

Using JavaScript API to interact with NEAR

Quick Reference

- [Installation](#)
- [Interacting with the Wallet](#)
- [Accounts](#)
- [Contracts](#)
- [Utilities](#)

What is near-api-js

near-api-js is a complete library to interact with the NEAR blockchain. You can use it in the browser, or in Node.js runtime.

You'll typically first create a connection to NEAR with [connect](#) using a [KeyStore](#) . With the connection object you now can:

- Interact with the [Wallet](#)
- in a browser.
- Instantiate an [Account](#)
- object to:* Send tokens
- - Deploy contracts
- - Inspect, create or delete accounts
- - Manage keys for accounts.
- Instantiate a [Contract](#)
- object to call smart contract methods.

The library also contains some [utility functions](#) .

tip To quickly get started with integrating NEAR in a web browser, read our [Web Frontend integration](#) article. info Note the difference between near-api-js and near-sdk-js :

The JavaScript SDK is a library for developing smart contracts. It contains classes and functions you use to write your smart contract code.

The JavaScript API is a complete library for all possible commands to interact with NEAR. It's a wrapper for the RPC endpoints, a library to interact with NEAR Wallet in the browser, and a tool for keys management.

Install

Include near-api-js as a dependency in your package.

```
npm i --save near-api-js
```

Import

You can use the API library in the browser, or in Node.js runtime. Some features are available only in one of the environments. For example, the `WalletConnection` is only for the browser, and there are different `KeyStore` providers for each environment.

- Browser
- Node

```
import
```

```
*
```

```
as nearAPI
```

```
from
```

```
"near-api-js" ; const nearAPI =
```

```
require ( "near-api-js" ) ;
```

Key Store

If you sign transactions, you need to create a Key Store. In the browser, the LocalStorage KeyStore will be used once you ask your user to Sign In with the Wallet.

- Using Browser
- Using Credentials Directory
- Using a File
- Using a private key string

```
// creates keyStore using private key in local storage
```

```
const
{ keyStores }
= nearAPI ; const myKeyStore =
new
keyStores . BrowserLocalStorageKeyStore ( ) ; ClassBrowserLocalStorageKeyStore // creates a keyStore that searches for
keys in .near-credentials // requires credentials stored locally by using a NEAR-CLI command: near login //
https://docs.near.org/tools/cli#near-login
```

```
const
{ keyStores }
= nearAPI ; const homedir =
require ( "os" ) . homedir ( ) ; const
CREDENTIALS_DIR
=
".near-credentials" ; const credentialsPath =
require ( "path" ) . join ( homedir ,
CREDENTIALS_DIR ) ; const myKeyStore =
new
keyStores . UnencryptedFileSystemKeyStore ( credentialsPath ) ; ClassUnencryptedFileSystemKeyStore // creates
keyStore from a provided file // you will need to pass the location of the .json key pair
```

```
const
{
  KeyPair , keyStores }
=
require ( "near-api-js" ) ; const fs =
require ( "fs" ) ; const homedir =
require ( "os" ) . homedir ( ) ;
const
ACCOUNT_ID
=
"near-example.testnet" ;
// NEAR account tied to the keyPair const
NETWORK_ID
```

```

=
"testnet" ; // path to your custom keyPair location (ex. function access key for example account) const
KEY_PATH
=
"/.near-credentials/near-example-testnet/get_token_price.json" ;
const credentials =
JSON . parse ( fs . readFileSync ( homedir +
KEY_PATH ) ) ; const myKeyStore =
new
keyStores . InMemoryKeyStore ( ) ; myKeyStore . setKey ( NETWORK_ID , ACCOUNT_ID , KeyPair . fromString (
credentials . private_key ) ) ; ClassInMemoryKeyStore ClassKeyPair // creates keyStore from a private key string // you can
define your key here or use an environment variable
const
{ keyStores ,
KeyPair
}
= nearAPI ; const myKeyStore =
new
keyStores . InMemoryKeyStore ( ) ; const
PRIVATE_KEY
= "by8kdJoJHu7uUkKfoaLd2J2Dp1q1TigeWMG123pHdu9UREqPcshCM223kWadm" ; // creates a public / private key pair
using the provided private key const keyPair =
KeyPair . fromString ( PRIVATE_KEY ) ; // adds the keyPair you created to keyStore await myKeyStore . setKey ( "testnet" ,
"example-account.testnet" , keyPair ) ; ClassInMemoryKeyStore ClassKeyPair

```

Connecting to NEAR

The object returned from `connect` is your entry-point for all commands in the API. To sign a transaction you'll need [a KeyStore](#) to create a connection.

- TestNet
- MainNet
- LocalNet

```

const
{ connect }
= nearAPI ;
const connectionConfig =
{ networkId :
"testnet" , keyStore : myKeyStore ,
// first create a key store nodeUrl :
"https://rpc.testnet.near.org" , walletUrl :
"https://testnet.mynearwallet.com/" , helperUrl :
"https://helper.testnet.near.org" , explorerUrl :

```

```

"https://testnet.nearblocks.io" , } ; const nearConnection =
await
connect ( connectionConfig ) ; const
{ connect }
= nearAPI ;
const connectionConfig =
{ networkId :
"mainnet" , keyStore : myKeyStore ,
// first create a key store nodeUrl :
"https://rpc.mainnet.near.org" , walletUrl :
"https://wallet.mainnet.near.org" , helperUrl :
"https://helper.mainnet.near.org" , explorerUrl :
"https://nearblocks.io" , } ; const nearConnection =
await
connect ( connectionConfig ) ; const
{ connect }
= nearAPI ; const connectionConfig =
{ networkId :
"local" , nodeUrl :
"http://localhost:3030" , walletUrl :
"http://localhost:4000/wallet" , } ; const nearConnection =
await

```

connect (connectionConfig) ; [Moduleconnect Edit this page](#) Last updated on Jan 31, 2024 by gagdiez Was this page helpful?
Yes No

[Previous](#) [Home](#) [Next](#) [Wallet](#)