Hello Lido!

If you are ready to explore further, theLido component serves as an excellent fully-fledged example, as it demonstrates how to interact with a smart contract deployed on the Ethereum mainnet.

Ethers.js This component uses theEthers JavaScript library to interact with Ethereum smart contracts. Followthis link for the officialethers.js documentation. Web3 connect The Lido example uses theWeb3Connect component to provide aWalletConnect modal so the user can connect with any Web3 Ethereum wallet like Ledger or MetaMask.

Fork the component

1. Navigate tothe component
2. SelectFork
3. Feel free to make any changes
4. Click onSave
5. to deploy the component

note To deploy the component, you'll need to sign in with a NEAR account and to make a deposit of a small amount of NEAR for the storage cost. This is because the components are stored in the NEAR network.

Source code

if

(state . chainId

===

undefined

&& ethers !==

undefined

&& Ethers . send ("eth_requestAccounts" ,

[ ]) [0])

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

=>

{ if

(chainIdData ?. chainId)

{ State . update ({

chainId : chainIdData . chainId

}) ; } }) ; } if

(state . chainId

!==

undefined

&& state . chainId

!==

1)

{ return

< p

Switch to Ethereum

Mainnet < / p

```
        ; }
// FETCH LIDO ABI
const lidoContract =
"0xae7ab96520de3a18e5e111b5eaab095312d7fe84" ; const tokenDecimals =
18 ;
const lidoAbi =
fetch ( "https://raw.githubusercontent.com/lidofinance/lido-subgraph/master/abis/Lido.json" ) ; if
( ! lidoAbi . ok )
{ return
"Loading" ; }
const iface =
new
ethers . utils . Interface ( lidoAbi . body ) ;
// FETCH LIDO STAKING APR
if
( state . lidoArp
===
undefined )
{ const apr =
fetch ( "https://api.allorigins.win/get?url=https://stake.lido.fi/api/sma-steth-apr" ) ; if
( ! apr )
return ; State . update ( {
lidoArp :
JSON . parse ( apr ?. body ?. contents )
??
"..."
} ) ; }
// HELPER FUNCTIONS
const
getStakedBalance
=
( receiver )
=>
{ const encodedData = iface . encodeFunctionData ( "balanceOf" ,
[ receiver ] ) ;
return
Ethers . provider ( ) . call ( { to : lidoContract , data : encodedData , } ) . then ( ( rawBalance )
```

```
=>
{ const receiverBalanceHex = iface . decodeFunctionResult ( "balanceOf" , rawBalance ) ;
return
Big ( receiverBalanceHex . toString ( ) ) . div ( Big ( 10 ) . pow ( tokenDecimals ) ) . toFixed ( 2 ) . replace ( / \d (?= ( \d {3} ) + . ) / g ,
"&," ) ; } ) ; } ;
const
submitEthers
=
( strEther , _referral )
=>
{ if
( ! strEther )
{ return
console . log ( "Amount is missing" ) ; } const erc20 =
new
ethers . Contract ( lidoContract , lidoAbi . body , Ethers . provider ( ) . getSigner ( ) ) ;
let amount = ethers . utils . parseUnits ( strEther , tokenDecimals ) . toHexString ( ) ;
erc20 . submit ( lidoContract ,
{
value : amount } ) . then ( ( transactionHash )
=>
{ console . log ( "transactionHash is "
+ transactionHash ) ; } ) ; } ;
// DETECT SENDER
if
( state . sender
===
undefined )
{ const accounts =
Ethers . send ( "eth_requestAccounts" ,
[ ] ) ; if
( accounts . length )
{ State . update ( {
sender : accounts [ 0 ]
} ) ; console . log ( "set sender" , accounts [ 0 ] ) ; } }
//if (!state.sender) return "Please login first";
// FETCH SENDER BALANCE
```

if

(state . balance

===

undefined

&& state . sender)

{ Ethers . provider ( ) . getBalance (state . sender) . then ((balance)

=>

{ State . update ({

balance :

Big (balance) . div (Big (10) . pow (18)) . toFixed (2)

}) ; }) ; }

// FETCH SENDER STETH BALANCE

if

(state . stakedBalance

===

undefined

&& state . sender)

{ getStakedBalance (state . sender) . then ((stakedBalance)

=>

{ State . update ({ stakedBalance }) ; }) ; }

// FETCH TX COST

if

(state . txCost

===

undefined)

{ const gasEstimate = ethers . BigNumber . from (1875000) ; const gasPrice = ethers . BigNumber . from (1500000000) ;

const gasCostInWei = gasEstimate . mul (gasPrice) ; const gasCostInEth = ethers . utils . formatEther (gasCostInWei) ;

let responseGql =

fetch ("https://api.thegraph.com/subgraphs/name/uniswap/uniswap-v2" , { method :

"POST" , headers :

{

"Content-Type" :

"application/json"

} , body :

JSON . stringify ({ query :

{ bundle(id: "1") { ethPrice } } , }) , }) ;

if

```
( ! responseGql )
return
"" ;
const ethPriceInUsd = responseGql . body . data . bundle . ethPrice ;
const txCost =
Number ( gasCostInEth )
*
Number ( ethPriceInUsd ) ;
State . update ( {
txCost :
`
{ txCost . toFixed ( 2 ) } `
} ) ; }
// FETCH CSS
const cssFont =
fetch ( "https://fonts.googleapis.com/css2?family=Manrope:wght@200;300;400;500;600;700;800" ) . body ; const css =
fetch ( "https://pluminite.mypinata.cloud/ipfs/Qmboz8aoSvVXLeP5pZbRtNKtDD3kX5D9DEnfMn2ZGSJWtP" ) . body ;
if
( ! cssFont ||
! css )
return
"" ;
if
( ! state . theme )
{ State . update ( { theme : styled . div
ont-family : Manrope , -apple-system , BlinkMacSystemFont , Segoe UI , Roboto , Oxygen , Ubuntu , Cantarell , Fira Sans , Droid Sans , Helvetica Neue , sans-serif ; { cssFont } { css } , } ) ; } const
Theme
= state . theme ;
// OUTPUT UI
const
getSender
=
( )
=>
{ return
! state . sender ?
"" : state . sender . substring ( 0 ,
6 )
```

```
+ "..."
+ state . sender . substring ( state . sender . length
-
4 , state . sender . length ) ; } ;
return
( < Theme
        < div className = "LidoContainer"
        < div className = "Header"
        Stake
Ether < / div
        < div className = "SubHeader"
        Stake
ETH and receive stETH while staking . < / div
< div className = "LidoForm"
        { state . sender
&&
( <
        < div className = "LidoFormTopContainer"
        < div className = "LidoFormTopContainerLeft"
        < div className = "LidoFormTopContainerLeftContent1"
        < div className = "LidoFormTopContainerLeftContent1Container"
        < span
        Available to stake < / span
        < div className = "LidoFormTopContainerLeftContent1Circle"
/
        < / div
        < / div
        < div className = "LidoFormTopContainerLeftContent2"
        < span
        { state . balance
??
( ! state . sender
?
"0"
:
"..." ) } & nbsp ; ETH < / span
        < / div
```

```
        < / div
        < div className = "LidoFormTopContainerRight"
        < div className = "LidoFormTopContainerRightContent1"
        < div className = "LidoFormTopContainerRightContent1Text"
        < span
        { getSender ( ) } < / span
        < / div
        < / div
        < / div
        < / div
        < div className = "LidoSplitter"
/
        < /
        ) } < div className = { state . sender
?
"LidoFormBottomContainer"
:
"LidoFormTopContainer" }
        < div className = "LidoFormTopContainerLeft"
        < div className = "LidoFormTopContainerLeftContent1"
        < div className = "LidoFormTopContainerLeftContent1Container"
        < span
        Staked amount < / span
        < / div
        < / div
        < div className = "LidoFormTopContainerLeftContent2"
        < span
        { state . stakedBalance
??
( ! state . sender
?
"0"
:
"..." ) } & nbsp ; stETH < / span
        < / div
        < / div
        < div className = "LidoFormTopContainerRight"
```

```
        < div className = "LidoAprContainer"

        < div className = "LidoAprTitle"

        Lido

APR < / div

        < div className = "LidoAprValue"

        { state . lidoArp

??

"..." } % < / div

        < / div

        < / div

        < / div

        < / div

        < div className = "LidoStakeForm"

        < div className = "LidoStakeFormInputContainer"

        < span className = "LidoStakeFormInputContainerSpan1"

        < svg width = "24" height = "24" viewBox = "0 0 24 24" fill = "currentColor"

        < path opacity = "0.6" d = "M11.999 3.75v6.098l5.248 2.303-5.248-8.401z"

        < / path

        < path d = "M11.999 3.75L6.75 12.151l5.249-2.303V3.75z"

        < / path

        < path opacity = "0.6" d = "M11.999 16.103v4.143l5.251-7.135L12 16.103z"

        < / path

        < path d = "M11.999 20.246v-4.144L6.75 13.111l5.249 7.135z"

        < / path

        < path opacity = "0.2" d = "M11.999 15.144l5.248-2.993-5.248-2.301v5.294z"

        < / path

        < path opacity = "0.6" d = "M6.75 12.151l5.249 2.993V9.85l-5.249 2.3z"

        < / path

        < / svg

        < / span

        < span className = "LidoStakeFormInputContainerSpan2"

        < input disabled = { ! state . sender } className = "LidoStakeFormInputContainerSpan2Input" value = { state .
        strEther } onChange = { ( e )

=>

State . update ( {

strEther : e . target . value

} ) } placeholder = "Amount" /

        < / span
```

```
        < span className = "LidoStakeFormInputContainerSpan3" onClick = { ( )
=>
{ State . update ( { strEther :
( state . balance
0.05 ?
parseFloat ( state . balance )
-
0.05 :
0 ) . toFixed ( 2 ) , } ) ; } }
        < button className = "LidoStakeFormInputContainerSpan3Content" disabled = { ! state . sender }
        < span className = "LidoStakeFormInputContainerSpan3Max"
        MAX < / span
        < / button
        < / span
        < / div
        { ! ! state . sender
?
( < button className = "LidoStakeFormSubmitContainer" onClick = { ( )
=>
submitEthers ( state . strEther , state . sender ) }
        < span
        Submit < / span
        < / button
        )
:
( < Web3Connect className = "LidoStakeFormSubmitContainer" connectLabel = "Connect with Web3" /
        ) }
< div className = "LidoFooterContainer"
        { state . sender
&&
( < div className = "LidoFooterRaw"
        < div className = "LidoFooterRawLeft"
        You will receive < / div
        < div className = "LidoFooterRawRight"
{ state . strEther
??
0 } stETH < / div
```

< / div

) } < div className = "LidoFooterRaw"

< div className = "LidoFooterRawLeft"

Exchange rate < / div

< div className = "LidoFooterRawRight"

1

ETH

=

1 stETH < / div

< / div

{ false

&&

(< div className = "LidoFooterRaw"

< div className = "LidoFooterRawLeft"

Transaction cost < / div

< div className = "LidoFooterRawRight"

{ state . txCost } < / div

< / div

) } < div className = "LidoFooterRaw"

< div className = "LidoFooterRawLeft"

Reward fee < / div

< div className = "LidoFooterRawRight"

10 % < / div

< / div

< / div

< / div

< / div

< / Theme

) ;

Fork the component

1. Navigate tothe component
2. SelectFork
3. Feel free to make any changes
4. Click onSave
5. to deploy the component

note To deploy the component, you'll need to sign in with a NEAR account and to make a deposit of a small amount of NEAR for the storage cost. This is because NEAR components are stored in the NEAR network. Edit this page Last updatedonJan 31, 2024 bygagdiez Was this page helpful? Yes No

Previous Best Practices for Ethereum developers on NEARNext Introduction