

Interacting with the Wallet

Wallet interaction is possible only in the browser, because NEAR's Wallet is web-based.

Most frequent action is Sign In. Your user is redirected to the Wallet page to authorize your application. Once the user has Signed In, an access key is saved in browser's LocalStorage. All following actions that require the access key will be allowed. In case a user needs to authorize a transaction that has a deposit attached, your user will be automatically redirected to the Wallet again.

Creating Wallet Connection

In Wallet connection you use a LocalStorage [KeyStore](#) .

- TestNet
- MainNet

```
const
{ connect , keyStores ,
WalletConnection
}
= nearAPI ;

const connectionConfig =
{ networkId :
"testnet" , keyStore :
new
keyStores . BrowserLocalStorageKeyStore ( ) , nodeUrl :
"https://rpc.testnet.near.org" , walletUrl :
"https://testnet.mynearwallet.com/" , helperUrl :
"https://helper.testnet.near.org" , explorerUrl :
"https://testnet.nearblocks.io" , } ;

// connect to NEAR const nearConnection =
await
connect ( connectionConfig ) ;

// create wallet connection const walletConnection =
new
WalletConnection ( nearConnection ) ; const
{ connect , keyStores ,
WalletConnection
}
= nearAPI ;

const connectionConfig =
{ networkId :
"mainnet" , keyStore :
new
```

```

keyStores . BrowserLocalStorageKeyStore ( ) , nodeUrl :
"https://rpc.mainnet.near.org" , walletUrl :
"https://wallet.mainnet.near.org" , helperUrl :
"https://helper.mainnet.near.org" , explorerUrl :
"https://nearblocks.io" , } ;
// connect to NEAR const nearConnection =
await
connect ( connectionConfig ) ;
// create wallet connection const walletConnection =
new
WalletConnection ( nearConnection ) ; ModulebrowserConnect ClassWalletConnection

```

Ask your user to Sign In

You first create a [WalletConnection](#) , and then call `requestSignIn` . This will redirect the current page to the Wallet authentication page. You can configure success and failure redirect URLs.

This action creates an access key that will be stored in the browser's local storage. You can optionally list `methodNames` you want to allow for the access key. If you don't specify `methodNames` or pass an empty array, then all methods are allowed to be called (the access key will be created with permissions to call all methods).

```

// const walletConnection = new WalletConnection(nearConnection); walletConnection . requestSignIn ( { contractId :
"example-contract.testnet.REPLACE_ME" , methodNames :
[ ] ,
// optional successUrl :
"REPLACE_ME://.com/success" ,
// optional redirect URL on success failureUrl :
"REPLACE_ME://.com/failure" ,
// optional redirect URL on failure } ) ; MethodWalletConnection.requestSignIn

```

tip Sign In is not required if you are using an alternative key store to local storage, or you are not signing transactions (meaning - you are only calling read-only/view methods on a contract)

Sign Out your user

```

// const walletConnection = new WalletConnection(nearConnection); walletConnection . signOut ( ) ;
MethodWalletConnection.signOut

```

Check if Signed In

```

// const walletConnection = new WalletConnection(nearConnection); if
( walletConnection . isSignedIn ( ) )
{ // user is signed in } MethodWalletConnection.isSignedIn

```

Get Wallet Account

Get the [Account](#) your user has signed in with in the Wallet.

Get Account ID (as string)

```

// const walletConnection = new WalletConnection(nearConnection); const walletAccountId = walletConnection .
getAccountId ( ) ; MethodWalletConnection.getAccountId

```

Get Account Object

```
// const walletConnection = new WalletConnection(nearConnection); const walletAccountObj = walletConnection . account (
) ; MethodWalletConnection.account ClassConnectedWalletAccount Edit this page Last updated on Jan 31, 2024 by gagdiez
```

Was this page helpful? Yes No

[Previous Using JavaScript API](#) [Next Account](#)