Lesson 8 Documentation

Constant.js

```
export const contractAddress = "0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512"
export const abi = [
  {
    inputs: [
      {
        internalType: "address",
        name: "priceFeed",
        type: "address",
      },
    ],
    stateMutability: "nonpayable",
    type: "constructor",
  },
  {
    inputs: [],
    name: "MINIMUM_USD",
    outputs: [
      {
        internalType: "uint256",
        name: "",
        type: "uint256",
      },
    ],
    stateMutability: "view",
    type: "function",
  },
  {
    inputs: [],
    name: "cheaperWithdraw",
    outputs: [],
    stateMutability: "payable",
    type: "function",
  },
```

```
{
  inputs: [],
  name: "fund",
  outputs: [],
  stateMutability: "payable",
  type: "function",
},
{
  inputs: [
    {
      internalType: "address",
      name: "fundingAddress",
      type: "address",
    },
  ],
  name: "getAddressToAmountFunded",
  outputs: [
    {
      internalType: "uint256",
      name: "",
      type: "uint256",
    },
  ],
  stateMutability: "view",
  type: "function",
},
{
  inputs: [
    {
      internalType: "uint256",
      name: "index",
      type: "uint256",
    },
  ],
```

```
  name: "getFunder",
  outputs: [
    {
      internalType: "address",
      name: "",
      type: "address",
    },
  ],
  stateMutability: "view",
  type: "function",
},
{
  inputs: [],
  name: "getOwner",
  outputs: [
    {
      internalType: "address",
      name: "",
      type: "address",
    },
  ],
  stateMutability: "view",
  type: "function",
},
{
  inputs: [],
  name: "getPriceFeed",
  outputs: [
    {
      internalType: "contract AggregatorV3Interface",
      name: "",
      type: "address",
    },
  ],
```

```
    stateMutability: "view",
    type: "function",
  },
  {
    inputs: [],
    name: "getVersion",
    outputs: [
      {
        internalType: "uint256",
        name: "",
        type: "uint256",
      },
    ],
    stateMutability: "view",
    type: "function",
  },
  {
    inputs: [],
    name: "withdraw",
    outputs: [],
    stateMutability: "payable",
    type: "function",
  },
]
```

Index.js

```
import { ethers } from "./ethers-5.6.esm.min.js"
import { abi, contractAddress } from "./constants.js"

const connectButton = document.getElementById("connectButton")
const withdrawButton = document.getElementById("withdrawButton")
const fundButton = document.getElementById("fundButton")
const balanceButton = document.getElementById("balanceButton")
connectButton.onclick = connect
withdrawButton.onclick = withdraw
fundButton.onclick = fund
balanceButton.onclick = getBalance

async function connect() {
  if (typeof window.ethereum !== "undefined") {
    try {
      await ethereum.request({ method: "eth_requestAccounts" })
    } catch (error) {
      console.log(error)
    }
    connectButton.innerHTML = "Connected"
    const accounts = await ethereum.request({ method: "eth_accounts" })
    console.log(accounts)
  } else {
    connectButton.innerHTML = "Please install MetaMask"
  }
}
```

```
async function withdraw() {
  console.log(`Withdrawing...`)
  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)
    }
  } else {
    withdrawButton.innerHTML = "Please install MetaMask"
  }
}

async function fund() {
  const ethAmount = document.getElementById("ethAmount").value
  console.log(`Funding with ${ethAmount}...`)
  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.fund({
        value: ethers.utils.parseEther(ethAmount),
      })
      await listenForTransactionMine(transactionResponse, provider)
    } catch (error) {
      console.log(error)
    }
  } else {
    fundButton.innerHTML = "Please install MetaMask"
  }
}
```

```
async function getBalance() {
  if (typeof window.ethereum !== "undefined") {
    const provider = new ethers.providers.Web3Provider(window.ethereum)
    try {
      const balance = await provider.getBalance(contractAddress)
      console.log(ethers.utils.formatEther(balance))
    } catch (error) {
      console.log(error)
    }
  } else {
    balanceButton.innerHTML = "Please install MetaMask"
  }
}

function listenForTransactionMine(transactionResponse, provider) {
  console.log(`Mining ${transactionResponse.hash}`)
  return new Promise((resolve, reject) => {
    provider.once(transactionResponse.hash, (transactionReceipt) => {
      console.log(
        `Completed with ${transactionReceipt.confirmations} confirmations. `
      )
      resolve()
    })
  })
}
```