README.md

**zkMe Widget Demo**

Example of zkMe widget integration.

**Environment Variables**

To run this project, you will need to add the following environment variables to your .env file

VITE\_APP\_APP\_ID Your AppId **(required)**

VITE\_APP\_API\_KEY Your Api key **(required)**

VITE\_APP\_DAPP\_ACCOUNT Your Dapp account **(required)**

VITE\_APP\_INFURA\_KEY Your infura key

**Choose your chain**

The project runs on the Ethereum Goerli testnet by default, if you want to test other chains, add the query parameter chainId=0x89 at the end of the URL, other optional values are 0x1b59, 0x13881, 0x82751, 0x2105, 0x34816d, atlantic-2.

**Recommended IDE Setup**

[VSCode](https://code.visualstudio.com/) + [Volar](https://marketplace.visualstudio.com/items?itemName=Vue.volar) (and disable Vetur) + [TypeScript Vue Plugin (Volar)](https://marketplace.visualstudio.com/items?itemName=Vue.vscode-typescript-vue-plugin).

**Type Support for .vue Imports in TS**

TypeScript cannot handle type information for .vue imports by default, so we replace the tsc CLI with vue-tsc for type checking. In editors, we need [TypeScript Vue Plugin (Volar)](https://marketplace.visualstudio.com/items?itemName=Vue.vscode-typescript-vue-plugin) to make the TypeScript language service aware of .vue types.

If the standalone TypeScript plugin doesn't feel fast enough to you, Volar has also implemented a [Take Over Mode](https://github.com/johnsoncodehk/volar/discussions/471#discussioncomment-1361669) that is more performant. You can enable it by the following steps:

1. Disable the built-in TypeScript Extension
   1. Run Extensions: Show Built-in Extensions from VSCode's command palette
   2. Find TypeScript and JavaScript Language Features, right click and select Disable (Workspace)
2. Reload the VSCode window by running Developer: Reload Window from the command palette.

**Customize configuration**

See [Vite Configuration Reference](https://vitejs.dev/config/).

**Project Setup**

pnpm install

**Compile and Hot-Reload for Development**

pnpm dev

**Type-Check, Compile and Minify for Production**

pnpm build

**Lint with**[**ESLint**](https://eslint.org/)

pnpm lint

index.html

<!DOCTYPE html>

<html lang=**"en"**>

<head>

<meta charset=**"UTF-8"**>

<link rel=**"icon"** href=**"/favicon.ico"**>

<meta name=**"viewport"** content=**"width=device-width, initial-scale=1.0, maximum-scale=1, viewport-fit=cover, user-scalable=0"**>

<title>**zkMe Pop-up Demo**</title>

</head>

<body>

<div id=**"app"**></div>

<script type=**"module"** src=**"/src/main.ts"**></script>

</body>

</html>

src/main.ts

**import** **{** createApp **}** **from** 'vue'

**import** App **from** './App.vue'

**import** router **from** './router'

**import** './assets/css/main.css'

**const** app **=** createApp**(**App**)**

app**.**use**(**router**)**

app**.**mount**(**'#app'**)**

src/App.vue

<script setup lang=**"ts"**>

</script>

<template>

<RouterView />

</template>

<style scoped>

</style>

src/router/index.ts

**import** **{** createRouter**,** createWebHistory **}** **from** 'vue-router'

**const** router **=** createRouter**({**

history**:** createWebHistory**(import.**meta**.**env**.**BASE\_URL**),**

scrollBehavior **()** **{**

**return** **{** top**:** 0 **}**

**},**

routes**:** **[**

**{**

path**:** '/'**,**

name**:** 'home'**,**

component**:** **()** **=>** **import(**'@/views/DappHome.vue'**),**

meta**:** **{**

noAuth**:** **true**

**}**

**},**

// {

// path: '/about',

// name: 'about',

// // route level code-splitting

// // this generates a separate chunk (About.[hash].js) for this route

// // which is lazy-loaded when the route is visited.

// component: () => import('../views/AboutView.vue')

// }

**]**

**})**

**export** **default** router

src/views/DappHome.vue

**<**script setup lang**=**"ts"**>**

**import** **{** useWallet**,** useChain**,** useVerify**,** useZkMe**,** useStake**,** useMchConf **}** **from** '@/composables'**;**

**import** **{** ZKME\_POPUP\_ORIGIN **}** **from** '@/utils/config'**;**

**import** **{** switchChain **}** **from** '@/utils/eth'**;**

**const** zkMePopupNode **=** shallowRef**<**HTMLIFrameElement**>()**

**const** lv **=** computed**(()** **=>** **{**

**const** **{** lv **=** '' **}:** **{** lv**?:** string **}** **=** useRoute**().**query

**return** lv

**})**

**const** **{**

connect**,**

connecting**,**

connectedAddress**,**

signer**,**

signingCosmWasmClient**,**

balance**,**

connBtnTxt

**}** **=** useWallet**(**zkMePopupNode**)**

**const** **{**

chainId**,**

cluster**,**

**}** **=** useChain**()**

**const** **{**

verify**,**

kycStatus**,**

checkingApproval**,**

needKyc**,**

accessToken

**}** **=** useVerify**(**connectedAddress**,** lv**)**

**const** **{**

onFinished**,**

showPopup**,**

**}** **=** useZkMe**(**connect**,** connectedAddress**,** signer**,** signingCosmWasmClient**)**

**const** **{**

stake**,**

staking**,**

showAnimation**,**

amount**,**

stakeBtnTxt

**}** **=** useStake**(**connect**,** signer**,** connecting**,** checkingApproval**,** kycStatus**)**

**const** feature **=** computed**(()** **=>** **{**

**return** **[**`100M ${cluster.value.rpc.symbol} staked`**,** '0% deposit fees'**,** 'Unstake instantly'**]**

**})**

**const** popupUrl **=** computed**(()** **=>** **{**

**const** **{** appId **}** **=** useMchConf**()**

**return** `${ZKME\_POPUP\_ORIGIN}?mchNo=${appId.value}&name=zkMe%20SDK%20DEMO&lv=${lv.value}&chainId=${chainId.value}&accessToken=${accessToken.value}`

**})**

onFinished**((**verifiedAddress**,** kycResults**)** **=>** **{**

**if** **(**verifiedAddress **!==** connectedAddress**.**value**)** **{**

**return**

**}**

**if** **(**lv**.**value **===** '2'**)** **{**

showPopup**.**value **=** **false**

kycStatus**.**value **=** 'valid'

**}** **else** **{**

kycResults **===** 'matching'

**?** verify**(**connectedAddress**.**value**)**

**:** **(**kycStatus**.**value **=** 'invalid'**)**

**}**

**})**

**async** **function** handleVerifyClick **()** **{**

// cosmos

**if** **(**cluster**.**value**.type** **===** 'cosmos'**)** **{**

showPopup**.**value **=** **true**

**return**

**}**

// EVM

**if** **(**window**.**ethereum**.**chainId **===** cluster**.**value**.**rpc**.**chainId**)** **{**

showPopup**.**value **=** **true**

**}** **else** **{**

**await** switchChain**()**

**try** **{**

**const** rp **=** **await** signer**.**value**?.**provider**.**getNetwork**()**

**if** **(**`0x${rp?.chainId.toString(16)}` **===** cluster**.**value**.**rpc**.**chainId**)** **{**

showPopup**.**value **=** **true**

**}**

**}** **catch** **(**err**:** any**)** **{**

console**.**log**(**err**.**message**)**

**if** **(**

err**.**message**.**includes**(**'network changed'**)** **&&**

**new** RegExp**(**`\\b => ${parseInt(cluster.value.rpc.chainId).toString()}\\b`**).**test**(**err**.**message**)**

**)** **{**

showPopup**.**value **=** **true**

**}**

**}**

**}**

**}**

**</**script**>**

**<**template**>**

**<**div **class=**"demo-tip"**>**zkMe SDK DEMO **-** Only to verify your zkMe identity**</**div**>**

**<**div **class=**"simulation-dapp-wrap"**>**

**<**div **class=**"simulation-dapp"**>**

**<**NavBar

**:**connecting**=**"connecting"

**:**conn**-**btn**-**txt**=**"connBtnTxt"

@connet**=**"connect"

**/>**

**<**ul **class=**"dot mui-fl-central"**>**

**<**li v**-for=**"(i, index) of feature" **:**key**=**"index"**>**

**{{** i **}}**

**</**li**>**

**</**ul**>**

**<**div **class=**"mui-fl-vert mui-fl-col"**>**

**<**h1 **class=**"heading"**>**zkMe SDK DEMO**</**h1**>**

**<**VerifyProgress

**:**connected**-**address**=**"connectedAddress"

**:**checking**-**approval**=**"checkingApproval"

**:**kyc**-**status**=**"kycStatus"

**:**kyc**-type=**"lv"

**:**animation**=**"showAnimation"

@verify**=**"handleVerifyClick"

**/>**

**</**div**>**

**<**StakePanel

v**-**model**=**"amount"

**:**staking**=**"staking || connecting"

**:**balance**=**"balance"

**:**stake**-**btn**-**txt**=**"stakeBtnTxt"

@stake**=**"stake"

**/>**

**</**div**>**

**</**div**>**

**<**Transition name**=**"fade"**>**

**<**div **class=**"popup-mask mui-fl-central" v**-**show**=**"showPopup && needKyc"**>**

**<**div **class=**"popup-wrap"**>**

**<**iframe v**-if=**"needKyc" ref**=**"zkMePopupNode" **:**src**=**"popupUrl" width**=**"100%" height**=**"100%"**></**iframe**>**

**</**div**>**

**</**div**>**

**</**Transition**>**

**</**template**>**

**<**style src**=**"@/assets/css/views/dapp-home.css" scoped**>**

**</**style**>**

src/composables/zkme.ts

**import** **type** **{** Ref**,** ShallowRef **}** **from** "vue"

**import** **type** **{** CosmosTransactionRequest**,** KycResults**,** ZkMeMessageBody **}** **from** "@/types"

**import** **{** ZKME\_POPUP\_ORIGIN **}** **from** "@/utils/config"

**import** **type** **{** JsonRpcSigner **}** **from** "ethers"

**import** **{** is **}** **from** '@/utils/common'

**import** **type** **{** SigningCosmWasmClient **}** **from** '@cosmjs/cosmwasm-stargate'

**import** **{** useChain **}** **from** "."

// import { switchChain } from "@/utils/eth"

**type** FinishedHook **=** **(**verifiedAddress**:** string**,** kycResults**?:** KycResults**)** **=>** void

**export** **function** useZkMe **(**

connect**:** **()** **=>** Promise**<**void**>,**

connectedAddress**:** Ref**<**string**>,**

signer**:** ShallowRef**<**JsonRpcSigner **|** **undefined>,**

signingCosmWasmClient**:** ShallowRef**<**SigningCosmWasmClient **|** **undefined>,**

**)** **{**

**const** showPopup **=** ref**(false)**

**const** kycResults **=** ref**<**KycResults**>()**

**const** verifiedAddress **=** ref**(**''**)**

**const** **{** cluster **}** **=** useChain**()**

**const** finishedHooks**:** FinishedHook**[]** **=** **[]**

**const** onFinished **=** **(**hook**:** FinishedHook**)** **=>** **{**

finishedHooks**.**push**(**hook**)**

**}**

**const** messageListener **=** **async** **(**ev**:** MessageEvent**<**ZkMeMessageBody**>)** **=>** **{**

**if** **(**ev**.**origin **!==** ZKME\_POPUP\_ORIGIN**)** **{**

**return**

**}**

**const** **{**

id**,**

method**,**

params**,**

kycResults**:** \_kycResults**,**

verifiedAddress**:** \_verifiedAddress**,**

event

**}** **=** ev**.**data

**if** **(**event **===** 'finished' **&&** \_verifiedAddress**)** **{**

kycResults**.**value **=** \_kycResults

verifiedAddress**.**value **=** \_verifiedAddress

finishedHooks**.**forEach**(**hook **=>** **{**

hook**(**\_verifiedAddress**,** \_kycResults**)**

**})**

**}** **else** **if** **(**event **===** 'close'**)** **{**

showPopup**.**value **=** **false**

**}**

**const** source **=** ev**.**source **as** Window

**const** handleApprove **=** **(**data**:** any**)** **=>** **{**

source**.**postMessage**({**

id**,**

message**:** 'ok'**,**

data

**},** ev**.**origin**)**

**}**

**const** handleReject **=** **(**message**:** string**)** **=>** **{**

source**.**postMessage**({**

id**,**

message

**},** ev**.**origin**)**

**}**

**if** **(**method **===** 'getUserAccounts'**)** **{**

**if** **(**connectedAddress**.**value**)** **{**

handleApprove**([**connectedAddress**.**value**])**

//

**}** **else** **{**

**try** **{**

**await** connect**()**

handleApprove**([**connectedAddress**.**value**])**

//

**}** **catch** **(**err**:** any**)** **{**

handleReject**(**err**.**message**)**

**}**

**}**

//

**}** **else** **if** **(**method **===** 'delegateTransaction'**)** **{**

**const** hasSigner **=** cluster**.**value**.type** **===** 'cosmos'

**?** signingCosmWasmClient**.**value

**:** signer**.**value

**if** **(**hasSigner**)** **{**

**try** **{**

// await switchChain(params[0].chainId)

**const** txHash **=** is**<**CosmosTransactionRequest**>(**params**,** 'senderAddress'**)**

**?** **(await** signingCosmWasmClient**.**value**!.**execute**(**

params**.**senderAddress**,**

params**.**contractAddress**,**

params**.**msg**,**

'auto'

**)).**transactionHash

**:** **(await** signer**.**value**!.**sendTransaction**(**params**!)).**hash

handleApprove**(**txHash**)**

//

**}** **catch** **(**err**:** any**)** **{**

handleReject**(**err**.**message**)**

**}**

**}** **else** **{**

handleReject**(**'Unauthorized'**)**

**}**

**}**

**} //** messageListener结束的地方

window**.**addEventListener**(**'message'**,** messageListener**)**

onUnmounted**(()** **=>** **{**

window**.**removeEventListener**(**'message'**,** messageListener**)**

**})**

watchEffect**(async** **()** **=>** **{**

**if** **(**connectedAddress**.**value**)** **{**

showPopup**.**value **=** **false**

**}**

**})**

**return** **{**

onFinished**,**

showPopup**,**

kycResults**,**

verifiedAddress**,**

**}**

**}**

src/composables/wallet.ts

**import** **type** **{** ShallowRef **}** **from** "vue"

**import** **{** BrowserProvider**,** formatEther**,** JsonRpcSigner **}** **from** "ethers"

**import** **{** CLUSTERS**,** ZKME\_POPUP\_ORIGIN **}** **from** "@/utils/config"

**import** **{** getSigner **}** **from** "@/utils/eth"

**import** **{** getSigningCosmWasmClient **}** **from** "@sei-js/core"

**import** **type** **{** SigningCosmWasmClient **}** **from** '@cosmjs/cosmwasm-stargate'

**import** **{** GasPrice **}** **from** '@cosmjs/stargate'

**import** **{** getOfflineSigner **}** **from** "@/utils/cosmos"

**let** isFirstListen **=** **true**

**const** connectedAddress **=** ref**(**''**)**

**const** signer **=** shallowRef**<**JsonRpcSigner**>()**

**const** signingCosmWasmClient **=** shallowRef**<**SigningCosmWasmClient**>()**

**export** **function** useWallet **(**popup**?:** ShallowRef**<**HTMLIFrameElement **|** **undefined>)** **{**

**const** connecting **=** ref**(false)**

**const** balance **=** ref**(**'0'**)**

**const** connBtnTxt **=** computed**(()** **=>** **{**

**if** **(!**connectedAddress**.**value**)** **{**

**return** 'Connect to a wallet'

**}**

**return** connectedAddress**.**value**.**substring**(**0**,** 5**)** **+** '...' **+** connectedAddress**.**value**.**substring**(**38**)**

**})**

**const** **{** cluster **}** **=** useChain**()**

**const** connect **=** **async** **()** **=>** **{**

**if** **(**connecting**.**value **||** connectedAddress**.**value**)** **return**

connecting**.**value **=** **true**

**try** **{**

**if** **(**cluster**.**value**.type** **===** 'cosmos'**)** **{**

**const** **{** address**,** offlineSigner **}** **=** **await** getOfflineSigner**()**

**const** \_signingCosmWasmClient **=** **await** getSigningCosmWasmClient**(**cluster**.**value**.**rpc**.**url**,** offlineSigner**,** **{**

gasPrice**:** GasPrice**.**fromString**(**'0.25usei'**)**

**})**

connectedAddress**.**value **=** address

signingCosmWasmClient**.**value **=** \_signingCosmWasmClient

**}** **else** **{**

// EVM

**const** \_signer **=** **await** getSigner**()**

connectedAddress**.**value **=** \_signer**.**address**.**toLowerCase**()**

signer**.**value **=** \_signer

**}**

**}** **finally** **{**

connecting**.**value **=** **false**

**}**

**}**

**if** **(**window**.**ethereum**)** **{**

isFirstListen **&&** window**.**ethereum**.**on**(**'accountsChanged'**,** **(**accounts**)** **=>** **{**

**if** **(**cluster**.**value**.type** **===** 'cosmos'**)** **{**

**return**

**}**

accounts **=** accounts**.**map**(**acc **=>** acc**.**toLowerCase**())**

popup**?.**value**?.**contentWindow**?.**postMessage**({**

event**:** 'accountsChanged'**,**

data**:** accounts

**},** ZKME\_POPUP\_ORIGIN**)**

connectedAddress**.**value **=** accounts**[**0**]** **||** ''

**if** **(**connectedAddress**.**value**)** **{**

signer**.**value **=** **new** JsonRpcSigner**(new** BrowserProvider**(**window**.**ethereum**),** connectedAddress**.**value**)**

**}** **else** **{**

signer**.**value **=** **undefined**

**}**

**})**

isFirstListen **&&** window**.**ethereum**.**on**(**'chainChanged'**,** **(**e**)** **=>** **{**

**if** **(**cluster**.**value**.type** **===** 'cosmos'**)** **{**

**return**

**}**

**if** **(**e **!==** cluster**.**value**.**rpc**.**chainId**)** **{**

window**.**location**.**reload**()**

**}** **else** **if** **(**connectedAddress**.**value**)** **{**

signer**.**value **=** **new** JsonRpcSigner**(new** BrowserProvider**(**window**.**ethereum**),** connectedAddress**.**value**)**

**}**

**})**

isFirstListen **=** **false**

**}**

watchEffect**(async** **()** **=>** **{**

**if** **(**cluster**.**value**.type** **===** 'cosmos'**)** **{**

**if** **(**signingCosmWasmClient**.**value**)** **{**

**const** **{** amount **}** **=** **await** signingCosmWasmClient**.**value**.**getBalance**(**connectedAddress**.**value**,** 'usei'**)**

balance**.**value **=** **(**Number**(**amount**)** **/** Math**.**pow**(**10**,** 6**)).**toString**()**

**}**

**}** **else** **if** **(**signer**.**value**)** **{**

**try** **{**

**const** rp **=** **await** signer**.**value**.**provider**.**getBalance**(**connectedAddress**.**value**)**

balance**.**value **=** formatEther**(**rp**)**

**}** **catch** **{**

//

**}**

**}**

**})**

// onUnmounted(() => {

// window.ethereum?.removeAllListeners()

// })

**return** **{**

connect**,**

connecting**,**

connectedAddress**,**

signer**,**

signingCosmWasmClient**,**

balance**,**

connBtnTxt

**}**

**}**

**export** **function** useChain **()** **{**

**const** chainId **=** computed**(()** **=>** **{**

**let** cid**:** string

**const** query **=** useRoute**()?.**query

**if** **(**query**)** **{**

cid **=** query**.**chainId **as** string

**}** **else** **{**

**const** url **=** **new** URL**(**location**.**href**)**

cid **=** url**.**searchParams**.get(**'chainId'**)** **||** ''

**}**

**!**cid **&&** **(**cid **=** '0x5'**)**

**return** cid**.**toLowerCase**()**

**})**

**const** chainIdAsDec **=** computed**(()** **=>** **{**

**const** c **=** parseInt**(**chainId**.**value**).**toString**()**

**if** **(**c **===** 'NaN'**)** **{**

**return** chainId**.**value

**}**

**return** c

**})**

**const** cluster **=** computed**(()** **=>** **{**

**return** CLUSTERS**[**chainIdAsDec**.**value**]** **||** CLUSTERS**[**'5'**]**

**})**

**return** **{**

chainId**,**

chainIdAsDec**,**

cluster

**}**