



# Lesson 8

HTML / Javascript Fund Me

Hikmah Nisya - 1103184094

Radzis Araaf Jaya Jamaludin - 1103184234

Raudhatul Rafiqah Assyahiddini - 1103180225

---



# Requirement

---

- git

You'll know you've installed it right if you can run: `git --version`

---

- Metamask

This is a browser extension that lets you interact with the blockchain.

---

- Nodejs

You'll know you've installed nodejs right if you can run: `node --version` And get an output like: `vx.x.x`

---

- Yarn instead of npm

You'll know you've installed yarn right if you can run: `yarn --version` And get an output like: `x.x.x`

You might need to install it with npm

---

# Typescript

- For this demo project, we do not have a typescript edition. Please see the NextJS projects for a professional typescript front end.



# Quickstart

## 1. Clone the repo

```
git clone https://github.com/PatrickAlphaC/html-fund-me-fcc  
cd html-fund-me-fcc
```

## 2. Run the file.

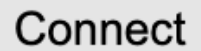
You can usually just double click the file to "run it in the browser". Or you can right click the file in your VSCode and run "open with live server".

Optionally:

If you'd like to run with prettier formatting, or don't have a way to run your file in the browser, run:

```
yarn  
yarn http-server
```

And you should see a small button that says "connect".



Connect

Hit it, and you should see metamask pop up.

# Execute Transaction

If you want to execute a transaction follow this:

Make sure you have the following installed:

1. You'll need to open up a second terminal and run:

```
git clone https://github.com/PatrickAlphaC/hardhat-fund-me-fcc
cd hardhat-fund-me-fcc
yarn
yarn hardhat node
```

This will deploy a sample contract and start a local hardhat blockchain.

2. Update your `constants.js` with the new contract address.

In your `constants.js` file, update the variable `contractAddress` with the address of the deployed "FundMe" contract. You'll see it near the top of the hardhat output.

3. Connect your [metamask](#) to your local hardhat blockchain.

PLEASE USE A METAMASK ACCOUNT THAT ISNT ASSOCIATED WITH ANY REAL MONEY. I usually use a few different browser profiles to separate my metamasks easily.

In the output of the above command, take one of the private key accounts and [import it into your metamask](#).

Additionally, add your localhost with chainid 31337 to your metamask.

5. Reserve the front end with `yarn http-server`, input an amount in the text box, and hit `fund` button after connecting

# Connnecting to MetaMask

Connecting" or "logging in" to MetaMask effectively means "to access the user's Ethereum account(s)".

You should only initiate a connection request in response to direct user action, such as clicking a button. You should always disable the "connect" button while the connection request is pending. You should never initiate a connection request on page load.

We recommend that you provide a button to allow the user to connect MetaMask to your dapp. Clicking this button should call the following method:

# Connnecting to MetaMask

```
ethereum.request({ method: 'eth_requestAccounts' });
```

js

```
const ethereumButton = document.querySelector('.enableEthereumButton');  
  
ethereumButton.addEventListener('click', () => {  
  //Will Start the metamask extension  
  ethereum.request({ method: 'eth_requestAccounts' });  
});
```

js



# Connnecting to MetaMask

- This promise-returning function resolves with an array of hex-prefixed Ethereum addresses, which can be used as general account references when sending transactions.
- Over time, this method is intended to grow to include various additional parameters to help your site request everything it needs from the user during setup.
- Since it returns a promise, if you're in an async function, you may log in like this:

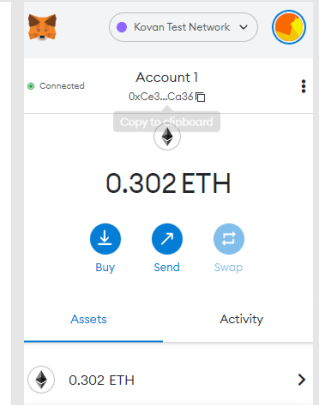
```
const accounts = await ethereum.request({ method: 'eth_requestAccounts' });  
const account = accounts[0];  
// We currently only ever provide a single account,  
// but the array gives us some room to grow.
```

# Example

Example:

Enable Ethereum

Account: 0xce325a533e37fb9683a894a6537708969489ca36






# RESET ACCOUNT

- MetaMask - RPC Error:
- [ethjs-query] while formatting outputs from RPC  
`'{"value":{"code":-32603,"data":{"code":-32000,"message":"Nonce too high. Expected nonce to be 2 but got 4. Note that transactions can't be queued when automining."}}}'`

# LISTENING FOR EVENT AND COMPLETE TRANSACTION

 `provider.on( eventName , listener ) ⇒ this`

Add a *listener* to be triggered for each *eventName* [event](#).

`provider.once( eventName , listener ) ⇒ this`

Add a *listener* to be triggered for only the next *eventName* [event](#), at which time it will be removed.

`provider.emit( eventName , ...args ) ⇒ boolean`

Notify all listeners of the *eventName* [event](#), passing *args* to each listener. This is generally only used internally.

`provider.off( eventName [ , listener ] ) ⇒ this`

Remove a *listener* for the *eventName* [event](#). If no *listener* is provided, all listeners for *eventName* are removed.

`provider.removeAllListeners( [ eventName ] ) ⇒ this`

Remove all the listeners for the *eventName* [events](#). If no *eventName* is provided, **all** events are removed.

`provider.listenerCount( [ eventName ] ) ⇒ number`

Returns the number of listeners for the *eventName* [events](#). If no *eventName* is provided, the total number of listeners is returned.

`provider.listeners( eventName ) ⇒ Array< Listener >`

Returns the list of Listeners for the *eventName* [events](#).

# LISTENING FOR EVENT AND COMPLETE TRANSACTION

## Syntax:

The below enlightened syntax illustrates the declaration of anonymous function using normal declaration:

```
function() {  
    // Function Body  
}
```

We may also declare anonymous function using arrow function technique which is shown below:

```
( () => {  
    // Function Body...  
} )();
```

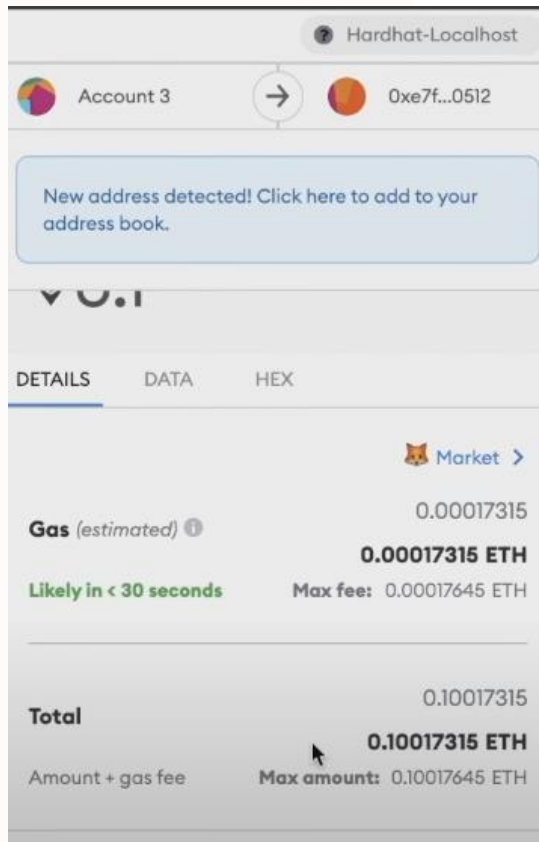
**Example 1:** In this example, we define an anonymous function that prints a message to the console. The function is then stored in the *greet* variable. We can call the function by invoking *greet()*.

## Javascript

```
<script>  
var greet = function () {  
    console.log("Welcome to GeeksforGeeks!");  
};  
  
greet();  
</script>
```

## Output:

```
Welcome to GeeksforGeeks!
```



```
Funding with 0.1... bundle.js:145
Mining bundle.js:191
0x6398c5e075bc25e63630931ab8985041dec665c
69dd2c047087164f9ed3c061e...
Completed with 1 bundle.js:195
confirmations.
Done! bundle.js:160
> window.ethereum
< Proxy {_events: {...}, _eventsCount: 1, _m...
```

# SUMMARY