Building a dApp on Sui

Understand the process of building dApps on Sui, including smart contract development, front-end integration, and wallet connectivity.

Agenda

1. Connecting Front-End to Sui Smart Contracts
2. Wallet Integration
3. Building an NFT Minting App
4. Building a Basic DeFi App
5. Practical Exercises

Connecting Front-End to Sui Smart Contracts

Sui TypeScript SDK

- Low-level blockchain interaction
- Call smart contract functions

dApp Kit for React

- SuiClientProvider for network
- WalletProvider for wallet management
- ConnectButton for user login

Example Usage

Wrap app with providers; use ConnectButton; get wallet address

Wallet Integration

Wallet Role

Sign transactions, manage user assets

>

dApp Kit Usage

WalletProvider + ConnectButton for seamless connection

>

Developer Tip

Access wallet info with useCurrentAccount hook

Building an NFT Minting App

Smart Contract

✓ Define NFT struct with key and store attributes

Mint Function

✓ Transfer newly created NFT to sender

Front-End

✓ Use SDK/dApp Kit to call mint with metadata

*Example code available for Move and JavaScript implementations.

Building a Basic DeFi App

Lending Pool Smart Contract

- Define lending pool struct
- Deposit, borrow, and repay functions
- Interest handling included

Front-End Interface

- Deposit SUI page
- Borrow using collateral
- Repay loan page

SDK Integration

- Invoke contract functions from UI

Optimizing Front-End Integration

React Hooks

Use hooks for state and account management

Provider Wrapping

Wrap app with SuiClient and Wallet providers

User Experience

Show wallet connection status & error handling

Practical Exercises

NFT Minting App

✓ Move code for NFT struct, Typescript for front-end mint call

Building a Basic DeFi App

✓ Using SDK: Call smart contract functions from front-end for deposits, borrows, repays

(https://mirror.xyz/greymate.eth/_P2NXvVoh9wISj_mqgavDymIERCnW2DgC1gigJNrmUI)

(https://docs.sui.io/guides/developer/app-examples)

```move
module example::marketplace {
    use sui::object::{Self, UID};
    use sui::transfer;
    use sui::tx_context::{Self, TxContext};
    use example::nft::MyNFT;

    struct Listing has key, store {
        id: UID,
        nft: MyNFT,
        price: u64,
        seller: address,
    }

    public fun list_for_sale(nft: MyNFT, price: u64, ctx: &mut TxContext): Listing {
        Listing {
            id: object::new(ctx),
            nft,
            price,
            seller: tx_context::sender(ctx),
        }
    }

    public fun buy(listing: Listing, ctx: &mut TxContext) {
        let buyer = tx_context::sender(ctx);
        assert!(buyer != listing.seller, 101); // Buyer cannot be the seller
        transfer::public_transfer(listing.nft, buyer);
        // Transfer payment logic would go here (e.g., using Sui's coin module)
    }
}
```

Thank You.