

NEAR for Ethereum developers

NEAR components are chain-agnostic, enabling you to create decentralized frontends tailored for any Ethereum dApps. Throughout this article, we'll navigate prevalent use-cases with code snippets. You'll also find links to exemplary NEAR components for reference.

Interact with Ethereum using Ethers.js

The NEAR VM has imported the [Ethers.js](#) library, allowing for seamless interaction with Ethereum nodes using JavaScript in a NEAR component.

The Ethers object exposes the provider connection:

- Ethers.provider()
- is a read-only connection to the blockchain, which allows querying the blockchain state (e.g., account, block or transaction details), querying event logs or evaluating read-only code using call.* Ethers.provider().getSigner()
- - abstracts the class that interacts with an account* Ethers.provider().getSigner().getBalance()
- - - returns a Promise that resolves to the account address.
- - Ethers.provider().getBlockNumber()
- - looks up the current block number (i.e. height)
- - Ethers.provider().getFeeData()
- - gets the best guess at the recommended FeeData

You can see various ways of using Ethers objects on the [Ethers documentation portal](#)).

Furthermore, numerous [basic tools](#) can be found in the ethers.utils object (be aware of the lowercase 'e').

- ethers.utils.parseUnits(value, unit)
- converts the decimal string value to a BigInt, assuming unit decimal places. The unit may be the number of decimal places or the name of a unit (e.g. "gwei" for 9 decimal places).
- ethers.utils.formatEther(wei)
- converts value into a decimal string using 18 decimal places.

Example showing the difference between Ethers.provider()

and ethers.utils :

```
Ethers . provider ( ) . getSigner ( ) . getBalance ( ) . then ( ( balance )
```

```
=>
```

```
{ console . log ( "Your ETH balance: " , ethers . utils . formatEther ( balance ) ) } ;
```

FAQ

How to get a user account?

```
const receiver =
```

```
Ethers . provider ( ) . send ( "eth_requestAccounts" ,
```

```
[ ] ) [ 0 ] ;
```

How to get the current chain ID?

```
Ethers . provider ( ) . getNetwork ( ) . then ( ( chainIdData )
```

```
=>
```

```
{ console . log ( chainIdData . chainId ) ; } ;
```

How to show a Web3Login button?

```
< Web3ConnectButton className = "my-class" connectLabel = "Connect Web3 Wallet" disconnectLabel = "Disconnect Web3 Wallet" connectingLabel = "Connecting..." /
```

How to load a contract ABI?

```
const abi =  
  
fetch ( https://eth.blockscout.com/api?module=contract&action=getabi&address=0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2 ) ;  
  
if  
  
( ! abi . ok )  
  
{ return  
  
"Loading" ; }  
  
console . log ( abi . body . result ) ;
```

How to load data from EVM nodes?

```
// create a contract interface const iface =  
  
new  
  
ethers . utils . Interface ( abi . body . result ) ;  
  
// encode the balanceOf get request const encodedBalanceData = iface . encodeFunctionData ( "balanceOf" ,  
[ receiver ] ) ;  
  
// perform a call Ethers . provider ( ) . call ( { to :  
  
"0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2" , data : encodedBalanceData , } ) . then ( ( rawBalance )  
  
=>  
  
{ // decode the result const receiverBalanceHex = iface . decodeFunctionResult ( "balanceOf" , rawBalance ) ; console . log  
( Big ( receiverBalanceHex ) . toFixed ( ) ) ; } ) ;
```

How to send a transaction to Ethereum nodes?

```
// create a contract instance const wEthContract =  
  
new  
  
ethers . Contract ( "0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2" , abi . body . result , Ethers . provider ( ) . getSigner  
( ) ) ;  
  
// perform a given method (withdraw in this case) wEthContract . withdraw ( balance ,  
  
{ value :  
  
0 } ) . then ( ( transactionHash )  
  
=>  
  
{ console . log ( transactionHash ) ; } ) ;
```

How to convert HEX values to human readable numbers?

```
// use Big().toFixed method  
  
Big ( balanceHex ) . toFixed ( )
```

How to locate the ABI of a Smart Contract?

Utilize platforms like etherscan.com and blockscout.com, or delve into the project's GitHub repositories and documentation.

Is there an example of applications with a description of how they were created?

Yes, there is an article that describes [best practices for creating applications for Ethereum developers on NEAR](#), with a breakdown of individual steps.

Examples

Below is a curated list of NEAR components, developed by the community, that epitomize the best practices for creating NEAR components:

- [Unwrap ETH](#)
- [ERC-20 Token sender](#)
- [Lido Component](#)
- [Swap component](#)
- (a set of swap dApps for Ethereum Mainnet, Polygon Mainnet, Polygon zkEVM, Aurora, NEAR, etc)
- [1inch](#)
- [Liquity](#) [Edit this page](#) Last updated on Jan 9, 2024 by gagdiez Was this page helpful? Yes No

[Previous WebSockets & QueryAPI](#) [Next Best Practices for Ethereum developers on NEAR](#)