Основи Ethereum DApp

Вступ до Децентралізованих Додатків

Запорізький національний університет

2024

Що таке DApp?

- **DApp (Децентралізований додаток)** – програмне забезпечення, що працює на децентралізованій мережі блокчейн.
- **Особливості:**
  - Відкрите програмне забезпечення
  - Децентралізована робота
  - Токенізована економіка
  - Смарт-контракти

DApp vs Класичні Веб-Додатки

Характеристика	Класичні Веб-Додатки	DApp
Централізований сервер	Так	Ні
Прозорість	Обмежена	Висока
Безпека	Залежить від серверів	Вища завдяки блокчейну
Контроль даних	Власник додатку	Користувачі
Масштабованість	Обмежена серверними ресурсами	Залежить від мережі блокчейн

Основні Компоненти DApp

1. **Смарт-контракти**
   - Логіка додатку на блокчейні
2. **Фронтенд**
   - Інтерфейс користувача
3. **Бекенд (за потреби)**
   - Додаткові сервіси
4. **Блокчейн Мережа**
   - Ethereum або інша платформа
5. **Криптогаманці**
   - MetaMask, WalletConnect тощо

Бібліотеки для Взаємодії з Блокчейном

Web3.js

- Офіційна бібліотека від Ethereum
- Підтримує всі функції Ethereum
- Поширена та добре документована

Ethers.js

- Легка та сучасна альтернатива Web3.js
- Модульна структура
- Краще підтримує TypeScript

Приклади Використання Ethers.js

```
// Імпорт бібліотеки
const { ethers } = require('ethers');

// Підключення до провайдера
const provider =
  new ethers.providers.InfuraProvider('homestead', 'YOUR_INFURA_PROJECT_ID');

// Створення гаманець
const wallet = ethers.Wallet.createRandom();

// Отримання балансу
provider.getBalance(wallet.address).then(balance => {
  console.log(`Баланс: ${ethers.utils.formatEther(balance)} ETH`);
});
```

Процес Розробки DApp

1. **Планування**
    - Визначення цілей та функціоналу
2. **Розробка Смарт-Контрактів**
    - Написання та тестування контрактів
3. **Розгортання на Блокчейні**
    - Використання інструментів як Truffle або Hardhat
4. **Розробка Фронтенду**
    - Інтеграція з контрактами через бібліотеки
5. **Тестування та Валідація**
    - Перевірка безпеки та функціоналу
6. **Запуск та Моніторинг**
    - Впровадження та підтримка додатку

Інтеграція Фронтенду з Блокчейном

Приклад на JavaScript

```
<!DOCTYPE html>
<html>
<head>
  <title>Simple DApp</title>
  <script src="https://unpkg.com/ethers@5.0.8/dist/ethers.umd.min.js"></script>
</head>
<body>
  <button id="connect">Підключити Метамаск</button>
  <script>
    document.getElementById('connect').onclick = async () => {
      if (typeof window.ethereum !== 'undefined') {
        await ethereum.request({ method: 'eth_requestAccounts' });
        const provider = new ethers.providers.Web3Provider(window.ethereum);
        const signer = provider.getSigner();
        const address = await signer.getAddress();
        alert('Підключено: ' + address);
      } else {
        alert('Метамаск не встановлено');
      }
    };
  </script>
</body>
</html>
```

Приклад Розробки Простого DApp на Основі ERC20 Гаманця

Основні Функції

1. **Підключення Криптогаманця**
2. **Перегляд Балансу ERC20 Токенів**
3. **Відправлення Токенів**
4. **Перегляд Транзакцій**

Підключення Криптогаманця

```
async function connectWallet() {
  if (window.ethereum) {
    try {
      await ethereum.request({ method: 'eth_requestAccounts' });
      const provider = new ethers.providers.Web3Provider(window.ethereum);
      const signer = provider.getSigner();
      const address = await signer.getAddress();
      console.log('Підключено:', address);
      return { provider, signer, address };
    } catch (error) {
      console.error('Помилка підключення:', error);
    }
  } else {
    alert('Метамаск не встановлено');
  }
}
```

Перегляд Балансу ERC20 Токенів

```
const ERC20_ABI = [
  "function balanceOf(address owner) view returns (uint256)",
  "function decimals() view returns (uint8)"
];

async function getTokenBalance(provider, address, tokenAddress) {
  const contract = new ethers.Contract(tokenAddress, ERC20_ABI, provider);
  const balance = await contract.balanceOf(address);
  const decimals = await contract.decimals();
  return ethers.utils.formatUnits(balance, decimals);
}

// Використання
const tokenAddress = '0xYourTokenAddress';
getTokenBalance(provider, address, tokenAddress).then(balance => {
  console.log('Баланс токенів:', balance);
});
```

Відправлення Токенів

```
async function sendTokens(signer, tokenAddress, to, amount) {
  const contract = new ethers.Contract(tokenAddress, ERC20_ABI, signer);
  const decimals = await contract.decimals();
  const tx =
    await contract.transfer(to, ethers.utils.parseUnits(amount, decimals));
  await tx.wait();
  console.log('Токени відправлено:', tx.hash);
}

// Використання
const recipient = '0xRecipientAddress';
const amount = '10.0';
sendTokens(signer, tokenAddress, recipient, amount);
```

Перегляд Транзакцій

```
async function getTransactions(provider, address) {
  const history = await provider.getHistory(address);
  history.forEach(tx => {
    console.log(`Hash: ${tx.hash}, To: ${tx.to}`,
      `Value: ${ethers.utils.formatEther(tx.value)} ETH`);
  });
}

// Використання
getTransactions(provider, address);
```

Висновок

- DApps пропонують децентралізовану альтернативу класичним веб-додаткам.
- Використання бібліотек як Web3.js та Ethers.js спрощує інтеграцію з блокчейном.
- Розробка DApp включає створення смарт-контрактів, фронтенду та їхню інтеграцію.
- Практичні приклади допомагають зрозуміти процес створення реальних додатків на основі Ethereum.