflesuite.com/ganache/>
- 브라우저/웹클라이언트: 크롬과 **메타마스크**
 - 메타마스크는 특정한 블록체인에 연결해 계정을 관리(디지털서명 등)하도록 지원
 - <https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn>



■ Node.js

- 확장성 있는 네트워크 애플리케이션(특히 서버 사이드) 개발에 사용되는 소프트웨어 플랫폼
- Javascript 코드를 실행하는 프로그램
- Javascript 는 웹 페이지내에서 HTML과 함께 사용하여 동적인 기능 구현이 목적
- Node.js는 서버 측면에서 Javascript 코드를 실행하여 다양한 서버 기능을 구현할 수 있음(서버 애플리케이션 개발)
- 빠른 실행 속도, 단순한 구조, 확장성
- 풍부한 모듈: Express, web3.js 등
- Netflix, Linkedin, Paypal, Uber, Facebook 등이 node.js 를 활용

■ npm(Node Package Manager)

- Node.js 패키지 매니저
- Node.js 개발에 필요한 여러 패키지를 설치하고 관리하는 도구
- 간단한 명령어를 통해서 패키지를 검색, 설치, 업데이트가 가능
- 설치하는 node.js를 설치할 때 자동으로 설치됨

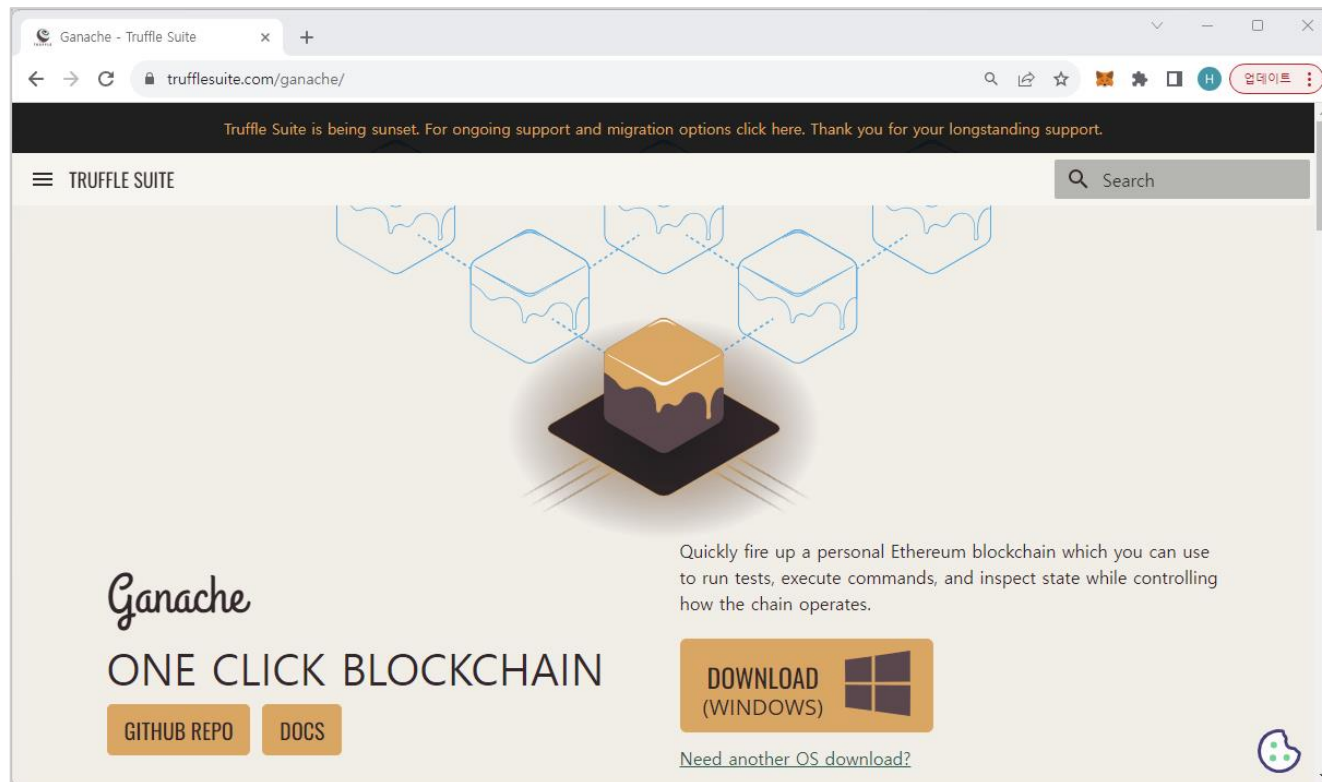
■ express

- Node.js 환경에서 웹 애플리케이션 및 API를 개발하기 위한 프레임워크
- Node.js 모듈 중 하나로 HTTP 요청과 응답을 처리
 - HTTP(HyperText Transfer Protocol): 웹상에서 정보를 주고받을 수 있는 프로토콜
- Node.js의 사실상의 표준 서버 프레임워크

■ Truffle

- 스마트 컨트랙트 개발시 개발, 배포, 테스트 환경을 제공하는 프레임워크
- `npm install -g truffle`
 - Node.js 패키지 매니저를 사용하여 Truffle을 전역으로 설치
- `truffle init`
 - 계약을 테스트하고 배포하는 데 필요한 디렉터리 구조가 잡혀있는 Truffle 프로젝트를 생성
 - `contracts/`: Solidity contracts를 위한 폴더
 - `migrations/`: 배포 스크립트 파일을 위한 폴더
 - `test/`: 컨트랙트의 테스트 파일을 위한 폴더
 - `truffle-config.js`: Truffle 설정 파일

- Ganache(가나쉬)
 - 트리플의 이더리움 테스트 블록체인
 - 10개의 테스트 계정 제공



■ Ganache(가나쉬) QUICKSTART

The screenshot shows the Ganache desktop application window. The top navigation bar includes icons for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below this is a status bar with various network parameters like CURRENT BLOCK, GAS PRICE, GAS LIMIT, HARDFORK, NETWORK ID, RPC SERVER, and MINING STATUS. The main content area displays the 'ACCOUNTS' tab, showing a list of accounts with their addresses, balances, transaction counts, and indices. A mnemonic phrase is also visible at the top of the account list.

ADDRESS	BALANCE	TX COUNT	INDEX
0xC8e9dEda45C037D2f44b6eE82D3519076f38341D	100.00 ETH	0	0
0xA1581EfA6E5C9d439CD5724B0103FADDd8b5Fb5f	100.00 ETH	0	1
0x14498e243E1a3569F9626d9ba8DC6aa2e0D25195	100.00 ETH	0	2
0xa825D4E6Bf540B0EF867cF141f926d458ba2084C	100.00 ETH	0	3
0x687Df89dc77f3bBB2fEF68FE36CB97e3F751CB5E	100.00 ETH	0	4
0x6B17841F8b364804D789a3Db3F2de8449e4B01fB	100.00 ETH	0	5
0xe0657bD70FC9b9ea0Fe37f7B3d3a90e2925254cD	100.00 ETH	0	6

■ 메타마스크

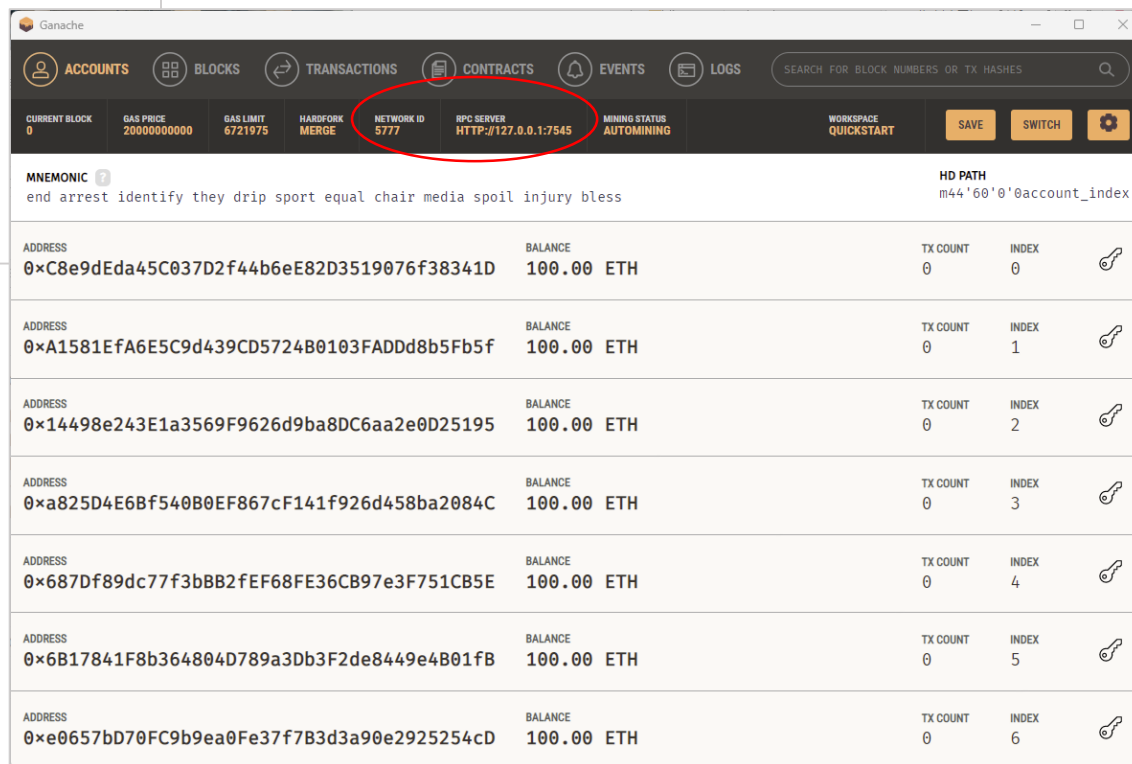
- 블록체인 네트워크와 상호작용하고 스마트 컨트랙트 기반의 분산 응용 프로그램(DApps)을 사용자들이 쉽게 관리하고 실행할 수 있도록 해주는 웹 브라우저 확장 프로그램
- 개인 키를 웹 브라우저 내에서 관리할 수 있으며, 이더를 보낼 수 있고 스마트 컨트랙트와 상호작용할 수 있음
- 사용자가 DApp을 웹 브라우저에서 실행하며, 필요한 트랜잭션을 서명하여 스마트 컨트랙트와 상호작용할 수 있게 함
- 메타마스크 설치(chrome 웹 스토어)
 - <https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn>

■ Truffle를 이용한 스마트컨트랙트 배포

■ truffle-config.js

- 배포할 네트워크 설정: 가나쉬

```
networks: {  
  development: {  
    host: "127.0.0.1",  
    port: 7545,  
    network_id: "5777"  
  }  
}
```



The screenshot shows the Ganache application window. The 'CONTRACTS' tab is highlighted with a red circle. The interface displays various network settings and a list of accounts.

ACCOUNTS	BLOCKS	TRANSACTIONS	CONTRACTS	EVENTS	LOGS
CURRENT BLOCK 0	GAS PRICE 20000000000	GAS LIMIT 6721975	HARDFORK MERGE	NETWORK ID 5777	RPC SERVER HTTP://127.0.0.1:7545

Below the settings, the 'MNEMONIC' is displayed: end arrest identify they drip sport equal chair media spoil injury bless. The 'HD PATH' is m44'60'0'0'account_index.

ADDRESS	BALANCE	TX COUNT	INDEX
0xC8e9dEda45C037D2f44b6eE82D3519076f38341D	100.00 ETH	0	0
0xA1581EfA6E5C9d439CD5724B0103FADDd8b5Fb5f	100.00 ETH	0	1
0x14498e243E1a3569F9626d9ba8DC6aa2e0D25195	100.00 ETH	0	2
0xa825D4E6Bf540B0EF867cF141f926d458ba2084C	100.00 ETH	0	3
0x687Df89dc77f3bBB2fEF68FE36CB97e3F751CB5E	100.00 ETH	0	4
0x6B17841F8b364804D789a3Db3F2de8449e4B01fB	100.00 ETH	0	5
0xe0657bD70FC9b9ea0Fe37f7B3d3a90e2925254cD	100.00 ETH	0	6

■ Truffle를 이용한 스마트컨트랙트 배포

■ 배포 스크립트 작성

- Ballot 스마트컨트랙트 배포를 위한 스크립트 작성
- 2_deploy_contracts.js

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

배포할 스마트
컨트랙트 명시

생성자의 파라미터
포함

- 1_initial_migration.js: Migrations 스마트컨트랙트 배포를 위한 스크립트
- Migrations 스마트컨트랙트(Migration.sol): 배포 정보 관리

■ 웹어플리케이션 개발과 설정

■ npm init

- 패키지(애플리케이션)를 생성(초기화) 해주는 명령어
- package.json 생성: 프로젝트 정보와 의존성(dependencies)을 관리하는 문서
 - package name: 패키지 명, 기본 값은 폴더 명
 - version: 패키지 버전, (1.0.0)는 기본 값
 - description: 패키지에 대한 설명
 - entry point: 시작 파일 명, (index.js)는 기본 값
 - test command: npm test를 호출할 때 실행되는 명령
 - git repository: 패키지가 저장되어 있는 Git 저장소의 URL
 - keywords
 - author: 원작자의 이름
 - License: (ISC)는 기본 값

■ 웹어플리케이션 개발과 설정

■ npm init

- 참고: npm help init
- package.json

```
{
  "name": "ballot-app",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

■ 웹어플리케이션 개발과 설정

■ package.json 수정

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

Node.js 서버 실행
스크립트 설정

express 모듈 의존성
설정

■ 웹어플리케이션 개발과 설정

- express기반 웹어플리케이션 메인스크립트 파일(index.js)
 - 포트 설정, 초기 화면 설정

```
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!');
});
```

■ 웹어플리케이션 개발과 설정

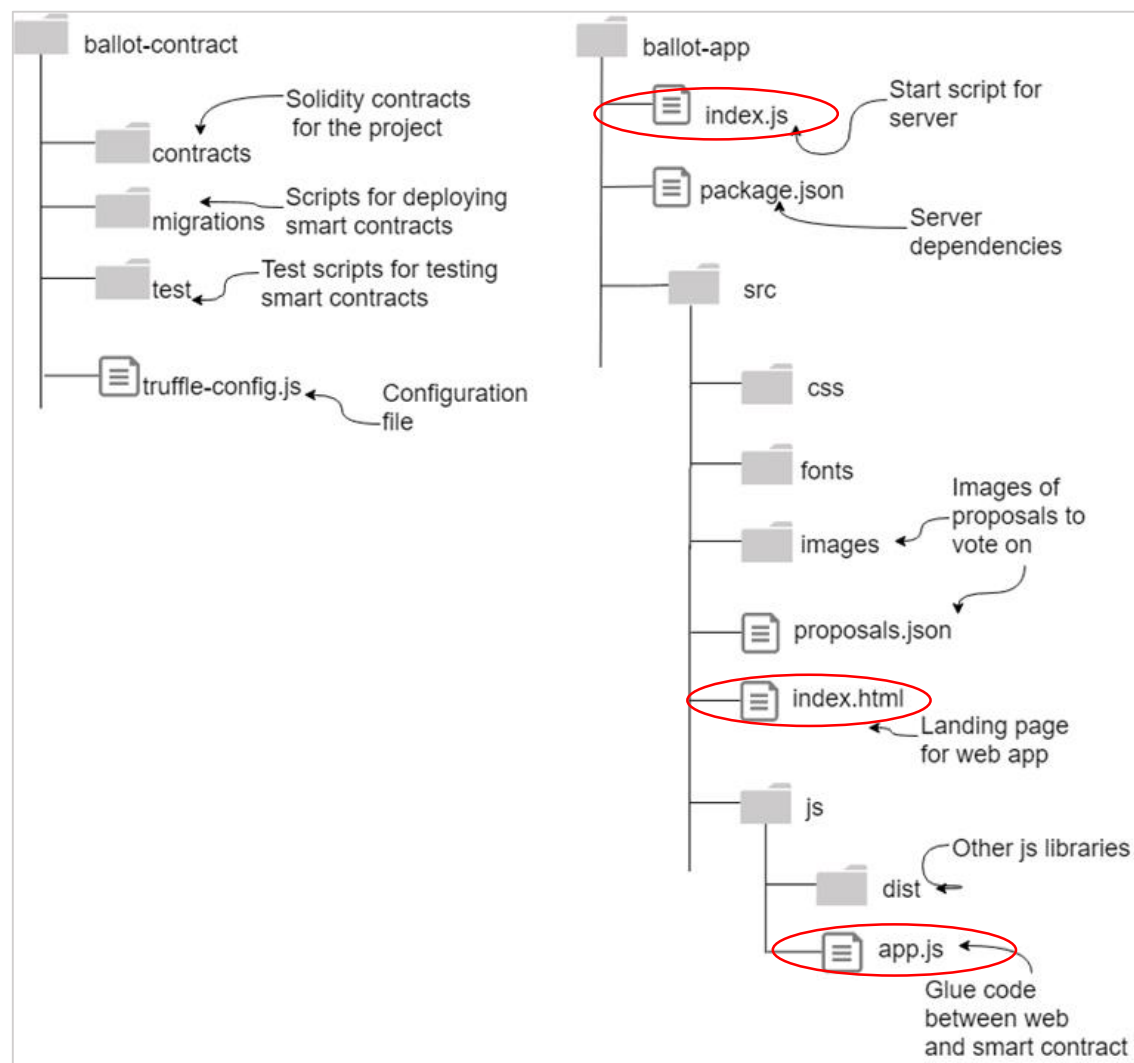
■ npm install

- package.json 파일에 이미 정의된 의존성(dependencies) 모듈(Modules)이 있으면, 나열된 모든 모듈을 로컬 node_modules 폴더에 설치

■ npm start

- 서버를 로컬호스트에서 시작
- index.js 파일에서 설정한 포트(3000)에서 app.js를 론칭해 입력값을 대기하도록 한다

■ 전자투표 Dapp을 위한 전체 구성



- **index.js**

- Node.js 서버를 구동하는 스크립트
- index.html 렌더링
- 리스닝 포트 설정

- **Index.html**

- 웹 어플리케이션을 위한 랜딩페이지

- **app.js**

- ABI 파일을 사용해 스마트컨트랙트와 상호작용
- ABI(Application Binary Interface)
 - 스마트컨트랙트의 함수를 호출하기 위한 인터페이스
 - 컴파일할 때 build 디렉토리에 JSON 파일로 저장

■ JSON(JavaScript Object Notation)

- 키-값 쌍으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷
- 인터넷에서 자료를 주고 받을 때 그 자료를 표현하는 방법으로 알려져 있다
- JSON은 텍스트로 이루어져 있으므로, 사람과 기계 모두 읽고 쓰기 쉽다.
- 프로그래밍 언어와 플랫폼에 독립적이므로, 서로 다른 시스템 간에 객체를 교환하기에 좋다.

```
{  
  "이름": "홍길동",  
  "나이": 55,  
  "성별": "남",  
  "주소": "서울특별시 양천구 목동",  
  "특기": ["검술", "코딩"],  
  "가족관계": {"#": 2, "아버지": "홍판서", "어머니": "춘섬"},  
  "회사": "경기 수원시 팔달구 우만동"  
}
```

- app.js
 - 스마트컨트랙트와 웹UI를 연결
 - 객체 선언: {} 블록, key:value 구조

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

- 시작 코드

```
$(function() {  
  $(window).load(function() {  
    App.init();  
  });  
});
```


- app.js
 - proposals.json에 있는 후보자들 정보 읽어서 HTML로 표시

```
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();  
},
```

- app.js
 - web3Provider 연결(가나쉬 연결)

```
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  
    App.web3Provider = new Web3.providers.HttpProvider(App.url);  
  }  
  web3 = new Web3(App.web3Provider);  
  
  return App.initContract();  
},
```

- 웹페이지를 서비스하는 코드 app.js
- Truffle에서 배포한 스마트컨트랙트 연결

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

- 참고:

- <https://trufflesuite.com/docs/truffle/how-to/contracts/interact-with-your-contracts/>
- <https://archive.trufflesuite.com/guides/pet-shop/>

- 웹페이지를 서비스하는 코드 app.js
 - 해당 프로바이더에 존재하는 계정주소 가져오기

```
populateAddress : function(){  
  new Web3(new Web3.providers.HttpProvider(App.url)).eth.getAccounts((err, accounts) => {  
    jQuery.each(accounts,function(i){  
      var optionElement = '<option value="'+accounts[i]+'">'+accounts[i]+'</option>';  
      jQuery('#enter_address').append(optionElement);  
    });  
  });  
},
```

- web3.eth.getAccounts()
 - 해당 노드에 존재하는 계정들 가져오기

- app.js
 - 웹페이지의 각 버튼 클릭시 호출할 함수 연결

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

■ app.js

■ Register 버튼 클릭

```
handleRegister: function(addr){
  var voteInstance;
  App.contracts.vote.deployed().then(function(instance) {
    voteInstance = instance;
    web3.eth.defaultAccount = web3.eth.accounts[0]
    return voteInstance.register(addr);
  }).then(function(result){
    if(result){
      if(parseInt(result.receipt.status) == 1)
        alert(addr + " registration done successfully")
      else
        alert(addr + " registration not done successfully due to revert")
    } else {
      alert(addr + " registration failed")
    }
  }).catch(function(err){
    console.log(err.message);
    alert(addr + " registration failed")
  });
},
```

- web3.eth.defaultAccount: from을 요구하는 eth 모듈 함수에서 from을 명시하지 않고 기본 값을 사용하도록 설정
- web3.eth.accounts[0]: 현재 메타마스크의 사용자 계정 가져오기

- app.js

- Vote 버튼 클릭

```
handleVote: function(event) {  
  event.preventDefault();  
  var proposalId = parseInt($(event.target).data('id'));  
  var voteInstance;  
  
  var account = web3.eth.accounts[0];
```

```
App.contracts.vote.deployed().then(function(instance) {  
  voteInstance = instance;  
  return voteInstance.vote(proposalId, {from: account});  
}).then(function(result){  
  if(result){  
    console.log(result.receipt.status);  
    if(parseInt(result.receipt.status) == 1)  
      alert(account + " voting done successfully")  
    else  
      alert(account + " voting not done successfully due to revert")  
  }  
  else {  
    alert(account + " voting failed")  
  }  
}).catch(function(err){  
  console.log(err.message);  
  alert(account + " voting failed")  
});  
},
```

- app.js
 - Declare Winner 버튼 클릭

```
handleWinner : function() {  
  console.log("To get winner");  
  var voteInstance;  
  App.contracts.vote.deployed().then(function(instance) {  
    voteInstance = instance;  
    web3.eth.defaultAccount = web3.eth.accounts[0]  
    return voteInstance.reqWinner();  
  }).then(function(res){  
    console.log(res);  
    alert(App.names[res] + " is the winner ! :)");  
  }).catch(function(err){  
    console.log(err.message);  
  })  
}
```


■ 실행 과정

1. 스마트 컨트랙트 작성
2. 가나쉬 실행: 테스트 블록체인 네트워크
3. 스마트 컨트랙트 컴파일&배포(truffle-config.js: 가나쉬 배포 설정)
 - `cd ballot-contract`
 - `truffle migrate`
4. 웹서버(node.js) 실행
 - `cd ballot-app`
 - `npm install`
 - `npm start`

■ 실행 과정

5. 웹브라우저(메타마스크) 실행

- 브라우저에서 localhost:3000 접속
- 메타마스크를 가나쉬 네트워크에 연결
- 가나쉬 계정 가져오기(3개): 편의상 Account2, Account3, Account4 라고 함
 - Account2: chairperson
 - Account3: 투표자1
 - Account4: 투표자2
- 가나쉬 계정(3개) localhost:3000 에 연결

■ 실행 과정


6. 테스트

- Account2가 Account2, Account3, Account4 등록
- Account2가 4개의 제안 중 하나에 투표
- Account3가 4개의 제안 중 하나에 투표
- Account4가 4개의 제안 중 하나에 투표
- Declare Winner 버튼 클릭하여 우승자 확인(모든 계정 가능)

■ 초기 화면


Pick your Favourite

Milli




Vote

Murphy




Vote

Radar



Vote

Riley



Vote

Address :

Register

Declare Winner


■ register 성공

localhost:3000


localhost:3000 내용:
0xef1f687d22414ff358cdd4151da630324e645c01 registration done successfully

확인


Milli



Vote




Vote



Vote

Riley



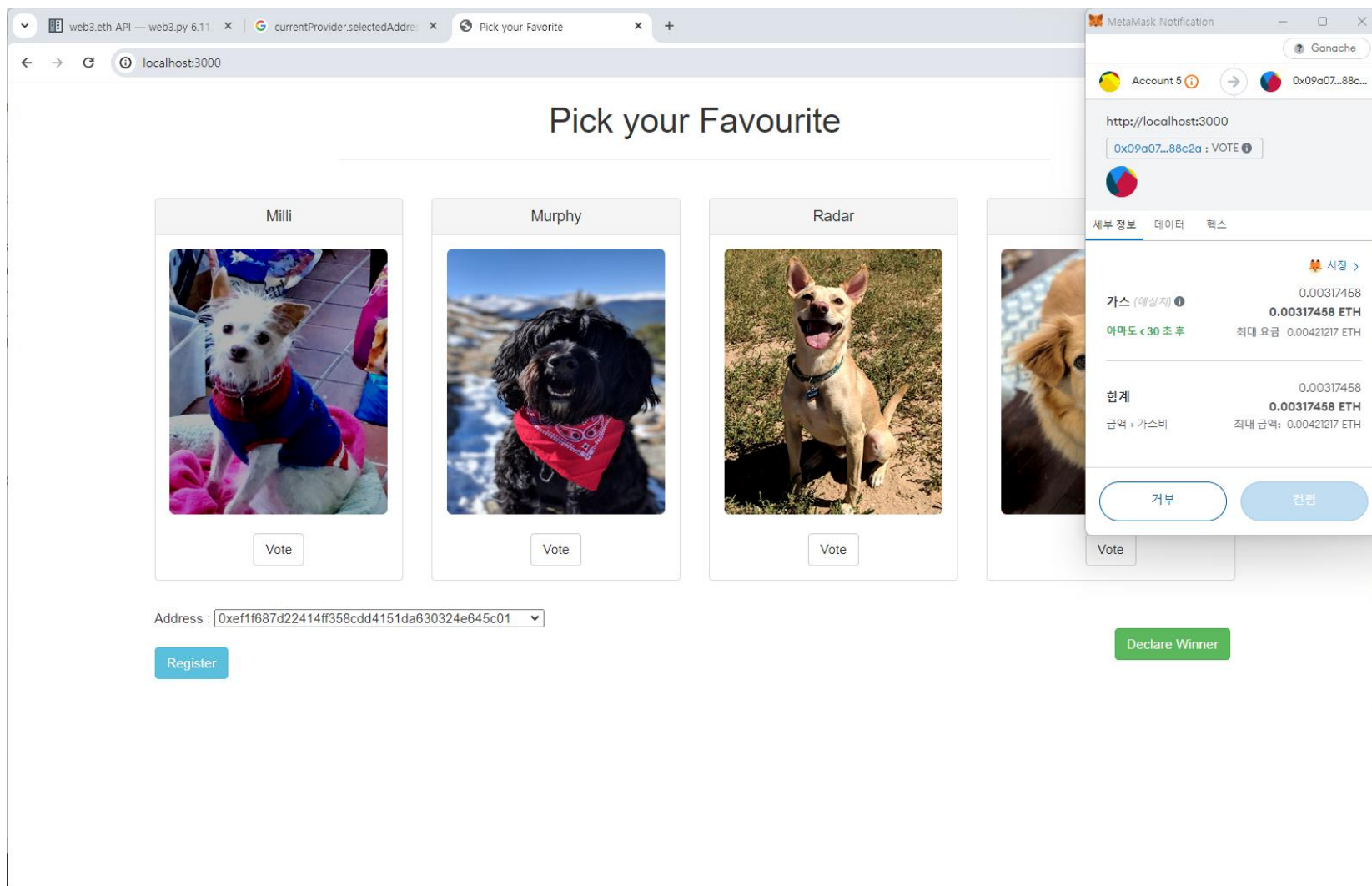
Vote

Address : 0xef1f687d22414ff358cdd4151da630324e645c01

Register

Declare Winner

■ vote 버튼 클릭시 메타마스크 컨펌 화면



■ vote 성공


web3.eth API — web3.py 6.11 x currentProvider.selectedAddr: x Pick your Favorite x +

localhost:3000


localhost:3000 내용:
0x918732ed2534b523df35c92d6f8ec3537db9b6a6 voting done successfully

확인


Milli



Vote




Vote



Vote

Riley



Vote

Address : 0xef1f687d22414ff358cdd4151da630324e645c01

Register

Declare Winner

MetaMask • 지갑
컨펌된 트랜잭션
7 트랜잭션이 컨펌됨

■ Declare Winner 버튼 클릭

