Web3 Unleashed: Build a dapp with Next.js and the MetaMask API¶

Written byEric Bishard

Last updated 10/12/2022

Overview¶

In this edition ofWeb3 Unleashed , we're interviewing the MetaMask DevRel teamGui Bibeau andEric Bishard about the MetaMask API and other tools and features on the horizon like their MetaMask SDK. We will be building a dapp live with them to custom integrate MetaMask into a NextJS application!

If you would like to follow along with the Web3 Unleashed Episode #7 demo during or after the stream, below are the steps we walked through to build our dapp integration with MetaMask API.

Prerequisites¶

- MetaMask Extension
- NodeJS
- NPM
- Git

Clone and Install Dependencies¶

Clone theMetaMask API Web3 Unleashed repo . Then, to get your project started, change into the directory and install the dependencies:

git clone https://github.com/metamask/nextjs-starter.gitcd

nextjs-starter npm i &&

npm run dev This will give us a starting point in a NextJS application to build our demo.

Connecting the User¶

We will start by updating thehooks/useMetamask.tsx file. This is our global app context that utilizes "out of the box" Context API in React.

Updatehooks/useMetamask.tsx file:

import

React ,

{

useEffect ,

type

PropsWithChildren

}

from

"react" ; type

ConnectAction

=

{

type :

"connect" ;

wallet :

string

}; type

DisconnectAction

=

{

type :

"disconnect"

}; type

PageLoadedAction

=

{

type :

"pageLoaded" ;

isMetamaskInstalled :

boolean

}; type

LoadingAction

=

{

type :

"loading"

}; type

Action

=

|

ConnectAction

|

DisconnectAction

|

PageLoadedAction

|

LoadingAction ; type

Dispatch

=

(action :

Action)

=>

void ; type

Status

=

"loading"

|

"idle"

|

"pageNotLoaded" ; type

State

=

{

wallet :

string

|

null ;

isMetamaskInstalled :

boolean ;

status :

Status ; }; const

MetamaskContext

=

React . createContext <

{

state :

State ;

dispatch :

Dispatch

}

|

undefined

(undefined); const

initialState :

State

=

{

wallet :

```
null ,
isMetamaskInstalled :
false ,
status :
"loading" , }
as
const ; function
metamaskReducer ( state :
State ,
action :
Action ) :
State
{
switch
( action . type )
{
case
"connect" :
{
const
{
wallet
}
=
action ;
return
{
... state ,
wallet ,
status :
"idle"
};
}
case
"disconnect" :
{
return
```

```
{
... state ,
wallet :
null
};
}
case
"pageLoaded" :
{
const
{
isMetamaskInstalled
}
=
action ;
return
{
... state ,
isMetamaskInstalled ,
status :
"idle"
};
}
case
"loading" :
{
return
{
... state ,
status :
"loading"
};
}
default :
{
throw
new
```

```
Error ( "Unhandled action type" );
}
} } function
MetamaskProvider ({
children
} :
PropsWithChildren )
{
const
[ state ,
dispatch ]
=
React . useReducer ( metamaskReducer ,
initialState );
const
value
=
{
state ,
dispatch
};
useEffect (()
=>
{
if
( typeof
window
!==
undefined )
{
// start by checking if window.ethereum is present, indicating a wallet extension
const
ethereumProviderInjected
=
typeof
window . ethereum
!==
```

"undefined" ;

// this could be other wallets so we can verify if we are dealing with metamask

// using the boolean constructor to be explicit and not let this be used as a falsy value (optional)

const

isMetamaskInstalled

=

ethereumProviderInjected

&&

Boolean (window . ethereum . isMetaMask);

dispatch ({

type :

"pageLoaded" ,

isMetamaskInstalled

});

}

},

[]);

return

(

< MetamaskContext . Provider

value

{ value }

{ children }

< /MetamaskContext.Provider>

); } function

useMetamask ()

{

const

context

=

React . useContext (MetamaskContext);

if

(context

===

undefined)

{

throw

new

Error ("useMetamask must be used within a MetamaskProvider");

}

return

context ; } export

{

MetamaskProvider ,

useMetamask

}; The above change is by far one of our largest changes that we will do at one time but this file is in charge of helping us keep our application state in sync with the wallet state and is crucial so that we can build the components and features that we want.

After this change you will might notice red squiggly lines under thewindow.ethereum object, this is only because if we want TypeScript to stop yelling at us in our code editor, we need to tell it whatwindow.ethereum is type-wise.

Add the filetypes.d.ts to the app root:

type

InjectedProviders

=

{

isMetaMask? :

true ; }; interface

Window

{

ethereum :

InjectedProviders

&

{

on :

(...args :

any [])

=>

void ;

removeListener ?:

(...args :

any [])

=>

void ;

request < T

=

any

(args :

any) :

Promise < T

;

}; } You should no longer see those warnings in yourhooks/useMetamask.tsx file.

Create acomponents/Loading.tsx file:

import

{

type

FC

}

from

"react" ; const

dot

=

rounded-full h-2 w-2 mx-0.5 bg-current animate-[blink_1s_ease_0s_infinite_normal_both]" ; let

style

=

{

animationDelay :

"0.2s"

}; export

const

Loading :

FC

=

()

=>

{

return

(

< span

className

"inline-flex text-center items-center leading-7 h-6"

< span

className

{ dot }

key

"dot_1"

/>

< span

className

{ dot }

style

{ style }

key

"dot_2"

/>

< span

className

{ dot }

style

{ style }

key

"dot_3"

/>

< /span>

); }; With our type definitions added, our MetamaskContext Provider updated, and ourLoading.tsx in place, we can now make changes to ourcomponents/Wallet.tsx file and add a loading state for our app.

Update thecomponents/Wallet.tsx file to:

import

Image

from

"next/future/image" ; import

Link

from

"next/link" ; import

{

```
useMetamask
}
from
"../hooks/useMetamask" ; import
{
Loading
}
from
"./Loading" ; export
default
function
Wallet ()
{
const
{
dispatch ,
state :
{
status ,
isMetamaskInstalled
},
}
=
useMetamask ();
const
showInstallMetamask
=
status
!==
"pageNotLoaded"
&&
! isMetamaskInstalled ;
const
showConnectButton
=
status
!==
```

```
"pageNotLoaded"
&&
isMetamaskInstalled ;
const
handleConnect
=
async
()
=>
{
dispatch ({
type :
"loading"
});
const
accounts
=
await
window . ethereum . request ({
method :
"eth_requestAccounts" ,
});
if
( accounts . length
0 )
{
dispatch ({
type :
"connect" ,
wallet :
accounts [ 0 ]
});
}
};
return
(
< div
```

className

"bg-truffle"

< div

className

"mx-auto max-w-2xl py-16 px-4 text-center sm:py-20 sm:px-6 lg:px-8"

< h2

className

"text-3xl font-bold tracking-tight text-white sm:text-4xl"

< span

className

"block"

Metamask

API

intro < /span>

< /h2>

< p

className

"mt-4 text-lg leading-6 text-white"

Follow

along

with

the { " " }

< Link

href

"https://github.com/GuiBibeau/web3-unleashed-demo"

target

"_blank"

< span

className

"underline cursor-pointer"

Repo < /span>

< /Link>{" "}

in

order

to

learn

how

to

use

the

Metamask

API .

< /p>

{ showConnectButton

&&

(

< button

onClick

{ handleConnect }

className

"mt-8 inline-flex w-full items-center justify-center rounded-md border border-transparent bg-ganache text-white px-5 py-3 text-base font-medium sm:w-auto"

{ status

===

"loading"

?

< Loading

/>

:

"Connect Wallet" }

< /button>

)}

{ showInstallMetamask

&&

(

< Link

href

"https://metamask.io/"

target

"_blank"

< a

className

"mt-8 inline-flex w-full items-center justify-center rounded-md border border-transparent bg-ganache text-white px-5 py-3 text-base font-medium sm:w-auto"

Connect

Wallet

< /a>

< /Link>

)}

< /div>

< /div>

); } This imports the loading component, further destructures the return value of ouruseMetaMask() custom hook, sets up variables to track if MetaMask is installed or connected for conditional rendering, and gives us ahandleConnect() function for dispatching changes to our state reducer.

If we are tracking our changes we can see that we have touched 4 files by creating or updating/refactoring. At this point we should be able to connect a user to our dapp with MetaMask.

Run the project and attempt to connect to your MetaMask wallet.

npm run dev Two things are happening now:

1. If a user does not have MetaMask installed they will get a "Connect Wallet" button that simply takes you to download MetaMask.
2. If MetaMask is installed they will see a "Connect Wallet" button that actually connects their wallet to the dapp.

We are not yet hiding the button once connected or displaying any wallet information. As well, you will notice in MetaMask that you are connected to the dapp. To test the Install link you can go into your extension manager and disable MetaMask temporarily. Checkout the Diff to see what changed

Use the MetaMask API to get User Info¶

We want to display the balance from our wallet, and the public address of the wallet account that is connected to our dapp. For this we need to make a few changes again to thehooks/useMetamask.tsx and add the logic and JSX/HTML in ourcomponents/Wallet.tsx .

Updatehooks/useMetamask.tsx to:

import

React ,

{

useEffect ,

type

PropsWithChildren

}

```
from
"react" ; type
ConnectAction
=
{
type :
"connect" ;
wallet :
string ;
balance :
string
}; type
DisconnectAction
=
{
type :
"disconnect"
}; type
PageLoadedAction
=
{
type :
"pageLoaded" ;
isMetamaskInstalled :
boolean
}; type
LoadingAction
=
{
type :
"loading"
}; type
Action
=
|
ConnectAction
|
```

```
DisconnectAction
|
PageLoadedAction
|
LoadingAction ; type
Dispatch
=
( action :
Action )
=>
void ; type
Status
=
"loading"
|
"idle"
|
"pageNotLoaded" ; type
State
=
{
wallet :
string
|
null ;
isMetamaskInstalled :
boolean ;
status :
Status ;
balance :
string
|
null ; }; const
initialState :
State
=
{
```

```
wallet :
null ,
isMetamaskInstalled :
false ,
status :
"loading" ,
balance :
null , }
as
const ; function
metamaskReducer ( state :
State ,
action :
Action ) :
State
{
switch
( action . type )
{
case
"connect" :
{
const
{
wallet ,
balance
}
=
action ;
return
{
... state ,
wallet ,
balance ,
status :
"idle"
};
```

```
}
case
"disconnect" :
{
return
{
... state ,
wallet :
null
};
}
case
"pageLoaded" :
{
const
{
isMetamaskInstalled
}
=
action ;
return
{
... state ,
isMetamaskInstalled ,
status :
"idle"
};
}
case
"loading" :
{
return
{
... state ,
status :
"loading"
};
```

```
}
default :
{
throw
new
Error ( "Unhandled action type" );
}
} } const
MetamaskContext
=
React . createContext <
{
state :
State ;
dispatch :
Dispatch
}
|
undefined
        ( undefined ); function
MetamaskProvider ({
children
} :
PropsWithChildren )
{
const
[ state ,
dispatch ]
=
React . useReducer ( metamaskReducer ,
initialState );
const
value
=
{
state ,
dispatch
```

```
};
useEffect (()
=>
{
if
( typeof
window
!==
undefined )
{
// start by checking if window.ethereum is present, indicating a wallet extension
const
ethereumProviderInjected
=
typeof
window . ethereum
!==
"undefined" ;
// this could be other wallets so we can verify if we are dealing with metamask
// using the boolean constructor to be explecit and not let this be used as a falsy value (optional)
const
isMetamaskInstalled
=
ethereumProviderInjected
&&
Boolean ( window . ethereum . isMetaMask );
dispatch ({
type :
"pageLoaded" ,
isMetamaskInstalled
});
}
},
[]);
return
(
< MetamaskContext . Provider
```

value

{ value }

{ children }

< /MetamaskContext.Provider>

); } function

useMetamask ()

{

const

context

=

React . useContext (MetamaskContext);

if

(context

===

undefined)

{

throw

new

Error ("useMetamask must be used within a MetamaskProvider");

}

return

context ; } export

{

MetamaskProvider ,

useMetamask

}; We have done some slight refactoring to account for the ability to track the state of the wallet balance, addedbalance to ourinitialState , and updated ourconnect action in our reducer

Updatecomponents/Wallet.tsx to:

import

Image

from

"next/future/image" ; import

Link

from

"next/link" ; import

{

useMetamask

```
}
from
"../hooks/useMetamask" ; import
{
Loading
}
from
"./Loading" ; export
default
function
Wallet ()
{
const
{
dispatch ,
state :
{
status ,
isMetamaskInstalled ,
wallet ,
balance
},
}
=
useMetamask ();
const
showInstallMetamask
=
status
!==
"pageNotLoaded"
&&
! isMetamaskInstalled ;
const
showConnectButton
=
status
```

!==

"pageNotLoaded"

&&

isMetamaskInstalled

&&

! wallet ;

const

handleConnect

=

async

()

=>

{

dispatch ({

type :

"loading"

});

const

accounts

=

await

window . ethereum . request ({

method :

"eth_requestAccounts" ,

});

if

(accounts . length

0)

{

const

balance

=

await

window . ethereum ! . request ({

method :

"eth_getBalance" ,

params :

[accounts [0],

"latest"],

});

dispatch ({

type :

"connect" ,

wallet :

accounts [0],

balance

});

}

};

return

(

< div

className

"bg-truffle"

< div

className

"mx-auto max-w-2xl py-16 px-4 text-center sm:py-20 sm:px-6 lg:px-8"

< h2

className

"text-3xl font-bold tracking-tight text-white sm:text-4xl"

< span

className

"block"

Metamask

API

intro < /span>

< /h2>

< p

className

"mt-4 text-lg leading-6 text-white"

Follow

along

with

the { " " }

< Link

href

"https://github.com/GuiBibeau/web3-unleashed-demo"

target

"_blank"

< span

className

"underline cursor-pointer"

Repo < /span>

< /Link>{" "}

in

order

to

learn

how

to

use

the

Metamask

API .

< /p>

{ wallet

&&

(

< div

className

" px-4 py-5 sm:px-6"

< div

className

"-ml-4 -mt-4 flex flex-wrap items-center justify-between sm:flex-nowrap"

< div

className

"ml-4 mt-4"

< div

className

"flex items-center"

< div

className

"ml-4"

< h3

className

"text-lg font-medium leading-6 text-white"

Address :

< span

{ wallet } < /span>

< /h3>

< p

className

"text-sm text-white"

Balance :

< span

{ balance } < /span>

< /p>

< /div>

< /div>

< /div>

< /div>

< /div>

)}

{ showConnectButton

&&

(

< button

onClick

{ handleConnect }

className

"mt-8 inline-flex w-full items-center justify-center rounded-md border border-transparent bg-ganache text-white px-5 py-3 text-base font-medium sm:w-auto"

{ status

===

"loading"

?

< Loading

/>

:

"Connect Wallet" }

< /button>

)}

{ showInstallMetamask

&&

(

< Link

href

"https://metamask.io/"

target

"_blank"

< a

className

"mt-8 inline-flex w-full items-center justify-center rounded-md border border-transparent bg-ganache text-white px-5 py-3 text-base font-medium sm:w-auto"

Connect

Wallet

< /a>

< /Link>

)}

< /div>

< /div>

); } We have also added balance to our destructured object so that we have access to it in our component, updated theshowConnectButton logic, and requested the balance using theeth_getBalance method.

We have also updated our JSX/HTML to include anaddress andbalance .

This is a great start, but our UI is still lacking and there is more logic we need to properly track our wallet state and update the page because if we connect to our wallet we get a funny display for our balance and if we refresh our page, we don't see our address and balance. But we will now fix those issues.

Checkout the Diff to see what changed

Two Way Communication with MetaMask¶

Again, we will be updating thehooks/useMetamask.tsx andcomponents/Wallet.tsx files. The idea will be to add a few more reducer actions includingLoading andIdle states for the page, we will fix our button to say"Install MetaMask" instead of"Connect MetaMask" , and we will parse thebalance to display a readable number.

Finally, we will add some code that uses thewallet_watchAsset MetaMask (RPC API) method to addUSDC token to our MetaMask wallet. This will enable our users to see those tokens in their wallet if they have them. If a dApp uses a particular token, we can programmatically do this for them rather than expecting to do it themselves manually through the MetaMask UI.

Updatehooks/useMetamask.tsx to:

import

React ,

{

useEffect ,

type

PropsWithChildren

}

from

"react" ; type

ConnectAction

=

{

type :

"connect" ;

wallet :

string ;

balance :

string

}; type

DisconnectAction

=

{

type :

"disconnect"

}; type

PageLoadedAction

=

{

type :

"pageLoaded" ;

isMetamaskInstalled :

boolean

}; type

LoadingAction

=

{

type :

"loading"

}; type

IdleAction

=

{

type :

"idle"

}; type

Action

=

|

ConnectAction

|

DisconnectAction

|

PageLoadedAction

|

LoadingAction

|

IdleAction ; type

Dispatch

=

(action :

Action)

=>

void ; type

Status

=

"loading"

|

"idle"

|

"pageNotLoaded" ; type

State

=

{

wallet :

string

|

null ;

isMetamaskInstalled :

boolean ;

status :

Status ;

balance :

string

|

null ; }; const

initialState :

State

=

{

wallet :

null ,

isMetamaskInstalled :

false ,

status :

"loading" ,

balance :

null , }

as

```
const ; function
metamaskReducer ( state :
State ,
action :
Action ) :
State
{
switch
( action . type )
{
case
"connect" :
{
const
{
wallet ,
balance
}
=
action ;
return
{
... state ,
wallet ,
balance ,
status :
"idle"
};
}
case
"disconnect" :
{
return
{
... state ,
wallet :
null ,
```

```
balance :
null
};
}
case
"pageLoaded" :
{
const
{
isMetamaskInstalled
}
=
action ;
return
{
... state ,
isMetamaskInstalled ,
status :
"idle"
};
}
case
"loading" :
{
return
{
... state ,
status :
"loading"
};
}
case
"idle" :
{
return
{
... state ,
```

```
status :
"idle"
};
}
default :
{
throw
new
Error ( "Unhandled action type" );
}
} } const
MetamaskContext
=
React . createContext <
{
state :
State ;
dispatch :
Dispatch
}
|
undefined
        ( undefined ); function
MetamaskProvider ({
children
} :
PropsWithChildren )
{
const
[ state ,
dispatch ]
=
React . useReducer ( metamaskReducer ,
initialState );
const
value
=
```

```
{
state ,
dispatch
};
useEffect (()
=>
{
if
( typeof
window
!==
undefined )
{
// start by checking if window.ethereum is present, indicating a wallet extension
const
ethereumProviderInjected
=
typeof
window . ethereum
!==
"undefined" ;
// this could be other wallets so we can verify if we are dealing with metamask
// using the boolean constructor to be explecit and not let this be used as a falsy value (optional)
const
isMetamaskInstalled
=
ethereumProviderInjected
&&
Boolean ( window . ethereum . isMetaMask );
dispatch ({
type :
"pageLoaded" ,
isMetamaskInstalled
});
}
},
[]);
```

return

(

< MetamaskContext . Provider

value

{ value }

{ children }

< /MetamaskContext.Provider>

); } function

useMetamask ()

{

const

context

=

React . useContext (MetamaskContext);

if

(context

===

undefined)

{

throw

new

Error ("useMetamask must be used within a MetamaskProvider");

}

return

context ; } export

{

MetamaskProvider ,

useMetamask

}; Updatecomponents/Wallet.tsx to:

import

Link

from

"next/link" ; import

{

useMetamask

}

from

"../hooks/useMetamask" ; import

{

Loading

}

from

"./Loading" ; export

default

function

Wallet ()

{

const

{

dispatch ,

state :

{

status ,

isMetamaskInstalled ,

wallet ,

balance

},

}

=

useMetamask ();

const

showInstallMetamask

=

status

!==

"pageNotLoaded"

&&

! isMetamaskInstalled ;

const

showConnectButton

=

status

!==

"pageNotLoaded"

```
&&
isMetamaskInstalled
&&
! wallet ;
const
showAddToken
=
status
!==
"pageNotLoaded"
&&
typeof
wallet
===
"string" ;
const
handleConnect
=
async
()
=>
{
dispatch ({
type :
"loading"
});
const
accounts
=
await
window . ethereum . request ({
method :
"eth_requestAccounts" ,
});
if
( accounts . length
0 )
```

```
{
const
balance
=
await
window . ethereum ! . request ({
method :
"eth_getBalance" ,
params :
[ accounts [ 0 ],
"latest" ],
});
dispatch ({
type :
"connect" ,
wallet :
accounts [ 0 ],
balance
});
// we can register an event listener for changes to the users wallet
window . ethereum . on ( "accountsChanged" ,
async
( newAccounts :
string [])
=>
{
if
( newAccounts . length
0 )
{
// uppon receiving a new wallet, we'll request again the balance to synchronize the UI.
const
newBalance
=
await
window . ethereum ! . request ({
method :
```

```
"eth_getBalance" ,
params :
[ newAccounts [ 0 ],
"latest" ],
});
dispatch ({
type :
"connect" ,
wallet :
newAccounts [ 0 ],
balance :
newBalance ,
});
}
else
{
// if the length is 0, then the user has disconnected from the wallet UI
dispatch ({
type :
"disconnect"
});
}
});
}
};
const
handleAddUsdc
=
async
()
=>
{
dispatch ({
type :
"loading"
});
await
```

window . ethereum . request ({

method :

"wallet_watchAsset" ,

params :

{

type :

"ERC20" ,

options :

{

address :

"0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48" ,

symbol :

"USDC" ,

decimals :

18 ,

image :

"https://cryptologos.cc/logos/usd-coin-usdc-logo.svg?v=023" ,

},

},

});

dispatch ({

type :

"idle"

});

};

return

(

< div

className

"bg-truffle"

< div

className

"mx-auto max-w-2xl py-16 px-4 text-center sm:py-20 sm:px-6 lg:px-8"

< h2

className

"text-3xl font-bold tracking-tight text-white sm:text-4xl"

< span

className

"block"

Metamask

API

intro < /span>

< /h2>

< p

className

"mt-4 text-lg leading-6 text-white"

Follow

along

with

the { " " }

< Link

href

"https://github.com/GuiBibeau/web3-unleashed-demo"

target

"_blank"

< span

className

"underline cursor-pointer"

Repo < /span>

< /Link>{" "}

in

order

to

learn

how

to

use

the

Metamask

API .

< /p>

{ wallet

&&

balance

&&

(

< div

className

" px-4 py-5 sm:px-6"

< div

className

"-ml-4 -mt-4 flex flex-wrap items-center justify-between sm:flex-nowrap"

< div

className

"ml-4 mt-4"

< div

className

"flex items-center"

< div

className

"ml-4"

< h3

className

"text-lg font-medium leading-6 text-white"

Address :

< span

{ wallet } < /span>

< /h3>

< p

className

"text-sm text-white"

Balance : { " " }

< span

{(parseInt (balance)

/

1000000000000000000). toFixed (4)}{ " " }

ETH

< /span>

< /p>

< /div>

< /div>

< /div>

< /div>

< /div>

)}

{ showConnectButton

&&

(

< button

onClick

{ handleConnect }

className

"mt-8 inline-flex w-full items-center justify-center rounded-md border border-transparent bg-ganache text-white px-5 py-3 text-base font-medium sm:w-auto"

{ status

===

"loading"

?

< Loading

/>

:

"Connect Wallet" }

< /button>

)}

{ showInstallMetamask

&&

(

< Link

href

"https://metamask.io/"

target

"_blank"

< a

className

"mt-8 inline-flex w-full items-center justify-center rounded-md border border-transparent bg-ganache text-white px-5 py-3 text-base font-medium sm:w-auto"

Install

Metamask

< /a>

< /Link>

)}

{ showAddToken

&&

(

< button

onClick

{ handleAddUsdc }

className

"mt-8 inline-flex w-full items-center justify-center rounded-md border border-transparent bg-ganache text-white px-5 py-3 text-base font-medium sm:w-auto"

{ status

===

"loading"

?

< Loading

/>

:

"Add Token" }

< /button>

)}

< /div>

< /div>

); } With those changes in place we can now install, connect to, and view information from our MetaMask wallet. We can also see a nicely formatted version of our ETH balance and we can seeUSDC tokens in our wallet.

Checkout the Diff to see what changed

We have one more UX improvement to push to our dapp.

More UX Goodies¶

We'd like to store some MetaMask state in the browser's local storage to help us create a Disconnect button, something that we feel makes the UX better in a dapp. We will register an event listener for changes to the user's wallet, so that when connecting and disconnecting the UX is just a little bit better. We will add a custom React Hook calleduseListen to help us achieve this and to co-locate some code that would otherwise be added in two different components so that our final code is a bit cleaner. We do a small refactor to get rid of auseEffect and we will display our buttons side by side when we have more than one showing on the page (Disconnect & Add Tokens) and we will use Tailwind's flex-box options to make this easy.

Updatehooks/useMetamask.tsx

import

React ,

{

useEffect ,

type

PropsWithChildren

}

from

"react" ; type

ConnectAction

=

{

type :

"connect" ;

wallet :

string ;

balance :

string

}; type

DisconnectAction

=

{

type :

"disconnect"

}; type

PageLoadedAction

=

{

type :

"pageLoaded" ;

isMetamaskInstalled :

boolean ;

wallet :

string

|

null ;

balance :

string

|

null ; }; type

LoadingAction

=

{

type :

"loading"

}; type

IdleAction

=

{

type :

"idle"

}; type

Action

=

|

ConnectAction

|

DisconnectAction

|

PageLoadedAction

|

LoadingAction

|

IdleAction ; type

Dispatch

=

(action :

Action)

=>

void ; type

Status

=

"loading"

|

"idle"

|

"pageNotLoaded" ; type

State

=

{

wallet :

string

|

null ;

isMetamaskInstalled :

boolean ;

status :

Status ;

balance :

string

|

null ; }; const

initialState :

State

=

{

wallet :

null ,

isMetamaskInstalled :

false ,

```
status :
"loading" ,
balance :
null , }
as
const ; function
metamaskReducer ( state :
State ,
action :
Action ) :
State
{
switch
( action . type )
{
case
"connect" :
{
const
{
wallet ,
balance
}
=
action ;
const
newState
=
{
... state ,
wallet ,
balance ,
status :
"idle"
}
as
State ;
```

```
const
info
=
JSON . stringify ( newState );
window . localStorage . setItem ( "metamaskState" ,
info );
return
newState ;
}
case
"disconnect" :
{
window . localStorage . removeItem ( "metamaskState" );
return
{
... state ,
wallet :
null ,
balance :
null
};
}
case
"pageLoaded" :
{
const
{
isMetamaskInstalled ,
balance ,
wallet
}
=
action ;
return
{
... state ,
isMetamaskInstalled ,
```

```
status :
"idle" ,
wallet ,
balance
};
}
case
"loading" :
{
return
{
... state ,
status :
"loading"
};
}
case
"idle" :
{
return
{
... state ,
status :
"idle"
};
}
default :
{
throw
new
Error ( "Unhandled action type" );
}
} } const
MetamaskContext
=
React . createContext <
{
```

state :

State ;

dispatch :

Dispatch

}

|

undefined

(undefined); function

MetamaskProvider ({

children

} :

PropsWithChildren)

{

const

[state ,

dispatch]

=

React . useReducer (metamaskReducer ,

initialState);

const

value

=

{

state ,

dispatch

};

return

(

< MetamaskContext . Provider

value

{ value }

{ children }

< /MetamaskContext.Provider>

); } function

useMetamask ()

{

const

```
context
=
React . useContext ( MetamaskContext );
if
( context
===
undefined )
{
throw
new
Error ( "useMetamask must be used within a MetamaskProvider" );
}
return
context ; } export
{
MetamaskProvider ,
useMetamask
}; Createhooks/useListen.tsx
import
{
useMetamask
}
from
"./useMetamask" ; export
const
useListen
=
()
=>
{
const
{
dispatch
}
=
useMetamask ();
return
```

```
()
=>
{
window . ethereum . on ( "accountsChanged" ,
async
( newAccounts :
string [])
=>
{
if
( newAccounts . length
0 )
{
// uppon receiving a new wallet, we'll request again the balance to synchronize the UI.
const
newBalance
=
await
window . ethereum ! . request ({
method :
"eth_getBalance" ,
params :
[ newAccounts [ 0 ],
"latest" ],
});
dispatch ({
type :
"connect" ,
wallet :
newAccounts [ 0 ],
balance :
newBalance ,
});
}
else
{
// if the length is 0, then the user has disconnected from the wallet UI
```

```
dispatch ({
type :
"disconnect"
});
}
});
}; }; Updatecomponents/Wallet.tsx
import
Link
from
"next/link" ; import
{
useListen
}
from
"../hooks/useListen" ; import
{
useMetamask
}
from
"../hooks/useMetamask" ; import
{
Loading
}
from
"./Loading" ; export
default
function
Wallet ()
{
const
{
dispatch ,
state :
{
status ,
isMetamaskInstalled ,
```

```
wallet ,
balance
},
}
=
useMetamask ();
const
listen
=
useListen ();
const
showInstallMetamask
=
status
!==
"pageNotLoaded"
&&
! isMetamaskInstalled ;
const
showConnectButton
=
status
!==
"pageNotLoaded"
&&
isMetamaskInstalled
&&
! wallet ;
const
isConnected
=
status
!==
"pageNotLoaded"
&&
typeof
wallet
```

```
===
"string" ;
const
handleConnect
=
async
()
=>
{
dispatch ({
type :
"loading"
});
const
accounts
=
await
window . ethereum . request ({
method :
"eth_requestAccounts" ,
});
if
( accounts . length
0 )
{
const
balance
=
await
window . ethereum ! . request ({
method :
"eth_getBalance" ,
params :
[ accounts [ 0 ],
"latest" ],
});
dispatch ({
```

```
type :
"connect" ,
wallet :
accounts [ 0 ],
balance
});
// we can register an event listener for changes to the users wallet
listen ();
}
};
const
handleDisconnect
=
()
=>
{
dispatch ({
type :
"disconnect"
});
};
const
handleAddUsdc
=
async
()
=>
{
dispatch ({
type :
"loading"
});
await
window . ethereum . request ({
method :
"wallet_watchAsset" ,
params :
```

{

type :

"ERC20" ,

options :

{

address :

"0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48" ,

symbol :

"USDC" ,

decimals :

18 ,

image :

"https://cryptologos.cc/logos/usd-coin-usdc-logo.svg?v=023" ,

},

},

});

dispatch ({

type :

"idle"

});

};

return

(

< div

className

"bg-truffle"

< div

className

"mx-auto max-w-2xl py-16 px-4 text-center sm:py-20 sm:px-6 lg:px-8"

< h2

className

"text-3xl font-bold tracking-tight text-white sm:text-4xl"

< span

className

"block"

Metamask

API

intro < /span>

< /h2>

< p

className

"mt-4 text-lg leading-6 text-white"

Follow

along

with

the { " " }

< Link

href

"https://github.com/GuiBibeau/web3-unleashed-demo"

target

"_blank"

< span

className

"underline cursor-pointer"

Repo < /span>

< /Link>{" "}

in

order

to

learn

how

to

use

the

Metamask

API .

< /p>

{ wallet

&&

balance

&&

(

< div

className

" px-4 py-5 sm:px-6"

< div

className

"-ml-4 -mt-4 flex flex-wrap items-center justify-between sm:flex-nowrap"

< div

className

"ml-4 mt-4"

< div

className

"flex items-center"

< div

className

"ml-4"

< h3

className

"text-lg font-medium leading-6 text-white"

Address :

< span

{ wallet } < /span>

< /h3>

< p

className

"text-sm text-white"

Balance : { " " }

< span

{(parseInt (balance)

/

1000000000000000000). toFixed (4)}{ " " }

ETH

< /span>

< /p>

< /div>

< /div>

< /div>

< /div>

< /div>

)}

{ showConnectButton

&&

(

< button

onClick

{ handleConnect }

className

"mt-8 inline-flex w-full items-center justify-center rounded-md border border-transparent bg-ganache text-white px-5 py-3 text-base font-medium sm:w-auto"

{ status

===

"loading"

?

< Loading

/>

:

"Connect Wallet" }

< /button>

)}

{ showInstallMetamask

&&

(

< Link

href

"https://metamask.io/"

target

"_blank"

< a

className

"mt-8 inline-flex w-full items-center justify-center rounded-md border border-transparent bg-ganache text-white px-5 py-3 text-base font-medium sm:w-auto"

Install

Metamask

< /a>

< /Link>

)}

{ isConnected

&&

(

< div

className

"flex w-full justify-center space-x-2"

< button

onClick

{ handleAddUsdc }

className

"mt-8 inline-flex w-full items-center justify-center rounded-md border border-transparent bg-ganache text-white px-5 py-3 text-base font-medium sm:w-auto"

{ status

===

"loading"

?

< Loading

/>

:

"Add Token" }

< /button>

< button

onClick

{ handleDisconnect }

className

"mt-8 inline-flex w-full items-center justify-center rounded-md border border-transparent bg-ganache text-white px-5 py-3 text-base font-medium sm:w-auto"

Disconnect

< /button>

< /div>

)}

< /div>

< /div>

); } Finally, we will update ourpages/index.tsx file with auseEffect to wrap all of these final changes up.

Updatepages/index/tsx

import

type

{

NextPage

}

from

"next" ; import

{

useEffect

}

from

"react" ; import

Wallet

from

"../components/Wallet" ; import

{

useListen

}

from

"../hooks/useListen" ; import

{

useMetamask

}

```
from
"../hooks/useMetamask" ; const
Home :
NextPage
=
()
=>
{
const
{
dispatch
}
=
useMetamask ();
const
listen
=
useListen ();
useEffect (()
=>
{
if
( typeof
window
!==
undefined )
{
// start by checking if window.ethereum is present, indicating a wallet extension
const
ethereumProviderInjected
=
typeof
window . ethereum
!==
"undefined" ;
// this could be other wallets so we can verify if we are dealing with metamask
// using the boolean constructor to be explecit and not let this be used as a falsy value (optional)
```

```
const
isMetamaskInstalled
=
ethereumProviderInjected
&&
Boolean ( window . ethereum . isMetaMask );
const
local
=
window . localStorage . getItem ( "metamaskState" );
// user was previously connected, start listening to MM
if
( local )
{
listen ();
}
// local could be null if not present in LocalStorage
const
{
wallet ,
balance
}
=
local
?
JSON . parse ( local )
:
// backup if local storage is empty
{
wallet :
null ,
balance :
null
};
dispatch ({
type :
"pageLoaded" ,
```

isMetamaskInstalled ,

wallet ,

balance

});

}

},

[]);

return

(

<>

< Wallet

/>

< />

); }; export

default

Home ; In this last page update topages/index.tsx , we have relocated theuseEffect from thehooks/useMetaMask.tsx page. This hook is consumingdispatch , so the proper next step would be to create a layout page with NextJS, but, since we only have one page, we simply added this code here.

We have updated ourhooks/useMetamask.tsx page'sPageLoadAction to includewallet andbalance as well as the code required to access our local storage and rehydrate our app.

With those changes in place we have also updated ourcomponents/Wallet.tsx page to use ouruseListen hook since we are using that code in multiple places now, updated theshowAddToken variable to a more descriptive name ofisConnected , and added ahandleDisconnect() function to dispatch an action clearing local storage in our browser.

This also required a slight update to our JSX/HTML to display our buttons more neatly.

Checkout the Diff to see what changed

Remove Listeners after Disconnect¶

We have one final change we want to make to ensure that we stop listening to changes once the user has disconnected their wallet.

We will update theand files. This will make the TypeScript definitions file aware of theremoveAllListeners() method we will be using, as well as adding the necessary code to thedisconnect case inside themetamaskReducer .

Updatetypes.d.ts file:

type

InjectedProviders

=

{

isMetaMask? :

true ; }; interface

Window

{

ethereum :

InjectedProviders

&

{

on :

(...args :

any [])

=>

void ;

removeListener :

(...args :

any [])

=>

void ;

removeAllListeners :

(...args :

any [])

=>

void ;

request < T

=

any

(args :

any) :

Promise < T

;

}; } Update the case statement in theuseMetamask.tsx file to:

case

"disconnect" :

{

window . localStorage . removeItem ("metamaskState");

if

(typeof

window . ethereum

!==

undefined)

{

window . ethereum . removeAllListeners (["accountsChanged"]);

}

return

{

... state ,

wallet :

null ,

balance :

null

};

} Again, here we have ensured that all listeners added after connecting the wallet stop listening once the user is disconnected.

You can always switch to the final branch of this repo to get to the completed state of this demo.

This concludes the demo! But you're just getting started; for a challenge, try updating the UI, try to add functionality to switch chains, and overall, have fun. If you have any questions or need help with MetaMask, reach out to our DevRel team on Twitter. You can contact Gui Bibeau and Eric Bishard with any questions or feedback.

One final note, Gui has a great resource and blog called frontend-devops where he waxes poetically about web, full stack development, and UX which. A great resource for Web2 developers getting into Web3 and seasoned developers alike!

Connect with us¶

If you want to talk about this content, make suggestions for what you'd like to see, or ask questions about the series, join our Discord ! See other episodes here . Lastly, don't forget to follow us on Twitter for the latest updates on all things Truffle.