Building a Frontend

Introduction

In this tutorial, we'll demonstrate how to set up a frontend project using React. You'll learn how to connect to a Sei wallet and interact with CosmWasm smart contracts deployed on Sei.

In this section, we'll use the@sei-js(opens in a new tab) library to build a React app that interacts with a CosmWasm contract.

Requirements

Before starting, ensure you have:

- Node.js & NPM installed
- A Sei wallet extension, such as:* Compass(opens in a new tab)
-
    - Fin(opens in a new tab)

Creating a React Project

If you're starting a project from scratch, we recommend using the TypeScript template from Vite for easier development and debugging.

npm

create

vite@latest

my-counter-frontend

--

--template

react-ts This command creates a new folder with a React project using TypeScript. Openmy-counter-frontend in your favorite IDE.

The rest of this tutorial will be in TypeScript. If you're not using TypeScript, you can easily adjust by removing the types.

Installing Dependencies

From the terminal at the root of your project, install the required dependencies:@sei-js/core and@sei-js/react .

npm

install

@sei-js/core

@sei-js/react @sei-js contains NPM libraries for writing applications that interact with Sei. Learn morehere(opens in a new tab) .

Wrapping App in SeiWalletProvider

Replace the code in yourApp.tsx file with the following to set up aSeiWalletProvider context, define your chain info, and set connection URLs.

import { SeiWalletProvider } from

"@sei-js/react" ; import

"./App.css" ; import Home from

"./Home.tsx" ;

function

App () { return (// Set up SeiWalletProvider for easy wallet connection and to use hooks in @sei-js/react < SeiWalletProvider chainConfiguration = {{ chainId :

"atlantic-2" , restUrl :

"https://rest.atlantic-2.seinetwork.io" , rpcUrl :

"https://rpc.atlantic-2.seinetwork.io" , }} wallets = {["compass" ,

"fin"]}

    < Home /> </ SeiWalletProvider

    ); }

export

default App;

Detailed outline ofApp.tsx

Wallet Provider Setup

- ImportsSeiWalletProvider
- from the@sei-js/react
- package.
- WrapsApp
- withSeiWalletProvider
- for wallet context.

Chain Configuration

- Specifies thechainConfiguration
- prop inSeiWalletProvider
- to set up the blockchain network:* chainId
-
    - : Identifies the Sei network (in this case,atlantic-2
-
    -).
-
    - restUrl
-
    - : REST URL for the Sei (You may need your own provider).
-
    - rpcUrl
-
    - : RPC URL for the Sei (You may need your own provider).

Supported Wallets

- Sets up supported wallet types ('compass'
- ,'fin'
- ,'leap'
- ,'keplr'
-) in the wallets prop.

Adding Home Component

Create a file namedHome.tsx in yoursrc directory with the following code:

import { useCallback , useEffect , useState } from

"react" ; import { useCosmWasmClient , useSigningCosmWasmClient , useWallet , WalletConnectButton , } from

"@sei-js/react" ;

// atlantic-2 example contract const

CONTRACT_ADDRESS

= "sei1kaq032fm8nz9ct58c8s7chgds5zz68pvy4npwx3r9a0kq9ygw0sscuas7c" ;

```
function

Home () { const [ count ,

setCount ] =

useState < number

|

undefined

    (); const [ error ,

setError ] =

useState < string

    ( "" ); const [ isIncrementing ,

setIsIncrementing ] =

useState < boolean

    ( false );

// Helpful hook for getting the currently connected wallet and chain info const { connectedWallet ,

accounts } =

useWallet ();

// For querying cosmwasm smart contracts const { cosmWasmClient: queryClient } =

useCosmWasmClient ();

// For executing messages on cosmwasm smart contracts const { signingCosmWasmClient: signingClient } =

useSigningCosmWasmClient ();

const

fetchCount

=

useCallback ( async () => { const

response

=

await

queryClient ?.queryContractSmart ( CONTRACT_ADDRESS , { get_count : {} , }); return

response ?.count; } , [queryClient]);

useEffect (() => { fetchCount () .then (setCount); } , [connectedWallet , fetchCount]);

const

incrementCounter

=

async () => { setIsIncrementing ( true ); try { const

senderAddress

= accounts[ 0 ].address;

// Build message content const
```

msg

= { increment : {} };

// Define gas price and limit const

fee

= { amount : [{ amount :

"20000" , denom :

"usei" }] , gas :

"200000" , };

// Call smart contract execute msg await

signingClient ?.execute (senderAddress ,

CONTRACT_ADDRESS , msg , fee);

// Updates the counter state again const

updatedCount

=

await

fetchCount (); setCount (updatedCount);

setIsIncrementing (false); setError (""); } catch (error) { if (error instanceof

Error) { setError (error .message); } else { setError ("unknown error"); } setIsIncrementing (false); } };

// Helpful component for wallet connection if (! connectedWallet) return < WalletConnectButton />;

return (< div

    < h1 Count is: {count ? count :

"---" }</ h1

    < button

disabled

{isIncrementing} onClick = {incrementCounter}> {isIncrementing ?

"incrementing..."

:

"increment" } </ button

    {error && < p

style

{{ color :

"red" }}>{error}</ p

    } </ div

    ); }

export

default Home; We deployed a counter contract on theatlantic-2 testnet. Contract address:sei1kaq032fm8nz9ct58c8s7chgds5zz68pvy4npwx3r9a0kq9ygw0sscuas7c . Learn more about this contracthere(opens in a new tab) .

Detailed outline ofHome.tsx

Render Logic

- ShowsWalletConnectButton
- if no wallet is connected.
- Otherwise, displays the counter value and an "Increment" button.
- Disables the button when a transaction is pending.
- Catches and displays error messages, if any.

Wallet and Client Setup

- UtilizesuseWallet
- hook from@sei-js/react
- to get the connected wallet and accounts.
- UsesuseCosmWasmClient
- anduseSigningCosmWasmClient
- hooks from@sei-js/react
- to instantiatequeryClient
- andsigningClient
- . The signing client can be used for querying, but it is not recommended to use a signing client when only a query client is needed.

Fetching Counter Value

- fetchCount
- function usesqueryClient.queryContractSmart
- to query the smart contract with the query{ get_count: {} }
- .

Incrementing Counter

- incrementCounter
- function:* ExtractssenderAddress
-
    - from the connected wallet's accounts.
-
    - Builds a message with{ increment: {} }
-
    - to be sent to the smart contract.
-
    - Specifies the transaction fee with gas price and gas limit.
-
    - CallssigningClient.execute
-
    - to execute the contract with the built message and fee.

By leveraging@sei-js and its hooks, the component provides a way to connect to a wallet, query a CosmWasm smart contract to get the current counter state, and increment it with an execute message.

Running the App

Runnpm run dev and navigate tohttp://localhost:5173/(opens in a new tab) to view your application.

If this is your first time using Sei, you might see an error which says your account does not exist on chain. You can claim some testnet tokens for development purposes from a faucethere .

Conclusion

Congratulations on creating a website for querying and executing a smart contract on Sei! Explore more possibilities with your frontend at our@sei-js repo(opens in a new tab) .

Last updated onMarch 12, 2024