


Week 2: ERC20 직접 구현하기, 프론트엔드 연동해보기

Jin Hyung Park | DSRV

각자의 과제 구현 사항 발표하기

- 발표 순서: 유승우 > 신경섭 > 유태성 > 조소연
- Pull Request를 보며 서로 코드에 대한 토론을 진행합니다.


EIP에는 무엇이 있을까?



Standard Track EIP

Core
Networking
Interface
ERC

ETH 모든 것이라고 생각하면 될 것 같다.
코어 부분 부터 인터페이스 (UI가 아니다.)까지



Informational EIP

ETH 디자인 문제에 대한 설명
새로운 기능 제안 X

정보용 EIP는 무시하거나,
조언을 따를 수 있다.



Meta EIP

이더리움의 프로세스에 대한 설명이나 방법
제안 / Meta EIP 는 제안 될 수 있지만
이더리움 코드에 적용할 수 없다. / 이더리움
프로토콜 이외 영역에도 적용 된다. /
Informational EIP 처럼 무시할 수는 없다. /
Ethereum IDE & 환경 변경등이 포함 된다.

류기혁 (+82)-10-9271-1197

EIP 관련 내용
더 살펴보러 가기

EIP-20, 토큰의 규격을 정의한 인터페이스

- 마치 `duck-type` 처럼, 아래 함수와 이벤트가 모두 구현되어 있으면 ERC20 규격이라고 한다.
 - `totalSupply`
 - `balanceOf(address)`
 - `transfer`, `transferFrom`
 - `approve`, `allowance`
 - Event: `Transfer`, `Approval`
- [EIP-20 제안 링크](#)

OpenZeppelin이란 무엇일까?

스마트 컨트랙트 Auditing이 끝난, ERC 토큰의 (사실상) 표준 구현체

EIP 표준에 대해 본인이 알아서 구현해도 되고,

이미 있는 것을 가져다 써도 되지만 주로 OpenZeppelin을 사용



오픈제플린 인터페이스

ERC20의 구현체, 직접 살펴보기

- 우리는 OpenZeppelin의 구현체를 직접 뜯어보는 시간을 가져보려고 합니다.
- OpenZeppelin 구현체
 - extension 1) **cap**이 있다면?
 - extension 2) **burn**할 수 있다면?
- **maple-labs** 구현체
- Approve 개념을 이해하는 것이 제일 중요합니다.

ERC20의 구현체, 직접 따라 쳐보기

- 직접 손으로 코딩해보면서 Solidity 문법을 익혀보세요.
- [예시 코드](#)
- 위에서 따라 친 예시 코드를 Rinkeby 테스트넷에 배포해주세요.

Verification?

왜 해야 할까?

- 이더리움 컨트랙트에 업로드되는 것은 바이트 코드이다. 실제 내가 github 저장소를 공개했다고 하더라도, 정말 github 저장소에 올라온 코드가 컨트랙트에 올라갔는지 확인하기 어렵다.
- (중앙화된) etherscan이 컨트랙트를 불러올 때 바이트코드도 같이 불러온다. 이 때, 개발자는 본인이 컨트랙트를 배포했을 때 만든 바이트코드를 etherscan 서버에 업로드 한다.
- etherscan은 개발자가 제출한 바이트코드와 이더리움 컨트랙트에 올라간 바이트코드와 완전히 일치하면 'verified' 를 찍어준다.
- [BAYC Contract](#) [USDT Contract](#)

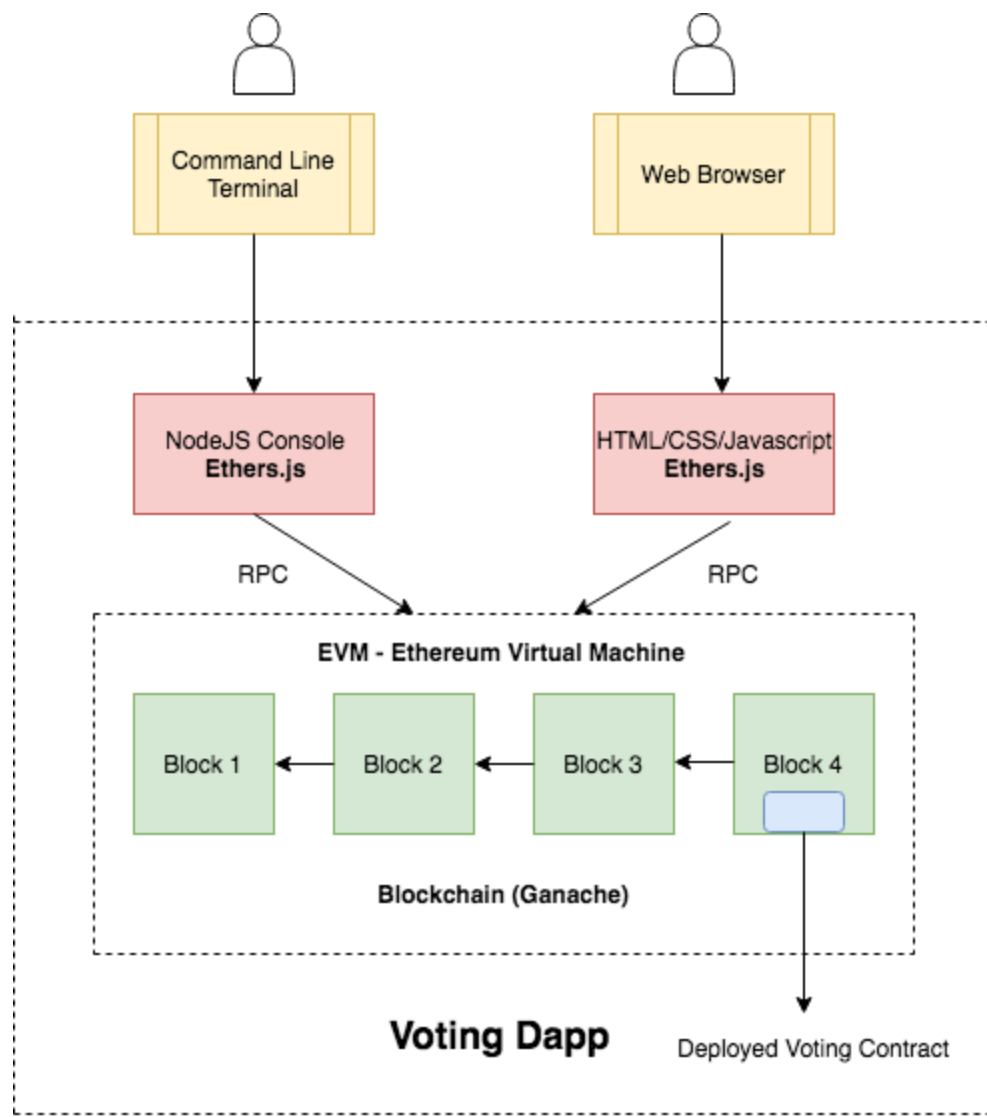
어떻게 하는 걸까? 실습해보자.

- `hardhat-etherscan` 프레임워크를 사용한다.

프론트엔드-이더리움 컨트랙트 연결하기

어떤 라이브러리가 필요할까?

- 우리는 이더리움 지갑과 연결해 지갑 API에 트랜잭션 요청 등을 보내는 라이브러리가 필요하다.
- ethers.js 또는 web3.js 를 활용하여 메타마스크 지갑에 연결한다.
- web3.js에는 메타마스크와 같은 회사(Infura) 이므로, Provider 사용처가 결정된다.
- 만약 AllThatNode 같은 다른 Provider를 사용하고 싶다면, ethers.js를 사용해야 한다.



간단한 나만의 Token Transfer 사이트 구현 미션

기술 스택

- React.js 를 사용하여 다음의 요구 사항을 구현한다.
- [완성 링크](#)

My ERC20 Token Transfer Web App

Long Live Smiling Leo, Long Live DSRV!

요구 사항

- Step 1: 현재 브라우저에 메타마스크 지갑이 설치되었는지 확인할 수 있다.
- Step 2: 메타마스크 지갑이 설치되어 있다면, 웹 사이트에 메타마스크 지갑을 연결할 수 있다.
- Step 3: Rinkeby Network에 연결되어 있지 않다면, 해당 네트워크로 연결하라는 메시지를 표시한다.
- Step 4: 토큰을 보낼 이더리움 지갑 주소, 토큰을 보낼 갯수를 입력할 수 있다.
- Step 5: 프론트엔드의 `Transfer` 버튼을 누르면 ERC20 컨트랙트의 `transfer` 함수가 실행된다.
- Step 6: Solidity의 `event` 후크 기능을 활용하여, 토큰 전송이 완료된 후 `Transfer event`가 성공적으로 `emit` 되었을 때 프론트엔드에서 토큰 전송 완료 사실을 사용자에게 노출한다.
- Step 7: `fleek.co` 에서 완성된 정적 페이지를 배포한다.