

LESSON8 HTML - JAVASCRIPT FUND ME - intro to front end -
fullstack 12.32.57

UYGULAMAYI TEST ETMEK İÇİN FUNDME.SOL CONTRACTI KULLANILDI.

<https://github.com/celalaksu/hardhat-fund-me-cc>

HARDHAT YEREL AĞ (LOCALHOST) TA DEPLOY EDİLELEREK KULLANILDI. Test etmek için gerekli kurulumlar yapılmalıdır.

Kurulumlara yukarıda verilen github reposundaki zip dosyasının içinde bulunan notlardan ulaşabilirsiniz.

```
mkdir html-fund-me-fcc
```

```
cd html-fund-me-fcc  
code .
```

index.html OLUŞTUR

! KARAKTERİNE BASINCA VSCODE BASİT HTML KODLARINI OTOMATİK EKLER.

Live Server - Ritwick Dey EKLENTİSİNİ KUR

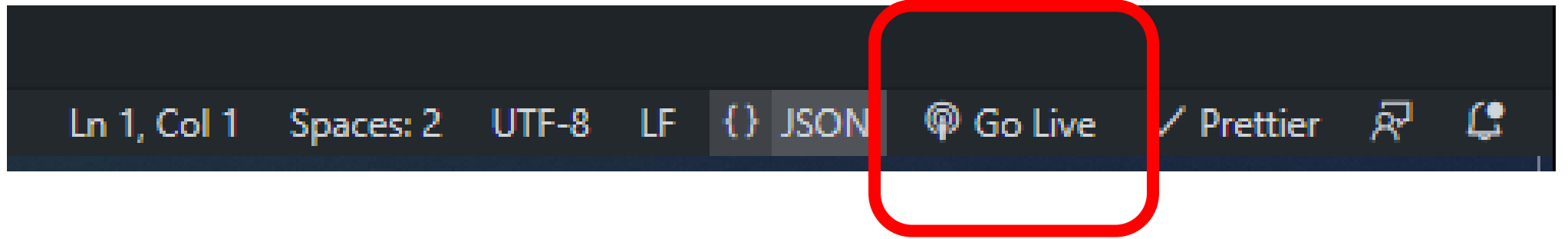
BU HTML SAYFALARINI ÖNİZLEMELİK İÇİN KULLANILIR. KISA YOLU ALT + L VE ALT + O DUR.

<http://127.0.0.1:5500/index.html> ŞEKLİNDE AÇILIR.

CTRL + SHIFT + P ile OPEN WITH LIVE SERVER komutu ile de önizleme yapılabilir.

yarn add --dev http-server KOMUTU İLE HTTP SERVER İÇİN GEREKLİ DOSYALAR YÜKLENİR.

VS CODE SAĞ ALT TARAFTAN SUNUC İŞLEMLERİ YAPILIR.



yarn http-server İLE DE SUNUCU BAŞLATILABİLİR.

CONNECTING HTML TO METAMASK 12.50.11

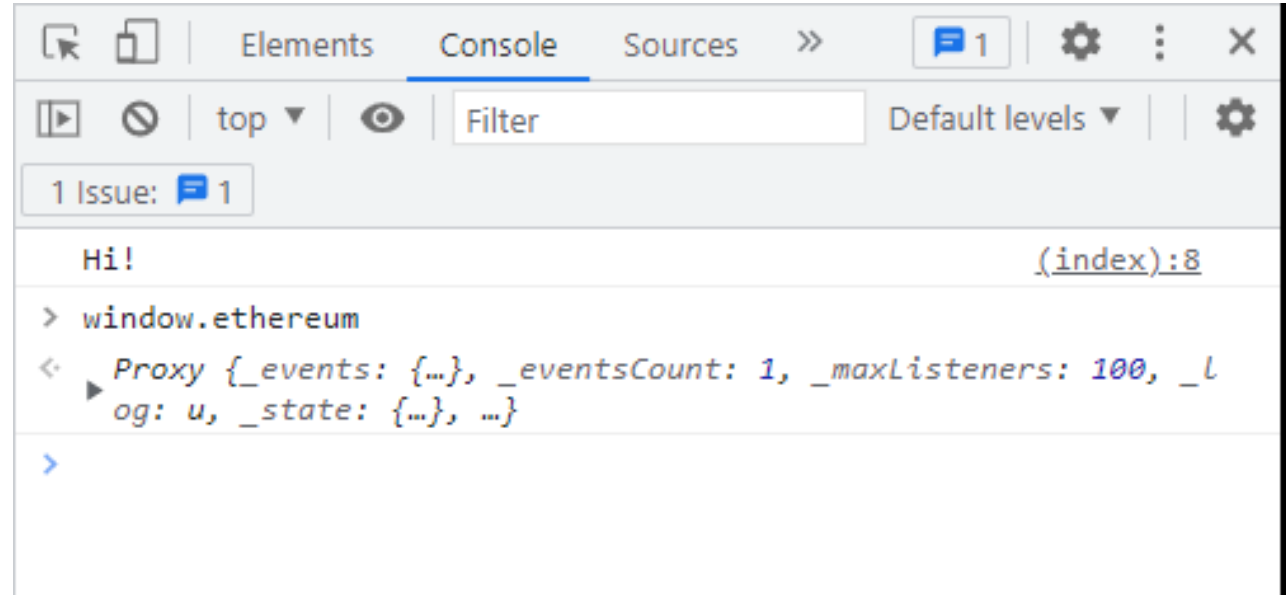
```
<body>
  <script>
    console.log("Hi!");
  </script>
</body>
```

Tarayıcıda SAĞ TIKLA - İNCELE - CONSOLE kısmında çıktı gözükür

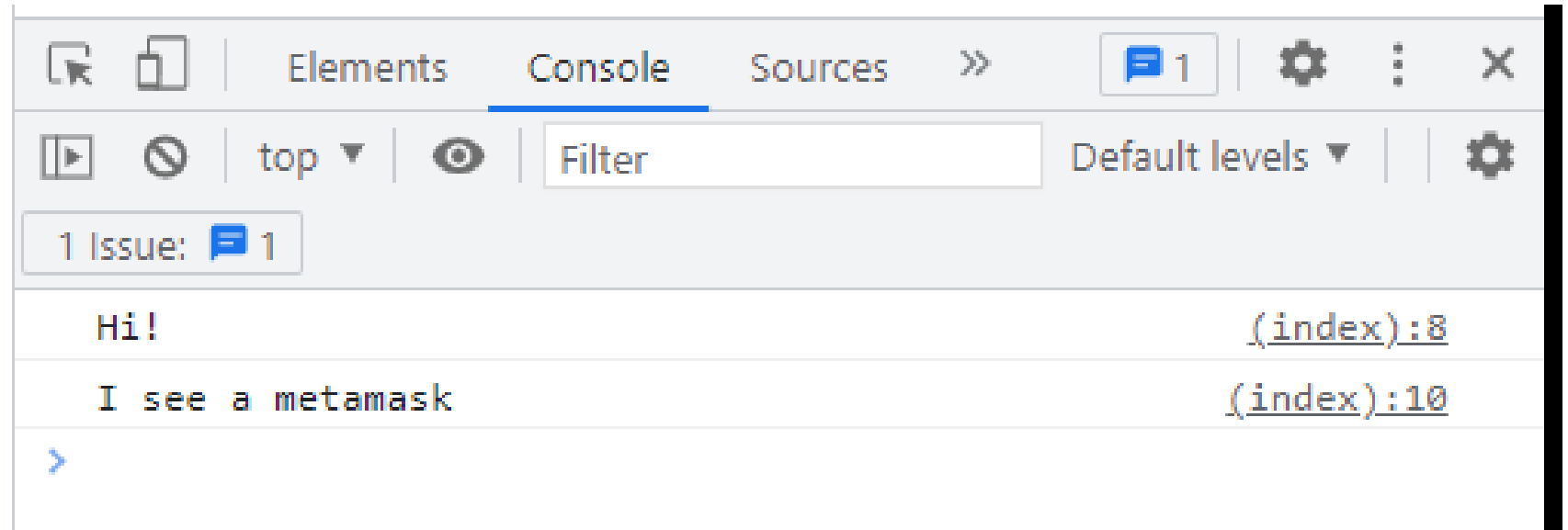
KONSOLE DA JAVASCRIPT KODLARI YAZARAK KONTROLLER YAPILABİLİR.

window.ethereum

<https://docs.metamask.io/guide/>



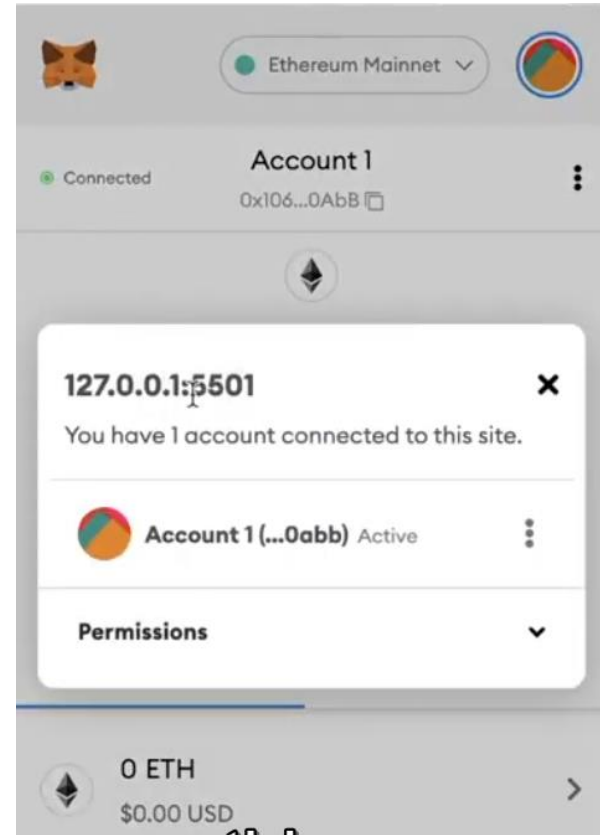
```
<body>
  <script>
    console.log("Hi!");
    if (typeof window.ethereum !==
"undefined") {
      console.log("I see a metamask");
    } else {
      console.log("No metamask");
    }
  </script>
</body>
```



METAMASK A BAĞLANMAK İÇİN

```
if (typeof window.ethereum !== "undefined") {  
    window.ethereum.request({ method: "eth_requestAccounts" });  
}
```

Sayfa yenilenince otomatik olarak metamask açılır.
Bunu fonk içine alıp gerekli yerde çağırmamız gerekir.




```
<body>
  <script>
    async function connect() {
      console.log("Hi!");
      if (typeof window.ethereum !== "undefined") {
        console.log("I see a metamask");
        await window.ethereum.request({ method: "eth_requestAccounts" });
        console.log("Connected");
      } else {
        console.log("No metamask");
      }
    }
  </script>
  <button id="connectButton" onclick="connect()">Connect</button>
</body>
```

Bağantıdan sonra butonun durumunu değiştirme

```
    console.log("Connected");  
    document.getElementById("connectButton").innerHTML = "Connected";  
} else {  
    console.log("No metamask");  
    document.getElementById("connectButton").innerHTML =  
        "Install Metamask";  
}
```

JAVASCRIPT IN ITS OWN FILES 12.57.08

İNDEx.JS oluştur ve js kodlarını oraya taşı.

```
<body>  
  <script src="./index.js" type="text/javascript"></script>
```

ES6 (FRONT END JS) VS NODEJS 12.59.12

NODE JS İLE JAVASCRIPT ARASINDAKİ FARKLAR.

NODE JS DE
 require() KULLANILIR

JAVASCRIPT TE
 import

FRONT END DE import require dan daha kullanışlıdır.

① README.md X

<> index.html

{ } .prettierrc X

```
1  {
2    "tabWidth": 4,
3    "useTabs": false,
4    "semi": false,
5    "singleQuote": false
6  }
7
```

```
// index.js
async function fund(ethAmount) {
  console.log(`Funding with ${ethAmount}....`)
}
```

```
<button id="fundButon" onclick="fund()">Fund</button>
```

Burada henüz gönderilecek eth miktarı ayarlanmamıştır.

Transaction göndermek için gerekli olanlar

Provider / connection to the blockchain

Signer / wallet / someone with some gas

Contract that we are interacting with

ABI and address

Ether.js ile çalışmak için gerekli olan dosyayı çalışmamıza almamız gerekir. İmport işleri için filan

<https://docs.ethers.io/v5/getting-started/> adresinden kullanımına ulaşabiliriz.

<https://cdn.ethers.io/lib/ethers-5.6.esm.min.js> dosyasını projeye aktar.

```
// index.js
import { ethers } from "../ethers-5.6.esm.min.js"
```

Ether.js front end verisiyonudur

Node.js de bu işlem yarn komutu ile birlikte paket ekleyerek yapılmaktadır.

Bu işlemden sonra index.html de ;

```
<script src="../index.js" type="module"></script>
```

Type kısmını modüle olarak değiştirmeliyiz. Bu bize farklı modülleri import yapmamızı sağlamaktadır.

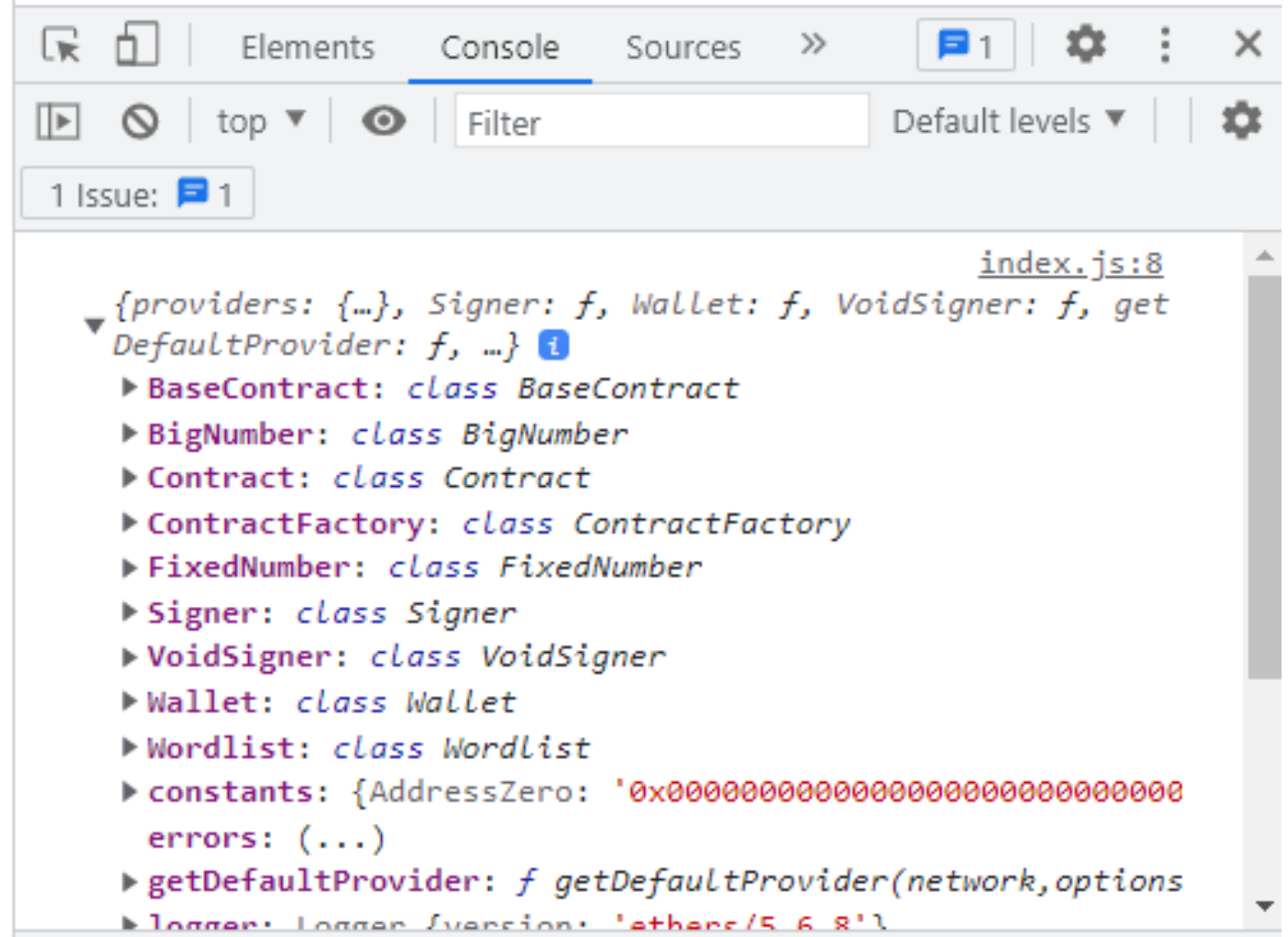
Bu durumda buttonlarda çalışmayacaktır. Bunların index.js de tanımlanmaları gerekir. Ayrıca onclick() metotları da çalışmayacaktır. Bunlarında index.js de atanmaları gerekir.

```
<button id="connectButton">Connect</button>  
<button id="fundButon">Fund</button>
```

```
// index.js  
const connectButton = document.getElementById("connectButton")  
const fundButton = document.getElementById("fundButton")  
connectButton.onclick = connect  
fundButton.onclick = fund
```

```
console.log(ethers)
```

Bağlandığımız metamask ile ilgili bilgileri görebiliriz



SENDING A TRANSACTION FROM A WEBSITE 13.07

```

async function fund(ethAmount) {
  console.log(`Funding with ${ethAmount}....`)
  if (typeof window.ethereum !== "undefined") {
    // metamask tan rpc_url sini alır
    const provider = new ethers.providers.Web3Provider(window.ethereum)
    // metamask ta bağlantı kurulan hesap signer hesabıdır
    const signer = provider.getSigner()
    console.log(signer)
  }
}

```

Fund fonk çalışında yandaki çıktıyı verir.

```

index.js:20
▼ JsonRpcSigner {_isSigner: true, provider: Web3Provider, _
  index: 0, _address: null} ⓘ
  ► provider: Web3Provider {_isProvider: true, _events: Arra
    _address: null
    _index: 0
    _isSigner: true
  ► [[Prototype]]: Signer

```

>

CONTRACT A BAĞLANMA:

ABI VE ADDRESS GEREKLİDİR

BU BİLGİLER İÇİN

constants.js dosyası oluştur ve bilgileri buraya gir.

ABI bilgilerine contracttaki ARTIFACTS/CONTRACTS/FUNDME.SOL/FUNDME.JSON

Dosyasından erişebiliriz.

Buradan köşeli parantezler dahil olacak şekilde kopyala ve constants.js dosyasına

```
export const abi = .....
```

Yapıştır.

Abi index.js ye import et.

```
//index.js  
import { abi } from "../constants.js"
```

CONTRACT ADRESİNİ ALMAK İÇİN

1 - contract ı yeniden deploy ederek alabiliriz.

2 - front end kısmında iken contract klasörüne giderek yarn hardhat node ile bulabiliriz.

Buradan kopyala ve constants.js ye export const contractAddress = "" olarak ekle

İndex.js de import et.

```
// index.js  
import { abi, contractAddress } from "../constants.js"
```

```
// indes.js fund()
```

```
console.log(signer)
const contract = new ethers.Contract(contractAddress, abi, signer)
const transactionResponse = await contract.fund({
  value: ethers.utils.parseEther(ethAmount),
})
```

Bu işlemden sonra fund() butonu tıklanırsa aşağıdaki hata gelir.

Çünkü eth miktarı
Verilmemiştir.

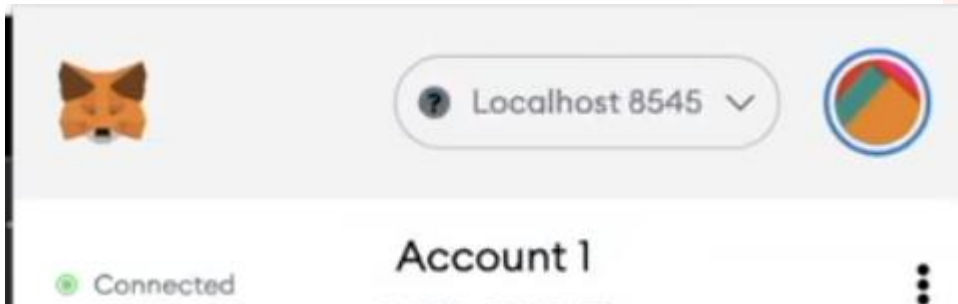
```
✖ ▶ Uncaught (in promise) Error: ethers-5.6.esm.min.js:1
value must be a string (argument="value", value=
{"isTrusted":true}, code=INVALID_ARGUMENT,
version=units/5.6.1)
    at Logger.makeError (ethers-5.6.esm.min.js:1:61080)
    at Logger.throwError (ethers-5.6.esm.min.js:1:61258)
    at Logger.throwArgumentError (ethers-5.6.esm.min.js:1:6
1340)
    at parseUnits (ethers-5.6.esm.min.js:1:525415)
    at Object.parseEther (ethers-5.6.esm.min.js:1:525713)
    at HTMLButtonElement.fund (index.js:34:33)
```

```
// index.js
async function fund() {
  const ethAmount = "77"
```

Bu durumda da hata oluşur.

Bu hatanın sebebi doğru blockchain ağına bağlanmamış olmamızdan kaynaklanmaktadır.

Metamask ta localhost ağına bağlanmamız gerekir.



```
✖ ▶ MetaMask - RPC Error: err: insufficient funds inpage.js:1
for gas * price + value: address
0xD1F68A3b433f02c3640Bb3271faB132bF5A88E4F have
66211616457756040 want 77000000000000000000 (supplied gas
15010499) ▶ Object

✖ ▶ Uncaught (in promise) Error: ethers-5.6.esm.min.js:1
insufficient funds for intrinsic transaction cost [ See: ht
tps://links.ethers.org/v5-errors-INSUFFICIENT_FUNDS ]
(error={"code":-32000,"message":"err: insufficient funds
for gas * price + value: address
0xD1F68A3b433f02c3640Bb3271faB132bF5A88E4F have
66211616457756040 want 77000000000000000000 (supplied gas
15010499)"},"method="estimateGas", transaction=
{"from":"0xD1F68A3b433f02c3640Bb3271faB132bF5A88E4F","to":"
0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512","value":
{"type":"BigNumber","hex":"0x042c96f40959140000"},"data":"0
xb60d4288","accessList":null}, code=INSUFFICIENT_FUNDS,
version=providers/5.6.8)
    at Logger.makeError (ethers-5.6.esm.min.js:1:61080)
    at Logger.throwError (ethers-5.6.esm.min.js:1:61258)
    at checkError (ethers-5.6.esm.min.js:1:463526)
    at Web3Provider.<anonymous> (ethers-5.6.esm.min.js:1:47
4486)
```

METAMASK A HARDHAT LOCAL HOST AĞINI EKLEMEMİZ GEREKİR.

RPC URL yi ağı
çalıştırarak
bulabilirsiniz.
(yarn hardhat node)

Ayarlar

Search in settings

- Genel
- Gelişmiş
- Kişiler
- Güvenlik ve Gizlilik
- Uyarılar
- Ağlar
- DeneySEL
- Hakkında

Ağlar > Ağ ekle

Kötü amaçlı bir ağ sağlayıcı blokzincir durumu hakkında yalan söyleyebilir ve ağ aktivitenizi kaydedebilir. Sadece güvendiğiniz özel ağları ekleyin.

Ağ Adı

Hardhat-Localhost

Yeni RPC URL adresi

http://127.0.0.1:8545

Zincir Kimliği

31337

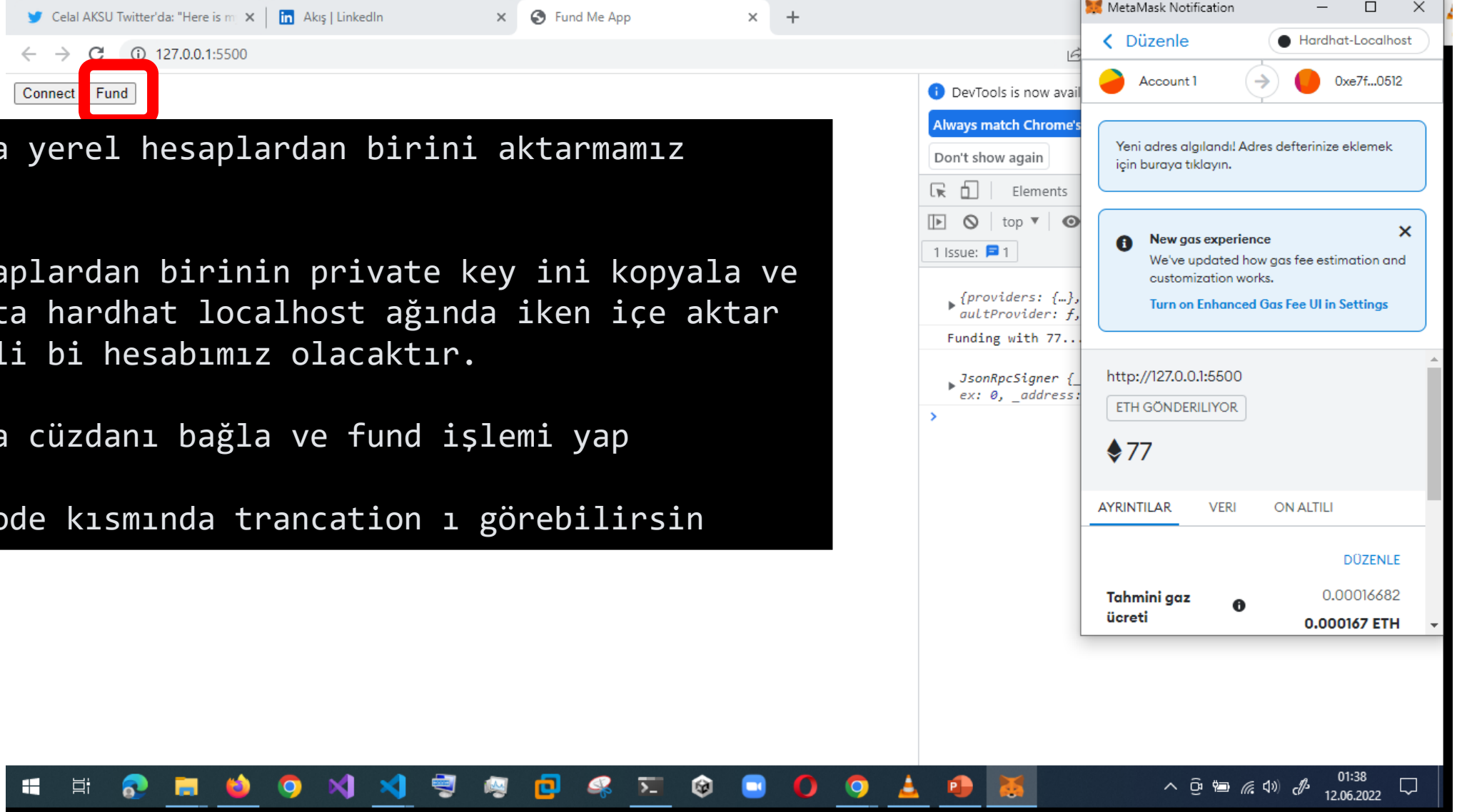
Para Birimi Sembolü

ETH

The network with chain ID 31337 may use a different currency symbol (GO) than the one you have entered. Please verify before continuing.

Blok Gezgini URL Adresi (İsteğe bağlı)

Doğru ağa bağlandıktan sonra fund() fonksiyonu çalışır. Tabi yeterli ETH olmalıdır.



Metamask a yerel hesaplardan birini aktarmamız gerekir.

Yerel hesaplardan birinin private key ini kopyala ve metamask ta hardhat localhost ağında iken içe aktar 1000 eth li bi hesabımız olacaktır.

Sonrasında cüzdanı bağla ve fund işlemi yap

Çalışan node kısmında transaction ı görebilirsin

MetaMask Notification

Düzenle

Hardhat-Localhost

Account 1 → 0xe7f...0512

Yeni adres algılandı! Adres defterinize eklemek için buraya tıklayın.

New gas experience
We've updated how gas fee estimation and customization works.
[Turn on Enhanced Gas Fee UI in Settings](#)

http://127.0.0.1:5500

ETH GÖNDERİLİYOR

77

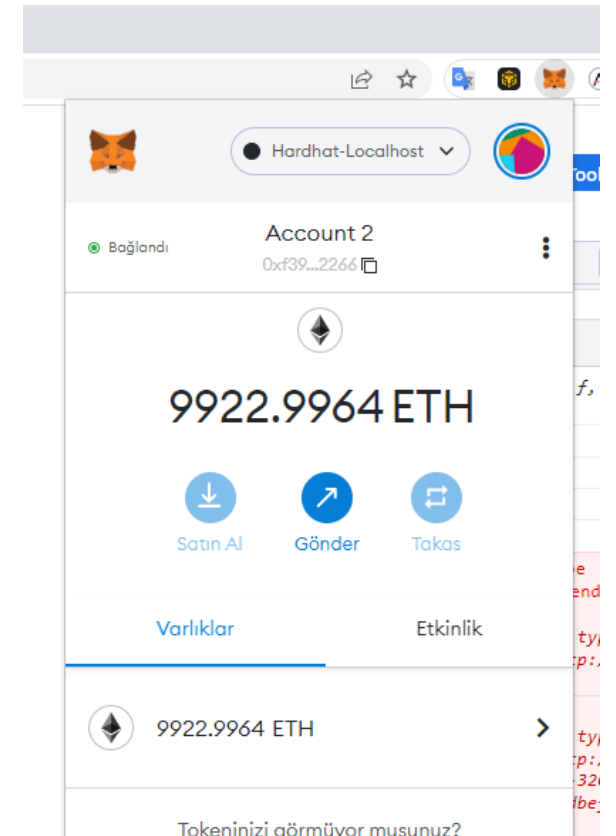
AYRINTILAR VERİ ON ALTILI

Tahmini gaz ücreti 0.00016682
0.000167 ETH

01:38
12.06.2022

eth_sendRawTransaction

Contract call: FundMe#fund
Transaction: 0x71119f2c4b7276eb48dd6b349b1235e6f62ef9f640cebf2a2e464885571a4562
From: 0xf39fd6e51aad88f6f4ce6ab8827279cfff92266
To: 0xe7f1725e7734ce288f8367e1bb143e90bb3f0512
Value: 77 ETH
Gas used: 104466 of 104466
Block #4: 0xedf0997abd23ebad312aac8b02043225e07f2c4a847d9663f89997d2f4b7d079



Hataların kontrol edilmesi.
Index.js fund()

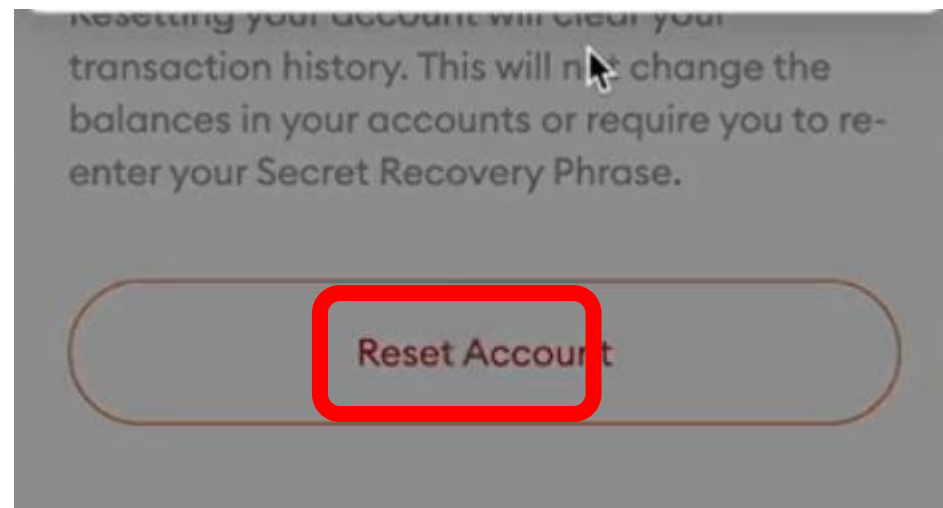
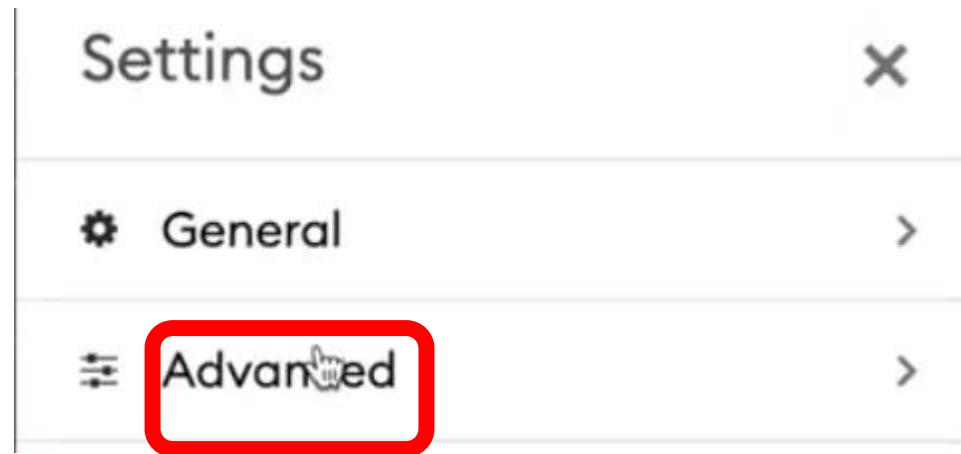
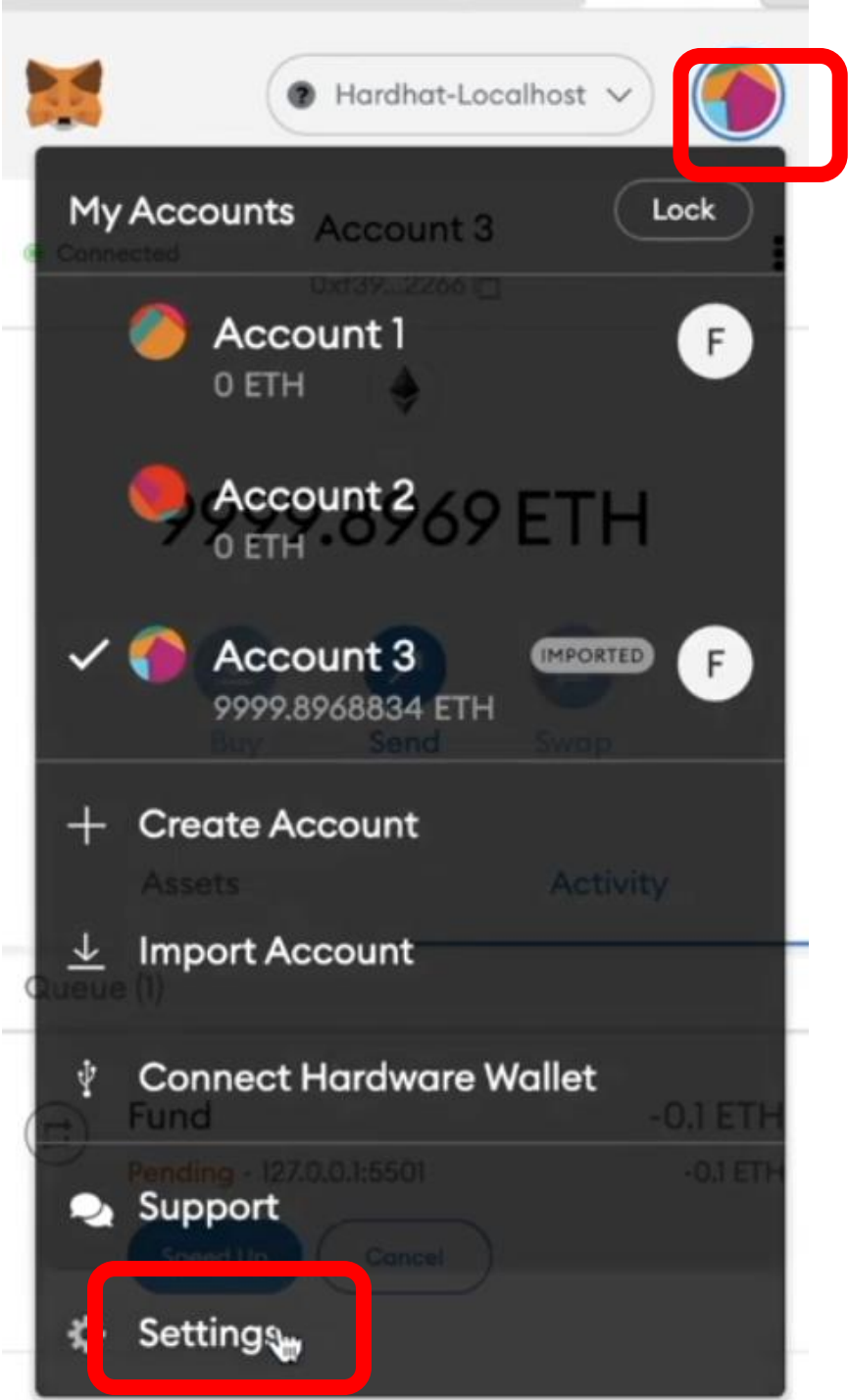
```
console.log(signer)
const contract = new ethers.Contract(contractAddress, abi, signer)
try {
  const transactionResponse = await contract.fund({
    value: ethers.utils.parseEther(ethAmount),
  })
} catch (error) {
  console.log(error)
}
}
```

RESETTING AN ACCOUNT IN METAMASK 13.19.01

Funding with undefined... bundle.js:145

✖ ▼ MetaMask – RPC Error: inpage.js:1
[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."}}}'
{code: -32603, message: `[ethjs-query] w
hile formatting outputs from RPC '{"value':

Bu hatanın gelmesinin sebebi hardhat node unun kapatılmasından dır. Bu durumda metamask hesabının resetlenmesi gerekir.



LISTENING FOR EVENTS AND COMPLETED TRANSACTIONS
13.20.06

Bu kısımda blockchain i dinleyerek işlemi tamamlayacağız.

Yani tx in mined edilip edilmediğini dinleyeceğiz

Yada

Herhangi bir olay için dinleme yapacağız. Bu işlemi event ile yapmıştık.

Tx i dinlemek için bir fonk yazacağız

```
function listenForTransactionMine(transactionResponse, provider){  
  console.log(`Mining ${transactionResponse.hash}...`)  
  // return new Promise()  
  // listen for this transaction to finish  
}
```

Ve bunu fund işleminde çağıracağız. Fund fonkda tx in bitmesini bekleyecek ve sonrasında fonksiyon çalışacaktır.

Index.js / fund()

```
    const transactionResponse = await contract.fund({  
      value: ethers.utils.parseEther(ethAmount),  
    })  
    // hey, wait for this TX to finish  
    await listenForTransactionMine(transactionResponse, provider)  
    console.log("Done!")  
  } catch (error) {
```

Dinleme için `provider.on(eventName, listener)` ile sürekli dinleme yapabiliriz.
Ya da
`provider.once(eventName, listener)` ile sadece bir kez dinleme yapabiliriz.

<https://docs.ethers.io/v5/api/providers/provider/#Provider-once>

İndex.js / `listenfortransactionmine()`

```
provider.once(transactionResponse.hash, () => {})
```

`() => {}` → işlemi listener fonksiyonudur. Ayrı olarak ta yapılabilir. Javascript te ki anonymous fonk yapısı kullanılarak listener fonk oluşturulacak.

```
function listenForTransactionMine(transactionResponse, provider) {
  console.log(`Mining ${transactionResponse.hash}...`)
  // return new Promise()
  // listen for this transaction to finish
  provider.once(transactionResponse.hash, (transactionReceipt) => {
    console.log(
      `Completed with ${transactionReceipt.confirmations} confirmation.`
    )
  })
}
```

Fund işlemi sonucu aşağıdaki gibi olacaktır. (not : yarn hardhat node ve metamask hesabını sıfırlaman gerekebilir.)

Mining 0x8764bcbdd71c08f6b802a320b826c86e795611ffdada5df1bb85abab8885 03278...	index.js:48
Done!	index.js:40
Completed with 1 confirmation.	index.js:52
>	

```
Mining index.js:48  
0x8764bcbd71c08f6b802a320b826c86e795611ffdada5df1bb85abab8885  
03278...
```

```
Done!
```

```
index.js:40
```

```
Completed with 1 confirmation.
```

```
index.js:52
```

Fund işleminde önce listenForTransactionMine fonk çalışmalı sonrasında fund fonk taki işlemlere devam edilmelidir.
Yani "Done" çıktısı sonda olmalıdır. Bu beklemeyi sağlamak için Promise() return etmemiz gerekir.
Promise ile birlikte listen işlemi tamamlandıktan sonra fund işlemlere devam edecektir.

Promise ta iki durum vardır:

resolve() → işlemin başarı ile sonuçlanması

reject() → işlemin tamamlanmaması


```

function listenForTransactionMine(transactionResponse, provider) {
  console.log(`Mining ${transactionResponse.hash}...`)
  // return new Promise()
  // listen for this transaction to finish
  return new Promise((resolve, reject) => {
    provider.once(transactionResponse.hash, (transactionReceipt) => {
      console.log(
        `Completed with ${transactionReceipt.confirmations} confirmation.`
      )
      resolve()
    })
  })
}

```

```

  ▶ JsonRpcSigner {_isSigner: true, provider: Web3Provider, _index: 0, _address: null}

```

```

Mining                                                                    index.js:48
0xe27adc9e28274886bee65d61a5ecc404b8cea89916fb68f4f5455d85b0c
b2d45...

```

```

Completed with 1 confirmation.                                             index.js:53

```

```

Done!                                                                      index.js:40

```

```

>

```

INPUT FORMS 13.30.42

```
// index.html
```

```
<button id="fundButton">Fund</button>  
<!-- FORM -->  
<label for="fund"> ETH Amount </label>  
<input id="ethAmount" placeholder="0.1" />
```

```
// index.js
```

```
async function fund() {  
  const ethAmount = document.getElementById("ethAmount").value  
  console.log(`Funding with ${ethAmount}....`)
```

READING FROM THE BLOCKCHAIN 13.33.33

```
<button id="balanceButton">getBalance</button>
```

```
async function getBalance() {  
  if (typeof window.ethereum !== "undefined") {  
    const provider = new ethers.providers.Web3Provider(window.ethereum)  
    const balance = await provider.getBalance(contractAddress)  
    console.log(ethers.utils.formatEther(balance))  
  }  
}
```

```
const getBanceButton = document.getElementById("balanceButton")  
getBanceButton.onclick = getBalance
```

```
index.js:11  
▶ {providers: {...}, Signer: f, Wallet: f, VoidSigner: f, getDef  
  aultProvider: f, ...}
```

```
462.2
```

```
index.js:30
```

WITHDRAW FUNCTION 13.53.53

```
<button id="withdrawButton">Withdraw</button>
```

```
const withdrawButton = document.getElementById("withdrawButton")  
withdrawButton.onclick = withdraw
```

```
async function withdraw() {  
  if (typeof window.ethereum !== "undefined") {  
    const provider = new ethers.providers.Web3Provider(window.ethereum)  
    const signer = provider.getSigner()  
    const contract = new ethers.Contract(contractAddress, abi, signer)  
    try {  
      const transactionResponse = await contract.withdraw()  
      await listenForTransactionMine(transactionResponse, provider)  
    } catch (error) {  
      console.log(error)  
    }  
  }  
}
```

9999.9957 ETH



Buy



Send



Swap

Withdraw ile bu hesaptan gönderilen bütün eth ler geri alınacağı için hesap full gözükecektir

Hesabın balance ı da 0-sıfır gözükecektir.

0.0

index.js:32



son