Eth 지갑 생성 및 전송

1. node.js 설치

   설치 확인

```
node -v

npm -v
```

2. Eth 지갑 생성

```
npm install ethereumjs-wallet --save

// --save
// package.json의 dependency 항목에 모듈을 추가한다는 의미
// npm5부터는 자동적용하므로 신경 X
```

3. wallet.js 생성

```
const Wallet = require("ethereumjs-wallet");
const fs = require("fs");

/**
 * 이더 지갑 생성 및 저장
 *
 */
// EthWallet 생성
function createEthWallet() {
    var Wallet = require('ethereumjs-wallet');
    const EthWallet = Wallet.default.generate();

    const address = EthWallet.getAddressString();
    const privateKey = EthWallet.getPrivateKeyString();

    console.log("address: " + address);
    console.log("privateKey: " + privateKey);

    const content
        = "address: "+address+"\n"+"privateKey: "+privateKey;

    const time = Date.now();
    const fileName = __dirname + "\\wallet\\ethWallet_" + time + ".txt"

    // 생성된 EthWallet 파일로 저장.
    const dir = fs.existsSync(__dirname + "\\wallet")
    // console.log(dir)

    function createFile() {
        fs.writeFile(fileName, content, err => {
            if (err) {
                console.error(err)
            }
        });
    }

    if (dir==true) {
        createFile();
    }
    else if (dir != true) {
        fs.mkdirSync(__dirname + "\\wallet");
        createFile();
```

```
    }
}

createEthWallet();
```

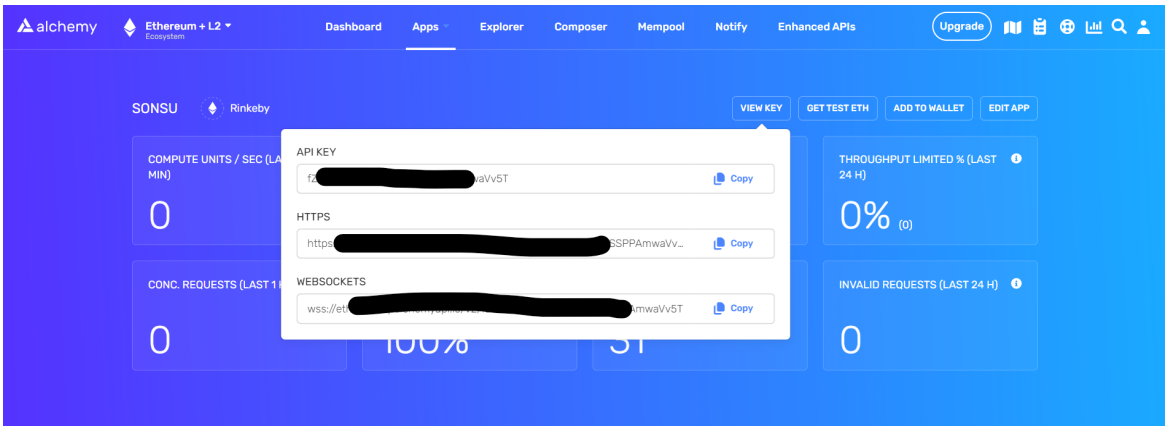
```
// wallet.js 실행
node wallet.js
```

4. 메타마스크에서 생성한 이더 지갑을 불러오고 테스트넷 설정을 활성화하기
    - Chrome Extension에서 메타마스크를 설치
    - `계정 가져오기`를 통해 생성된 이더 지갑의 프라이빗키를 붙여넣고 이더 지갑을 불러오기
    - `설정 - 고급 - 테스트 네트워크 보기` 활성화

4. 테스트용으로 사용할 이더 받아오기
    - https://rinkebyfaucet.com/ 로 가서 이더 주소를 넣고 이더를 받는다 (24시간에 0.1eth 수령 가능)
    - 우리는 rinkeby 테스트넷을 사용할 예정이기 때문에 이 사이트를 사용하는데, 다른 테스트넷을 사용할경우 그 곳에 알맞는 사이트를 찾아 이더를 수령하면 된다.

4. alchemy.io 사이트에서 계정 생성 후 APP 생성하기
    - Environment는 `Development`, Chain은 `Ethereuam`, Network는 `rinkeby`
    - Free plan, Capped Capacity option을 선택(앱을 처음 만드는 경우에만 선택)
    - 그리고 생성된 앱에서 `View Key`를 선택하면
        - `API KEY`
        - `HTTPS`
        - `WEBSOCKETS`

      가 보인다. 각 변수들은 여기 값들을 집어넣어주면 된다.

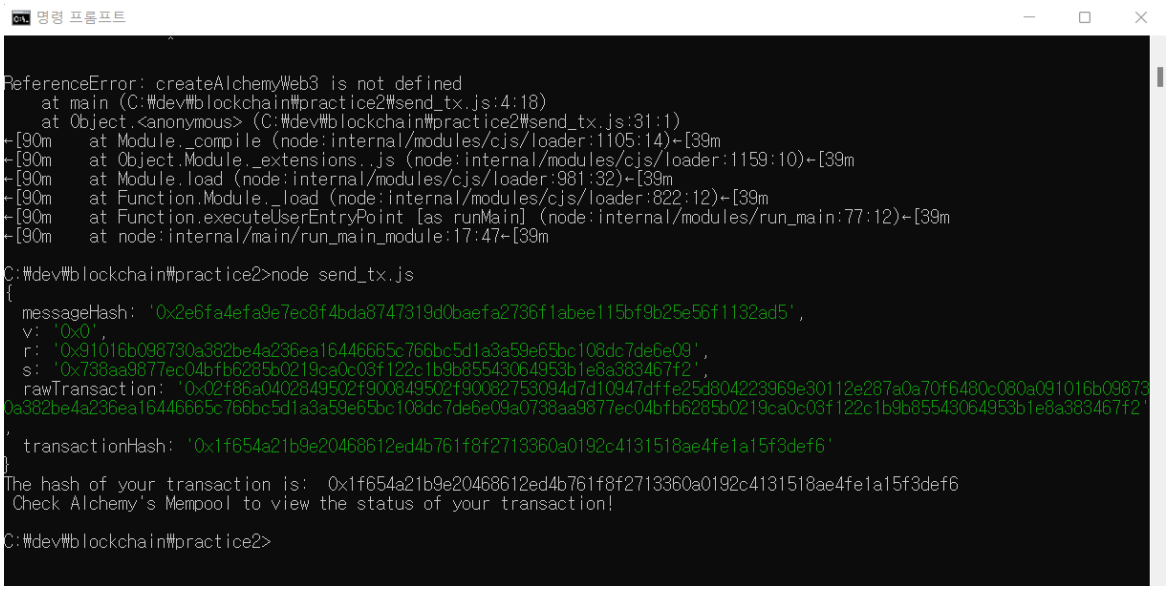


5. 전송파일 생성 (send_tx.js)

```
// alchemy-web3 패키지 설치
npm install @alch/alchemy-web3

// send_ts.js 생성

// import {createAlchemyWeb3} from "@alch/alchemy-web3";

async function main() {
    const API_URL = 'HTTPS'; //HTTPS
    const PRIVATE_KEY = 'PRIVATEKEY';  // 생성한 월렛의 PRIVATEKEY
    const { createAlchemyWeb3 } = require("@alch/alchemy-web3");
    const web3 = createAlchemyWeb3(API_URL);
    const frm = '0x7EC76bdc87CB0159222f5dbAb4522eA8F81a9635';  // 이더를 보내는 월렛 주소
    // TODO: replace this address with your own public address
    const nonce = await web3.eth.getTransactionCount(frm, 'latest'); // nonce starts counting from 0
    const transaction = {
        'to': '0xD7D10947Dffe25D804223969E30112e287A0a70F', // 이더를 받을 월렛 주소
        'value': 100,     // 보낼 이더 수량
                          // 단위는 wei, 1 eth = 1,000,000,000,000,000,000 wei
                          // 단위에 대한 상세한 내용은 https://gwei.io/kr/ 참고
        'gas': 30000,     // 전송 수수료
        'maxFeePerGas': 2500000000, // 이더리움 네트워크 상태의 복잡도에 따라 수수료는 상승. 그 상승폭의 제한을 얼마로 할 것인지 조절
        'nonce': nonce,   // 이 계정에서 전송하는 트랜잭션에 임시로 할당된 해시값. 0부터 순차적으로 상승하며 동일한 논스는 발생하지 않는다.
                          // 이중지불문제를 해결하기 위해 도입된 시스템.
        // optional data field to send message or execute smart contract
    };

    // web3.eth.getMaxPriorityFeePerGas().then(console.log);
    const signedTx = await web3.eth.accounts.signTransaction(transaction, PRIVATE_KEY);
    console.log(signedTx);

    web3.eth.sendSignedTransaction(signedTx.rawTransaction, function (error, hash) {
        if(!error) {
            console.log("The hash of your transaction is: ",
                hash, "\n Check Alchemy's Mempool to view the status of your transaction!")
        } else {
            console.log("Someting went wrong while submitting your transaction: ", error)
        }
    });
}

main();
```

5. send_tx.js 파일 실행

```
node send_tx.js
```



6. 트랜잭션 확인

   rinkeby 테스트넷에서 진행하였으니, rinkeby 네트워크에서 트랜잭션을 확인해본다.

   https://rinkeby.etherscan.io