Ethers tutorial - Block Explorer

Проект посвящен разработке собственного обозревателя блоков сети Ethereum средствами библиотеки `ethers.js` и `React`.

Установка приложения

Для фронтенда традиционно используется библиотека `React` на платформе node.js.

В качестве IDE рекомендуется VSCode.

Создайте новое приложение с помощью vite и перейдите в рабочий каталог:

```
npm create vite@latest dapp -- --template react
cd dapp
```

В файле `package.json` в раздел `dependencies` добавьте зависимость `"ethers": "6.13"`

Установите зависимости и запустите приложение

```
npm install
npm run dev
```

Если в браузере Chrome блокируется localhost по протоколу http, можно отключить правило безопасности для локальной разработки. Откройте новую вкладку с адресом `chrome://net-internals/#hsts`.
В разделе `Delete domain security policies` введите `localhost` и нажмите кнопку `Delete`.

Настройте стили приложения. Для этого в папке src замените файл App.css и очистите содержимое index.css

1. Настройка провайдера

Библиотека `ethers` - это компактный и функциональный инструмент для взаимодействия с блокчейном ethereum. Класс `Provider` является базовой абстракцией для чтения информации о состоянии аккаунтов, событий и запуска публичных идемпотентных методов смарт-контрактов.

Замените содержимое файла `src/App.jsx`

```
import { ethers } from "ethers";
import './App.css'

const providerUrl = 'http://77.51.210.148:48545/';
const provider = new ethers.JsonRpcProvider(providerUrl);
const network = await provider.getNetwork();

function App() {
  console.log(network);
  return (
    <>
    </>
  )
}

export default App
```

В окне браузера откройте панель разработчика (клавиша F12), в консоли должен появиться объект

```
_Network {
chainId: 11155111n
name: "sepolia"
}
```

2. Добавьте функцию BalanceReader

В качестве основы используем компонент `Wallet` из проекта `ecdsa-node`.

В папке src добавьте BalanceReader.jsx со следующим содержимым:

```
import { ethers } from "ethers";
import { useState } from 'react';

function BalanceReader({ provider }) {
  const [address, setAddress] = useState("");
  const [balance, setBalance] = useState(0);

  async function onChange(evt) {
    const address = evt.target.value;
    console.log("address: ", address);
    setAddress(address);
    try {
      const balance = await provider.getBalance(address);
      console.log("balance: ", balance);
      const newBalance = ethers.formatEther(balance);
      setBalance(newBalance);
    } catch(err){
      console.log(err);
    }
  }

  return (
    <div className="container wallet">
      <h1>Balance Reader</h1>
      <label>
        Address:
        <input placeholder="Type any address" value={address} onChange={onChange}></input>
      </label>
      <div className="balance">Balance: {balance} ETH</div>
    </div>
  );
}

export default BalanceReader;
```

Подключите компонент BalanceReader в приложении App.jsx

Проверьте состояние баланса, например аккаунта ‘0xeE2d4c1EF974a67E16caD4B19F209697694B4010’.

3. Добавьте функцию обозревателя блоков

Библиотека `ethers.js` через класс `Provider` позволяет читать состояние сети Ethereum, а именно: * количество блоков в блокчейне * содержимое транзакций в блоке * аккаунты отправителя, получателя

и количество монет в транзакции * и многое другое.

Существует множество готовых обозревателей блоков. Наиболее популярный обозреватель блоков сети Ethereum - Etherscan. Проверьте как он работает, это поможет при самостоятельной реализации функций собственного обозревателя блоков.

Мы разработаем компонент, который позволит: - Вывести список из 3 последних блоков. - По клику на блоке вывести его содержимое - заголовки транзакций. - По клику на заголовке транзакции вывести детали транзакции.

Создайте компонент `Block.jsx` и наполните его содержимое:

```
import React, { useState, useEffect } from 'react';

function Block({blocknum, provider}) {
    const [open, setOpen] = useState(false);
    const [block, setBlock] = useState();

    function toggle(){ setOpen(!open); }

    useEffect(() => {
        async function getBlock() {
          const blockInfo = await provider.getBlock(blocknum);
          console.log(blockInfo);
          setBlock(blockInfo);
        }
        if (!block){ getBlock(); }
      }
    );

    return (
        <div>
            <hr/>
            <button onClick={toggle}>Block # {blocknum}</button>
            {open &&
            <div>
                <p>Block hash: {block.hash}</p>
                <p>Parent hash: {block.parentHash} </p>
            </div>}

        </div>
    )
}

export default Block;
```

Этот компонент позволяет отобразить хеш текущего и предыдущего блока по его номеру. Вы можете добавить любую информацию о блоке, которая доступна через методы класса Block.

Номер последнего текущего блока берется из сети через вызов метода `provider.getBlockNumber()`. Добавьте компонент `BlockExplorer.jsx` со следующим содержимым:

```
import { useState, useEffect } from "react";
import Block from "./Block";

function BlockExplorer({provider}) {
    const [blockNumber, setBlockNumber] = useState(0);

    useEffect(() => {
```

```
        async function getBlockNumber() {
          const latestBlockNum = await provider.getBlockNumber();
          console.log("latest block: ", latestBlockNum);
          setBlockNumber(latestBlockNum);
        }
        getBlockNumber();
      }
    );
    return (
        <div className="container">
            <h1>Block Explorer</h1>
            <Block
                blocknum = {blockNumber}
                provider = {provider}
            />
            <Block
                blocknum = {blockNumber - 1}
                provider = {provider}
            />
            <Block
                blocknum = {blockNumber - 2}
                provider = {provider}
            />
        </div>
    )
}

export default BlockExplorer;
```

Подключите компонент `BlockExplorer` в приложении `App.jsx`.

4. Добавьте функцию VendingMachine

Класс `Contract` позволяет взаимодействовать со смарт-контрактами, размещенными в сети Ethereum. Список верифицированных смарт-контрактов можно найти в обозревателе sepolia.etherscan Взаимодействие со смарт-контрактом идет через бинарный интерфейс (ABI, application binary interface), через библиотеку ethers.

Добавьте компонент `VendingMachine.jsx` со следующим содержимым:

```
import { useState, useEffect } from "react";
import { ethers } from "ethers";

const vmContractAddress = "0x0D17C16BB7bD73Ef868cD452A1B10D069685249f";
const abi = [
    "function symbol() view returns (string)",
    "function getVendingMachineBalance() view returns (uint)",
    "function balanceOf(address addr) view returns (uint)",
    "function purchase(uint amount) payable returns ()"
  ];


function VendingMachine({ provider }) {
    const [writableContract, setWritableContract] = useState();
    const [address, setAddress] = useState("");
    const [symbol, setSymbol] = useState("");
    const [cupsInMachine, setCupsInMachine] = useState(0);
```

```
    const [purchaseCups, setPurchaseCups] = useState("");
    const [accountCups, setAccounCups] = useState(0);

    useEffect(() => {
        if (!symbol)  { updateVendingMachineState() }
        if (writableContract) { updateAccountCups() }
      }, [symbol, writableContract] );

    const setValue = (setter) => (evt) => setter(evt.target.value);

    async function updateVendingMachineState() {
        const readOnlyContract = new ethers.Contract(vmContractAddress, abi, provider);
        console.log(readOnlyContract);

        const symbol = await readOnlyContract.symbol();
        setSymbol(symbol);
        console.log(symbol);

        const cupsInMachine = await readOnlyContract.getVendingMachineBalance();
        setCupsInMachine(cupsInMachine.toString());
        console.log(cupsInMachine);
    }

    return (
        <>
        <div className="container">
            <h1>Vending Machine</h1>
            <div className="balance">TOTAL: {cupsInMachine} {symbol} </div>

        </div>
        </>

    );
}

export default VendingMachine;
```

Через публичного провайдера можно подключиться к смарт-контракту и прочитать его состояние:
`const readOnlyContract = new ethers.Contract(vmContractAddress, abi, provider);`

Для изменения состояния потребуется подключение кошелька, например, MetaMask. Добавьте функции для покупки виртуальных пирожных.

```
...

async function connectWallet(evt){
    evt.preventDefault();
    try {
        const walletProvider = new ethers.BrowserProvider(window.ethereum);
        console.log(walletProvider);

        const signer = await walletProvider.getSigner();
        console.log("Wallet signer: ", signer);

        const address = signer.address;
        setAddress(address)
        console.log("Wallet address: ", address);
```

```
        const writableContract = new ethers.Contract(vmContractAddress, abi, signer);
        setWritableContract(writableContract)
        console.log("Writable contract: ", writableContract);
    } catch (exeption){
        alert(exeption);
    }
}

async function updateAccountCups(){
    try {
        const accountCups = await writableContract.balanceOf(address);
        setAccounCups(accountCups.toString());
        console.log(accountCups);
    } catch (exeption){
        alert(exeption);
    }
}

async function purchaseCupcakes(evt) {
    evt.preventDefault();
    try {
        console.log(writableContract);
        const tx = await writableContract.purchase(
            purchaseCups,
            { value: ethers.parseUnits(purchaseCups, 'gwei') }
        );
        await tx.wait();
        updateAccountCups();
        updateVendingMachineState();
    } catch (exeption) {
        alert(exeption);
    }
}

...

<form>
    <input
        type="submit"
        className="button"
        value="Connect Wallet"
        onClick={ connectWallet }
        />
    <label>Purchase Cupcakes
    <input
        placeholder="1, 2, 3..."
        value={purchaseCups}
        onChange={ setValue(setPurchaseCups) }
        />
    </label>
    <input
        type="submit"
        className="button"
        value="Purchase"
        onClick={purchaseCupcakes}
        disabled={ !writableContract } />
```

```
</form>
<div className="balance">YOU HAVE: {accountCups} {symbol} </div>
```

Заключение

Мы познакомились с основными возможностями библиотеки `ethers.js`. В качестве домашнего задания выберите любой верифицированный смарт-контракт в тестовой сети sepolia. Напишите компонент для взаимодействия с ним и отображения результата вызова публичных методов.