Web3 애플리케이션(dApp)

■ Web3의 핵심

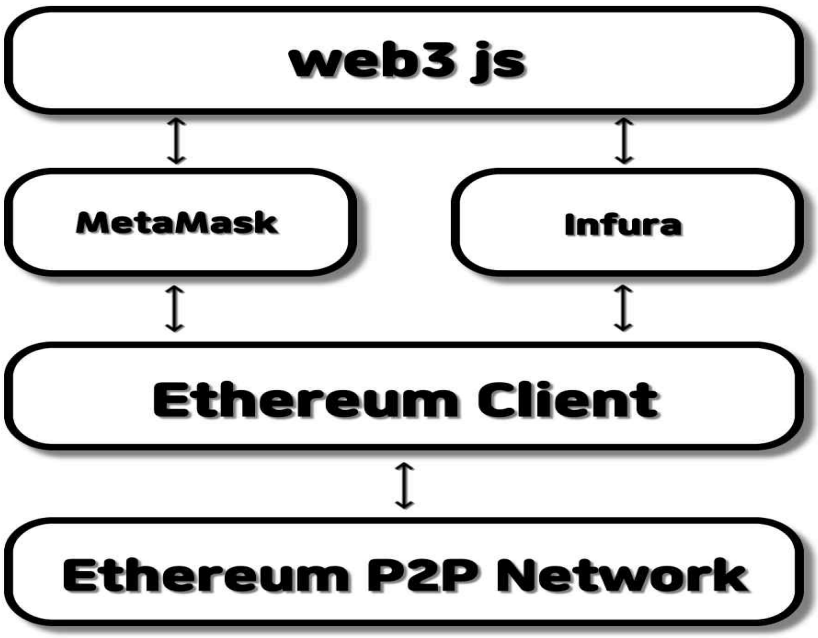
- 블록체인 노드와의 연결을 통해 데이터를 읽거나 트랜잭션을 발생시키는 것

- Web3 - 블록체인 기술 기반의 차세대 인터넷을 의미(이더리움과 같은 블록체인 네트워크와 상호작용하기 위한 라이브러리 또는 인터페이스의 총칭)

- web3.js - 이더리움과 같은 EVM(Ethereum Virtual Machine) 호환 블록체인 네트워크와 상호작용할 수 있도록 돕는 JavaScript 라이브러리 = 즉, web3는 개념이고, web3.js는 이 개념을 실제 개발에 구현하기 위한 도구

    * Web3를 사용하면 웹사이트나 앱이 스마트 계약을 호출하거나, 계정 잔액을 확인하거나, 트랜잭션을 전송할 수 있음

■ 간단한 예제(1)(Web3.js 라이브러리를 사용하여 이더리움 블록체인의 현재 블록 번호를 가져오는 것)



- 중앙화된 서버 대신 분산된 블록체인 네트워크에서 정보를 직접 '읽는' Web3 애플리케이션(dApp)의 가장 기본적인 형태 ⇒ JavaScript 환경(Node.js 또는 브라우저)에서 Web3.js 라이브러리를 사용하여 이더리움 네트워크(여기서는 Infura와 같은 공개 RPC 엔드포인트)에 연결하고 현재 블록 번호를 가져오는 방법

* 탈중앙화된 데이터 접근 : Web2 애플리케이션이 Google 서버나 AWS 데이터베이스에 요청을 보내는 것과 달리, 이 코드는 중앙 기관이 아닌 이더리움 블록체인 네트워크에 직접 정보를 요청
* Web3 라이브러리 : Web3.js는 dApp 프론트엔드(클라이언트)가 블록체인과 통신하기 위해 사용하는 "접착제", 즉 블록체인에 명령을 내릴 수 있는 API를 제공
* 스마트 컨트랙트 상호작용의 기초 : 블록 번호를 읽는 것은 가장 단순한 형태의 블록체인 상호작용이며, 스마트 컨트랙트에 저장된 데이터(예: 토큰 잔액, NFT 소유자 등)를 조회하는 것도 이와 동일한 기본 패턴을 따름

(1) 프로젝트 설정 (Node.js 환경 기준)

- 터미널에서 다음 명령을 실행하여 프로젝트를 초기화하고 web3 라이브러리를 설치
  * npm init -y
  * npm install web3

```
G:₩내 드라이브₩블록체인₩블록체인강의리스트₩강의목록₩11월 4일 강의₩확인₩web3_test>npm init -y
Wrote to G:₩내 드라이브₩블록체인₩블록체인강의리스트₩강의목록₩11월 4일 강의₩확인₩web3_test₩package.json:

{
  "name": "web3_test",
  "version": "1.0.0",
  "description": "",
  "main": "getBlock.js",
  "scripts": {
    "test": "echo ₩"Error: no test specified₩" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

```
G:₩내 드라이브₩블록체인₩블록체인강의리스트₩강의목록₩11월 4일 강의₩확인₩web3_test>npm install web3
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN notsup Unsupported engine for @noble/hashes@1.4.0: wanted: {"node":">= 16"} (current: {"node":"14.16.0","npm":"6.14.11"})
npm WARN notsup Not compatible with your version of node/npm: @noble/hashes@1.4.0
npm WARN notsup Unsupported engine for @ethereumjs/rlp@5.0.2: wanted: {"node":">=18"} (current: {"node":"14.16.0","npm":"6.14.11"})
npm WARN notsup Not compatible with your version of node/npm: @ethereumjs/rlp@5.0.2
npm WARN abitype@0.7.1 requires a peer of typescript@>=4.9.4 but none is installed. You must install peer dependencies yourself.
npm WARN web3_test@1.0.0 No description
npm WARN web3_test@1.0.0 No repository field.

+ web3@4.16.0
added 74 packages from 67 contributors and audited 74 packages in 53.522s

24 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

(2) 예제 코드(getBlock.js)

```
// web3 라이브러리 불러오기

const  {Web3} = require('web3');

// 중요: 블록체인 노드에 접근하기 위한 RPC 엔드포인트 URL
// 실제 서비스에서는 Infura, Alchemy 등의 계정을 만들어 사용
// 이 예제에서는 공개적으로 접근 가능한 이더리움 메인넷 RPC를 사용

//RPC 엔드포인트 URL이란 - 원격 프로시저 호출(Remote Procedure Call, RPC)을 수행할 수 있도록
//서버나 노드(Node)가 제공하는 네트워크 주소(URL)를 말함 --> 쉽게 말해, 내 컴퓨터(클라이언트)에
있는
//프로그램이 멀리 떨어진 다른 컴퓨터(서버 또는 블록체인 노드)에 있는 특정 기능(프로시저)을
//마치 내 프로그램의 함수를 호출하듯 요청하고 결과를 받을 수 있게 해주는 통신 창구의 주소

const rpcURL = 'https://eth.llamarpc.com';

// Web3 인스턴스 생성 및 노드에 연결
const web3 = new Web3(rpcURL);
async function getLatestBlockNumber() {
    try {
        console.log("이더리움 블록체인에 연결 중...");

        // web3.eth.getBlockNumber() 메서드를 사용하여 최신 블록 번호를 요청
        const blockNumber = await web3.eth.getBlockNumber();
        console.log('-----------------------------------------');
        console.log(`    현재 이더리움 블록 번호: ${blockNumber.toString()}`);
        console.log('-----------------------------------------');
        console.log('이것이 Web3를 사용하여 블록체인 데이터를 읽는 기본 동작입니다.');

    } catch (error) {
        console.error('    블록체인 연결 또는 조회 중 오류 발생:', error);
    }
}// 함수 실행
getLatestBlockNumber();
```

(3) 코드 실행

- 터미널에서 작성한 파일을 실행
    * node getBlock.js

■ web3의 간단한 예제(2)

- 가장 간단한 Web3 예제는 일반적으로 Python의 web3.py 또는 JavaScript의 ethers.js(혹은 web3.js) 라이브러리를 사용하여 블록체인의 현재 상태를 읽어오는 작업임

- 웹 개발에서 가장 널리 쓰이는 JavaScript(Ethers.js)를 사용한 간단한 예제를 소개

    * 공개된 이더리움 노드 (Infura 또는 Alchemy)에 연결하여 다음 두 가지를 조회
        - 현재 블록 번호
        - 특정 지갑 주소의 잔액

- 1) 전제 조건

    * Ethers.js 설치

        - Node.js 환경에서 테스트하는 경우 npm install ethers가 필요
        - 웹 환경에서는 CDN(Content Delivery Network, 지리적인 제약 없이 전 세계 사용자에게 빠르고 안전하게 컨텐츠 전송을 할 수 있는 기술)을 사용할 수 있음
        - RPC Endpoint - 블록체인에 연결하기 위한 RPC URL이 필요(예: Infura, Alchemy 등에서 제공하는 공개 URL)

- 2) JavaScript 코드 예제

    * 이 코드는 실제 웹 페이지의 <script> 태그 안이나 Node.js 환경에서 실행할 수 있음

    * 이 코드는 스마트 계약을 배포하거나 트랜잭션을 전송하는 복잡한 과정 없이 Web3가 블록체인 데이터에 접근하는 기본적인 방식을 보여줌

    * 이것이 Web3 개발의 첫걸음이자 가장 간단한 예제임

```
// Ethers.js 라이브러리가 로드되었다고 가정

// --- 설정 값 ---
// 1. 이더리움 메인넷에 연결할 RPC URL (공개 서비스 제공자의 URL)
const RPC_URL = "YOUR_INFURA_OR_ALCHEMY_RPC_URL"; // 실제 URL로 대체

// 2. 잔액을 조회할 대상 주소 (예시: Binance 핫 월렛 주소)
const TARGET_ADDRESS = "0x28C6c06298d514Db089934071355E5743bf21d60";
// ----------------

async function checkBlockchainStatus() {
    try {
        // 1. Provider 생성:
        // Ethers.js의 Provider는 블록체인에 데이터를 '읽는' 역할을 함
        // RPC URL을 사용하여 원격 노드에 연결함

        //ethers.providers.JsonRpcProvider(URL) -> Provider 생성 ->  블록체인
        //노드에 연결하는 읽기 전용 인터페이스를 만듬, 모든 데이터 조회는
        //Provider를 통해 이루어짐.
        const provider = new ethers.providers.JsonRpcProvider(RPC_URL);

        console.log("   Web3 Provider 연결 성공");

        // --- 2. 블록체인 상태 조회 ---

        // A. 현재 블록 번호 조회
        //provider.getBlockNumber()  --> 상태 조회 --> 현재 블록체인에 추가된
        //가장 최신 블록의 번호를 조회하는 RPC 호출을 보냄(읽기 기능)
        const blockNumber = await provider.getBlockNumber();
        console.log(`\n현재 블록 번호: ${blockNumber}`);
```

```
        // B. 계정 잔액 조회
        // 잔액은 'Wei' 단위(이더리움의 가장 작은 단위)로 반환됨
        //provider.getBalance(주소) / 계정 데이터 조회 / 특정 주소에 저장된
        //ETH 잔액을 조회(이 값은 Wei 단위(1 ETH = 1018  Wei)로 반환됨)
        const balanceWei = await provider.getBalance(TARGET_ADDRESS);

        // Wei 단위를 사람이 읽기 쉬운 Ether 단위로 변환함
        //ethers.utils.formatEther(Wei) / 단위 변환 / 블록체인에서 받은 Wei 단위를
        //ETH 단위로 변환하여 사용자에게 보기 쉽게 만들어 줌
        const balanceEth = ethers.utils.formatEther(balanceWei);

        console.log(`\n--- 계정 잔액 조회 결과 ---`);
        console.log(`대상 주소: ${TARGET_ADDRESS}`);
        console.log(`잔액 (Wei): ${balanceWei.toString()}`);
        console.log(`잔액 (ETH): ${balanceEth}`);

    } catch (error) {
        console.error("    Web3 연결 또는 조회 중 오류 발생:", error);
    }
}

// 함수 실행
checkBlockchainStatus();
```

- 3) 실행 방법

* 가장 최근에 Ethers.js를 사용한 Web3 예제 코드를 실행하는 가장 간단하고 보편적인 방법 두 가지를 소개

* 웹 브라우저 (가장 간단한 방법)

  - 웹 개발 환경이 필요 없이, HTML 파일 하나만 있으면 됨
  - Ethers.js 라이브러리를 CDN(Content Delivery Network)을 통해 로드함
  - 실행 코드 (web3_test.html) - web3_test.html 파일로 저장한 후,
    웹 브라우저(크롬, 파이어폭스 등)에서 파일을 직접 열어봄 --> 페이지가 로드되면
    개발자 도구 (일반적으로 F12 키 또는 마우스 오른쪽 클릭 → 검사 →
    Console 탭)를 열어 결과를 확인

HTML

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>Ethers.js Web3 간단 테스트</title>
</head>
<body>
    <h1>Web3 블록체인 조회 테스트</h1>
    <p>결과는 브라우저의 개발자 도구 (Console)에서 확인하세요.</p>

    <script src="https://cdn.ethers.io/lib/ethers-5.7.2.umd.min.js"
type="application/javascript"></script>

    <script>
        // --- 설정 값 (반드시 수정하세요!) ---
        // 1. Infura 또는 Alchemy에서 제공하는 무료 RPC URL로 대체하세요.
        //    (예: Infura의 Sepolia 테스트넷 또는 Mainnet URL)
        const RPC_URL = "YOUR_INFURA_OR_ALCHEMY_RPC_URL";

        // 2. 조회할 대상 주소
        const TARGET_ADDRESS = "0x28C6c06298d5114Db089934071355E5743bf21d60";
        // ------------------------------------

        async function checkBlockchainStatus() {
            // ethers.js가 CDN을 통해 전역 객체로 로드되었는지 확인
            if (typeof ethers === 'undefined') {
                console.error("    Ethers.js 라이브러리가 로드되지 않았습니다.");
                return;
            }
```

```
            try {
                // 1. Provider 생성: 원격 노드에 연결
                const provider = new ethers.providers.JsonRpcProvider(RPC_URL);

                console.log("    Web3 Provider 연결 성공");

                // A. 현재 블록 번호 조회
                const blockNumber = await provider.getBlockNumber();
                console.log(`\n현재 블록 번호: ${blockNumber}`);

                // B. 계정 잔액 조회
                const balanceWei = await provider.getBalance(TARGET_ADDRESS);
                const balanceEth = ethers.utils.formatEther(balanceWei);

                console.log(`\n--- 계정 잔액 조회 결과 ---`);
                console.log(`대상 주소: ${TARGET_ADDRESS}`);
                console.log(`잔액 (ETH): ${balanceEth}`);

            } catch (error) {
                console.error("    Web3 연결 또는 조회 중 오류 발생:", error);
            }
        }

        // 페이지 로드 후 함수 실행
        window.onload = checkBlockchainStatus;
    </script>
</body>
</html>
```

* Node.js 환경 (표준 개발 방식)

- Node.js 환경에서는 라이브러리를 직접 설치하여 실행(이는 실제 개발에서 가장 많이 사용되는 방식)

- 환경 설정
    * Node.js 설치 - Node.js가 설치되어 있어야 함
    * 프로젝트 초기화 - 새 폴더를 만들고 터미널에서 초기화함

```
mkdir web3-example
cd web3-example
npm init -y
```

    * Ethers.js 설치 - Ethers 라이브러리를 설치

```
npm install ethers
```

- 실행 코드 : index.js 파일로 저장

```
// Node.js 환경에서 ethers 라이브러리를 불러옴
const { ethers } = require("ethers");

// --- 설정 값 (반드시 수정하세요!) ---
const RPC_URL = "YOUR_INFURA_OR_ALCHEMY_RPC_URL";
const TARGET_ADDRESS = "0x28C6c06298d5114Db089934071355E5743bf21d60";
// ------------------------------------

async function checkBlockchainStatus() {
    try {
        const provider = new ethers.providers.JsonRpcProvider(RPC_URL);

        console.log("    Web3 Provider 연결 성공");

        const blockNumber = await provider.getBlockNumber();
        console.log(`\n현재 블록 번호: ${blockNumber}`);

        const balanceWei = await provider.getBalance(TARGET_ADDRESS);
        const balanceEth = ethers.utils.formatEther(balanceWei);

        console.log(`\n--- 계정 잔액 조회 결과 ---`);
        console.log(`대상 주소: ${TARGET_ADDRESS}`);
        console.log(`잔액 (ETH): ${balanceEth}`);

    } catch (error) {
        console.error("    Web3 연결 또는 조회 중 오류 발생:", error.message);
    }
}

checkBlockchainStatus();
```

- 실행 방법
    * 터미널에서 코드가 저장된 폴더로 이동
    * 다음 명령어를 실행
        - node index.js
    * 터미널에 블록 번호와 계정 잔액이 바로 출력되는 것을 확인할 수 있음