**react-node-ethereum 연동**

=================================== 1단계 ============================

1) ganache 설치/실행 (🡺개인 이더리움 네트웍)

2) MetaMask 설치 (🡺크롬 브라우저용 지갑, 지갑이란 private key 저장소, 외울 수 있으면 지갑 필요 없음 ^^)

3) webstorm 설치 (🡺solidity 개발이 쉬운 편집툴, VS Code보다 부팅 속도 느려 ㅠ.ㅠ but 자동 저장 기능은 좋음 ^^)

4) 솔리디티 개발을 편하게 해주는 플러그인 설치

webstorm settings - plugin 에 solidity 검색해서 설치하면 됨

5) webstorm> terminal > mkdir test1

6) webstorm> terminal > cd test1

7) test1> npm truffle -g (🡺 Node.js 용 이더리움 개발 프레임워크)

8) test1> truffle unbox react (🡺 클라이언트를 리액트로 작성하기 위한 세팅)

9) truffle-config.js에 다음 설정 (🡺가나슈에서 돌고 있는 개인 네트웍에 접속하기 위한 설정)

|  |
| --- |
| **const** path = **require**(**"path"**);  module.**exports** = {  *// See <http://truffleframework.com/docs/advanced/configuration>  // to customize your Truffle configuration!* **contracts\_build\_directory**: path.join(\_\_dirname, **"client/src/contracts"**),  **networks**: {  **develop**: {  **host**: **"127.0.0.1"**,  **port**: 8545,  **network\_id**: **"5777"** }  } }; |

10) test1> truffle compile (🡺 client>src>contracts 밑에 두 개의 json이 생김, 이 json 파일 안에 컴파일 된 bytecode가 들어 있음))

11) 반드시 가나슈 돌고 있는 상태에서

test1> truffle migrate --network develop (🡺컴파일 된 스마트 컨트랙트를 가나슈에 배포하는 것, develop이라는 이름은 9번에서 설정한 이더리움 네트워크의 이름)

12) 가나슈에서 블록이 생성된 것을 확인

|  |
| --- |
|  |

13) 가나슈에서 계정의 개인키 하나 복사

|  |
| --- |
|  |

14) 메타마스크에서 계정 가져오기 선택 (🡺 내 지갑에 개인키를 보관하기 위한 작업)

|  |
| --- |
|  |

15) 가나슈에서 복사한 개인 키 입력하고 가져오기 버튼 클릭

|  |
| --- |
|  |

16) test1> cd client

17) client> npm i 한 뒤 npm start ( 만약 메타마스크에서 인증 실패가 난다면 다른 계정을 가져와 시도해 본다)

|  |
| --- |
|  |

18) App.js를 다음과 같이 수정

|  |
| --- |
| **import React**, { Component } **from "react"**; **import** SimpleStorageContract **from "./contracts/SimpleStorage.json"**; **import** *getWeb3* **from "./utils/getWeb3"**;  **import "./App.css"**;  **class** App **extends** Component {  **state** = { **storageValue**: 0, **web3**: **null**, **accounts**: **null**, **contract**: **null** };   *componentDidMount* = **async** () => {  **try** {  *// Get network provider and web3 instance.* **const** web3 = **await** *getWeb3*();   *// Use web3 to get the user's accounts.* **const** accounts = **await** web3.**eth**.getAccounts();   *// Get the contract instance.* **const** networkId = **await** web3.**eth**.net.getId();  **const** deployedNetwork = SimpleStorageContract.networks[networkId];  **const** instance = **new** web3.**eth**.Contract(  SimpleStorageContract.abi,  deployedNetwork && deployedNetwork.**address**,  );   *// Set web3, accounts, and contract to the state, and then proceed with an  // example of interacting with the contract's methods.* **this**.setState({ web3, accounts, **contract**: instance }, **this**.*runExample*);  } **catch** (error) {  *// Catch any errors for any of the above operations.  alert*(  **`Failed to load web3, accounts, or contract. Check console for details.`**,  );  ***console***.error(error);  }  };   *runExample* = **async** () => {  **const** { accounts, contract } = **this**.**state**;   *// Stores a given value, 5 by default.* **await** contract.methods.set(5).send({ **from**: accounts[0] });   *// Get the value from the contract to prove it worked.* **const** response = **await** contract.methods.get().call();   *// Update state with the result.* **this**.setState({ **storageValue**: response });  };   *runExample2* = **async** () => {  *alert*(**this**.**input**.**value**);  **const** { accounts, contract } = **this**.**state**;   *// Stores a given value, 5 by default.* **await** contract.methods.set(**this**.**input**.**value**).send({ **from**: accounts[0] });   *// Get the value from the contract to prove it worked.* **const** response = **await** contract.methods.get().call();   *// Update state with the result.* **this**.setState({ **storageValue**: response });  };  render() {  **if** (!**this**.**state**.**web3**) {  **return** <**div**>Loading Web3, accounts, and contract...</**div**>;  }  **return** (  <**div className="App"**>  <**h1**>Good to Go!</**h1**>  <**p**>Your Truffle Box is installed and ready.</**p**>  <**h2**>Smart Contract Example</**h2**>  <**p**>  If your contracts compiled and migrated successfully, below will show  a stored value of 5 (by default).  </**p**>  <**p**>  Try changing the value stored on <**strong**>line 40</**strong**> of App.js.  </**p**>  <**div**>The stored value is: {**this**.**state**.**storageValue**}</**div**>  <**input ref=**{ref => **this**.**input** = ref} ></**input**><**button onClick=**{**this**.*runExample2*} > add</**button**>  </**div**>  );  } }  **export default** App; |

19) Ctrl+C로 잠시 리액트 실행 멈추고

20) client> yarn eject (🡺config, scripts 폴더 생김 + package.json 변경됨)

21) client> npm i express

22) client> mkdir server

23) client>server에서 server.js 작성

|  |
| --- |
| **const express** = *require*(**'express'**); **const** path = *require*(**'path'**); **const** os = *require*(**"os"**);  **const** app = **express**(); **const** PORT = ***process***.**env**.**PORT** || 4000;  app.use(**express**.**static**(path.*join*(***\_\_dirname***, **'..'**, **'public/'**)));  *// if you need api routes add them here* app.get(**"/api/getUsername"**, **function**(req, res, next){  res.send({ **username**: os.*userInfo*().**username** }); });  app.listen(PORT, () => {  ***console***.log(**`Check out the app at http://localhost:**${PORT}**`**); }); |

24) client/config/webpack.config.dev.js의 plugins: [ 위치 (338라인정도!!!) 위에 다음 내용 추가 (🡺 express 서버는 4000번 포트를 사용하도록 설정한 것)

|  |
| --- |
| }, **devServer**: {  **port**: 4000,  **open**: **true**,  **proxy**: {  **"/"**: **"http://localhost"** } },  **plugins**: [  … |

25) package.json 맨 아래에 다음 내용 추가

|  |
| --- |
| },  **"babel"**: {  **"presets"**: [  **"react-app"** ]  }  ,  **"proxy"** : **"http://localhost:4000"** } |

26) App.js 에 다음 내용 추가

|  |
| --- |
| *runExample2* = **async** () => {  *alert*(**this**.**input**.**value**);  *fetch*(**'/api/getUsername'**)  .then(res => res.json())  .then(user=>**this**.setState({**username**:user.**username**}));   **const** { accounts, contract } = **this**.**state**; |

27) 새 cmd에서 서버 가동

|  |
| --- |
|  |

28) WebStorm 터미널에서 리액트 가동 (가나슈 동작 중인지 확인하고)

client> npm start

29) add 버튼을 눌렀을 때 다음과 같이 적용되면 react-node-ethereum 잘 연동되고 있는 것임

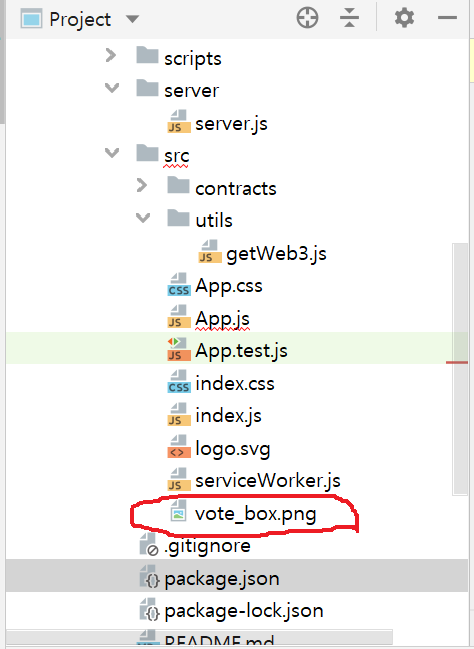
|  |
| --- |
|  |

========================== 2단계 : 투표 dApp 만들기 ======================

1) App.js에서 투표 앱을 만들기 위한 화면을 꾸민다

|  |
| --- |
| **import** React, { Component } **from "react"**; **import "./App.css"**; **import** logo **from './vote\_box.png'**;  **class** App **extends** Component {  **state** = {  **totalVoteNum**: 0, //총투표수  **titleMsg**: **"잠시만 기다려주세요. 투표 서버에 접속 중입니다..."**,  **candidate**:**""**, //후보자  **totalVotesFor** : 0 //득표수  };   *componentDidMount* = **async** () => {  *fetch*(**'http://localhost:4000/api/init'**)  .then(res => res.json())  .then(parsedData=> **this**.setState({**titleMsg**: parsedData.**titleMsg**}));  };   *vote* = **async** (e) => {  e.preventDefault();  **const** candidate=**this**.**voteForm**.**candidate**.**value**;  **if**(***window***.confirm(candidate+**"를 선택하셨습니다. 맞습니까?"**)) {  **const** a={  **'candidate'**:candidate  };  ***console***.log(a);  *fetch*(**'http://localhost:4000/api/vote'**, {  **method**: **'post'**,  **headers**: {**'Content-Type'**: **'application/json'**},  **body**: ***JSON***.stringify(a)  })  .then(res => res.json())  .then(parsedData => **this**.setState({  **titleMsg**: parsedData.**titleMsg**,  **totalVoteNum**: parsedData.**totalVoteNum**,  **candidate**: parsedData.**candidate**,  **totalVotesFor**: parsedData.**totalVotesFor** }));  }   };   render() {   **return** (  <div className=**"App"**>  <h1>{**this**.state.**titleMsg**} </h1>  <img src={logo} />  <h2>회장 선출</h2>  <h3>  다음 중 회장으로 선출되길 원하시는 사람 한 명을 선택하고 투표 버튼을 누르세요  </h3>  <form ref={**ref** => **this**.**voteForm** = ref} onSubmit={**this**.vote}>  <br/><input ***type***=**"radio" *name***=**"candidate" *value***=**"홍길동"** /> 홍길동  <br/><input ***type***=**"radio" *name***=**"candidate" *value***=**"이영애"** /> 이영애  <br/><input ***type***=**"radio" *name***=**"candidate" *value***=**"전지현"** /> 전지현  <br/><input ***type***=**"radio" *name***=**"candidate" *value***=**"박보검"** /> 박보검  <br/><input ***type***=**"radio" *name***=**"candidate" *value***=**"장동건"** /> 장동건  <br/><button ***type***=**"submit"**> 투표</button>  </form>  <div>전체 투표 **수**: {**this**.**state**.**totalVoteNum**}</div>  <div>{**this**.state.**candidate**} 후보자의 득표 **수**: {**this**.**state**.**totalVotesFor**}</div>   </div>  );  } }  **export default** App; |

2) src 밑에 vote\_box.png를 넣는다

3) 4000번 포트에서 동작 중인 server.js가 직접 이더리움 코어에 접속할 수 있도록 다음과 같이 설치하고 작성한다

client> npm i ethers cors

|  |
| --- |
| **const** SimpleStorageContract = *require*( **"../src/contracts/SimpleStorage.json"**); **const** Web3= *require*( **"web3"**); **let** web3=**new** Web3(); **let** accounts=**null**; **let** contract =**null**;  **const** ethers = *require*(**'ethers'**); **const** cors=*require*(**'cors'**);  **const express** = *require*(**'express'**); **const** path = *require*(**'path'**); **const** os = *require*(**"os"**);  **const** app = **express**(); **const** PORT = ***process***.**env**.**PORT** || 4000;  app.use(cors()); app.use(**express**.**static**(path.*join*(***\_\_dirname***, **'..'**, **'public/'**)));  app.use(**express**.**json**());  app.get(**"/api/init"**, **async** (req, res, next)=>{  **try** {  web3.setProvider(**new** Web3.providers.HttpProvider(**'http://localhost:8545'**));  accounts = **await** web3.**eth**.getAccounts();  **const** networkId = **await** web3.**eth**.net.getId();  **const** deployedNetwork = SimpleStorageContract.networks[networkId];  contract = **new** web3.**eth**.Contract(  SimpleStorageContract.abi,  deployedNetwork && deployedNetwork.**address**,  );  *//console.log(os.userInfo().username+"님 투표 가능하십니다");* res.send({ **titleMsg**: os.*userInfo*().**username**+**"님 투표 가능하십니다"** });  } **catch** (error) {  ***console***.error(error);  res.send({ **titleMsg**: error.**message** });  } });  app.post(**"/api/vote"**, **async** (req, res, next)=>{  **try** {   **const** candidate=req.**body**.**candidate**;  **const** \_candidate=ethers.utils.*formatBytes32String*(candidate);  **await** contract.**methods**.voteForCandidate(\_candidate).send({ **from**: accounts[0] });  *// 전체 투표 수* **const** totalVoteNum = **await** contract.**methods**.get().call();  *// 후보자 득표 수* **const** totalVotesFor= **await** contract.**methods**.**totalVotesFor**(\_candidate).call();  res.send(***JSON***.stringify({ totalVoteNum, candidate, totalVotesFor}));  } **catch** (error) {  ***console***.error(error);  res.send({ **titleMsg**: error.**message** });  } });  app.listen(PORT, () => {  ***console***.log(**`Check out the app at http://localhost:**${PORT}**`**); }); |

4) SimpleStorage.sol을 열어 이더리움 코어를 다음과 같이 수정한다

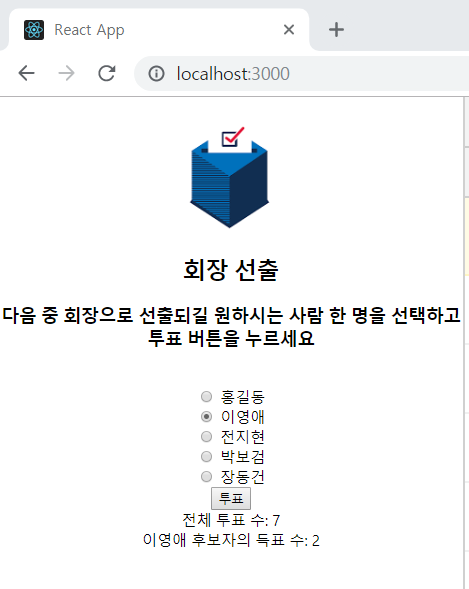
|  |
| --- |
| **pragma** solidity ^0.5.0;  **contract** SimpleStorage {  **uint** storedData;   **function** set(**uint** x) **public** {  storedData = x;  }   **function** get() **public view returns** (**uint**) {  **return** storedData;  } *////////////////////////// vote contract ////////////* **mapping** (**bytes32** => **uint8**) votesReceived;   **constructor**() **public**{  votesReceived[**"홍길동"**]=0;  votesReceived[**"이영애"**]=0;  votesReceived[**"전지현"**]=0;  votesReceived[**"박보검"**]=0;  votesReceived[**"장동건"**]=0;  }  **function** addCandidate(**bytes32** candidate) **public**{  votesReceived[candidate]=0;  }  **function** voteForCandidate(**bytes32** candidate) **public**{   votesReceived[candidate] +=1;  set(get()+1);  }  **function** totalVotesFor(**bytes32** candidate) **view public returns**(**uint8**){   **return** votesReceived[candidate];  } } |

5) truffle compile --all

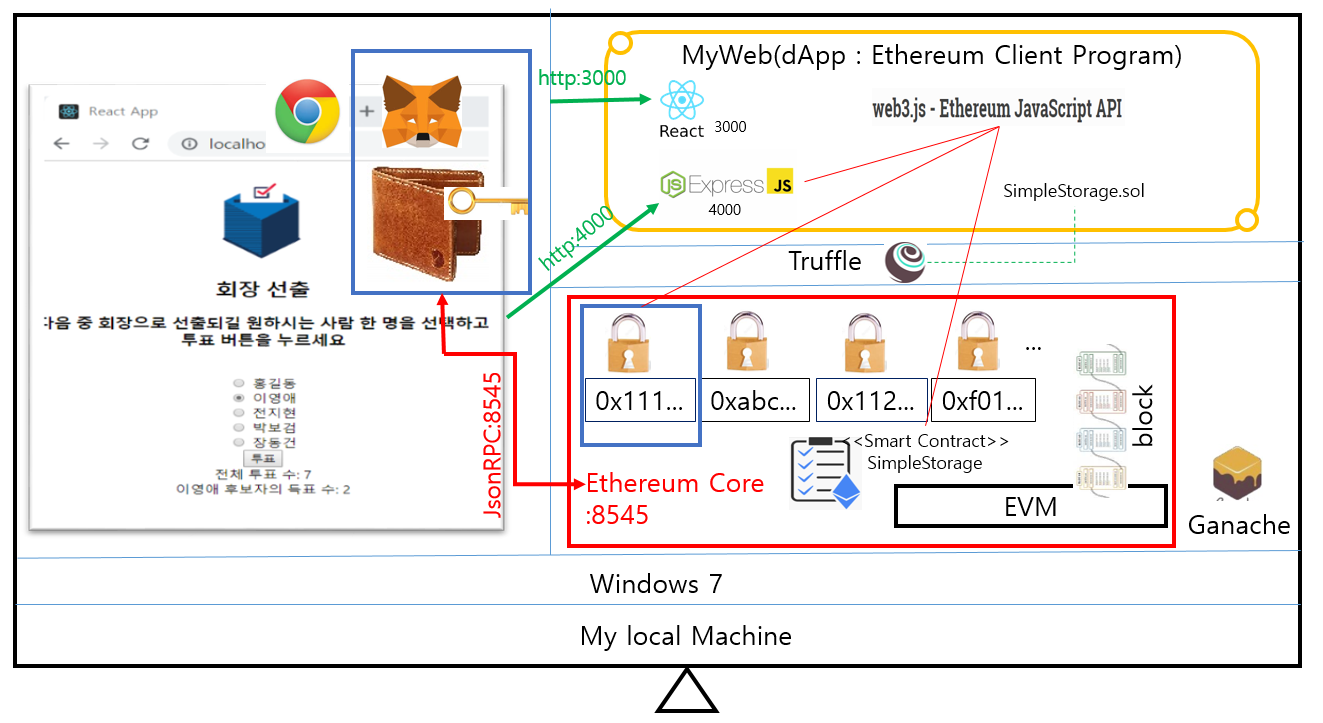
6) truffle migrate --reset --network develop

7) node server.js

8) client> npm start



\*\*\* 이 투표 앱의 아키텍처



\*\*\* 메인넷에 접속 되었을 때의 아키텍쳐

