# Exercises: Ethereum HD Wallet in JS

In this exercise, you shall implement a hierarchical wallet (**HD wallet**) for Ethereumin JavaScript, using the [Ethers.js library](https://github.com/ethers-io/ethers.js). The wallet will hold a **BIP39 mnemonic phrase** and will allow **deriving private keys** and addresses from it.

## Introduction

This exercise will focus on the concept of hierarchical wallets (HD Wallets), based on **BIP39** and **BIP44** specs. The [**HDNode** class](https://github.com/ethers-io/ethers.js/blob/master/wallet/hdnode.js) in the **Ethers.js** library holds a **seed key** + the ability to **derive private keys** by certain derivation path. Let’s play with it.

**Create** a new JavaScript project. To get **ethers.js** either install it through the NPM package manager:

npm install --save ethers

## Play with BIP39 and BIP44 Online

First, play a bit with the **BIP39 online implementation** here: <https://iancoleman.io/bip39>. Generate / load mnemonics, derive Ethereum and Bitcoin keys and addresses.

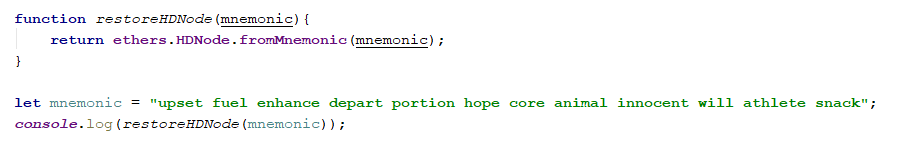
## Restore HD Wallet by Existing Mnemonic

Restore HD node by given existing mnemonic words.

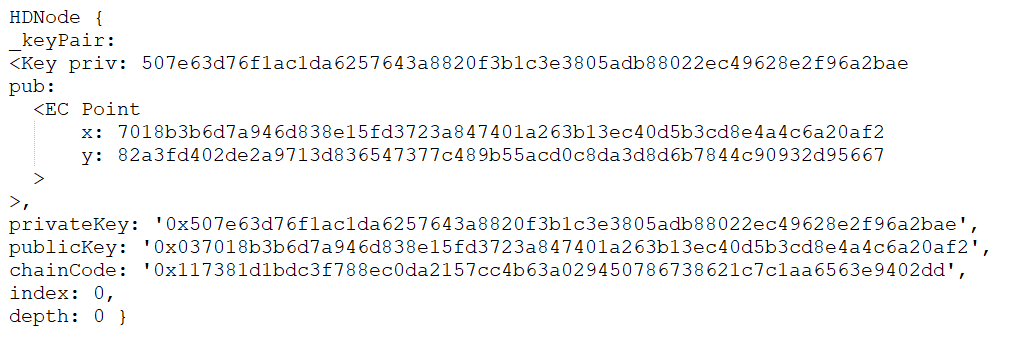
**upset fuel enhance depart portion hope core animal innocent will athlete snack**

A **Hierarchical Deterministic Wallet** represents a large tree of private keys, which can be reproduced from an initial seed. Each node in the tree is represented by an **HDNode** which can be descended into.

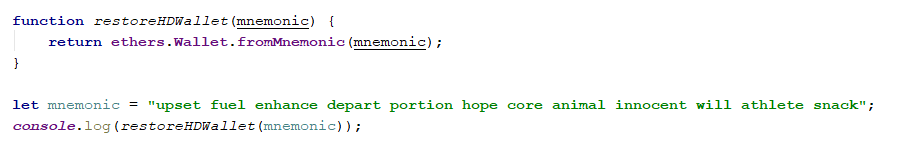
To create an HD node from a mnemonic, use **ethers.HDNode.fromMnemonic**:



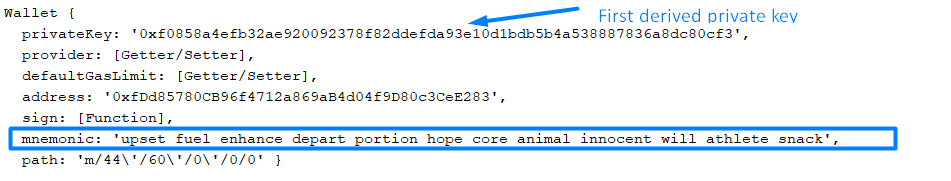
The HD node stores a **key pair**:



The **depth** represents the depth within the hierarchy of this node.



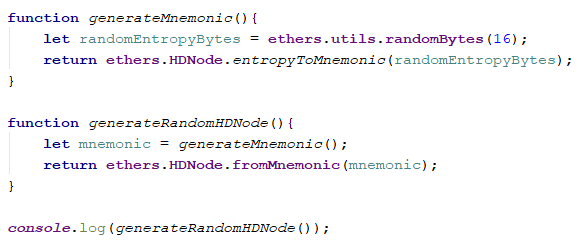
When an **ethers.Wallet** instance is created from a mnemonic, it actually uses **HDNode.fromMnemonic**, **derives once**, from the new HD node takes the private key, and builds the wallet.

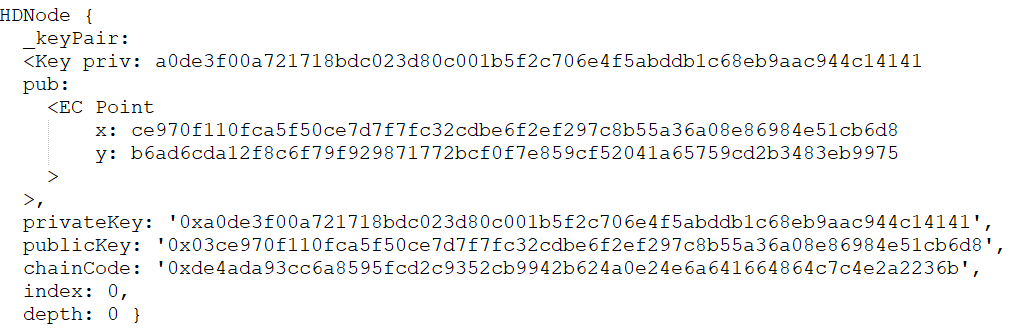


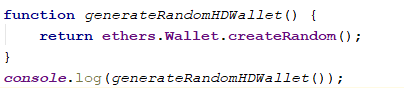
## New Random HD Wallet

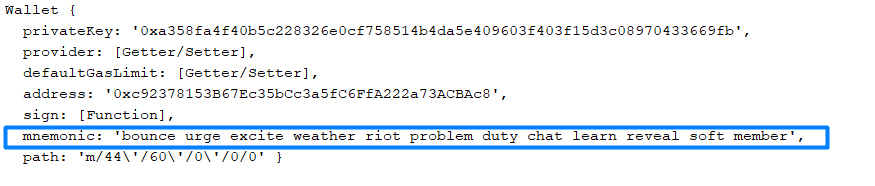
Generate new random HD node (generate random mnemonics)

To create a random HD node, you can either do **ethers.Wallet.createRandom** and build the HD node from the mnemonic, or from 16 bytes of entropy build the mnemonic using **ethers.HDNode.entropyToMnemonic**.









## Save HD Wallet as JSON

Encrypt and save given HD node to **JSON** document by given **password**.

To save the HD Wallet in encrypted JSON format, you need the Wallet to have the mnemonic phrase in it. The mnemonic is encrypted in the "**x-ethers**" part of the json.

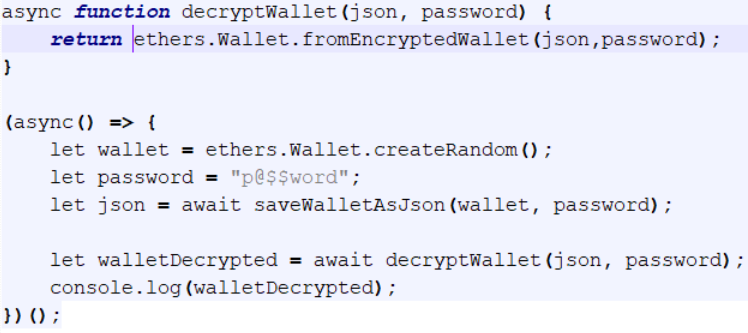


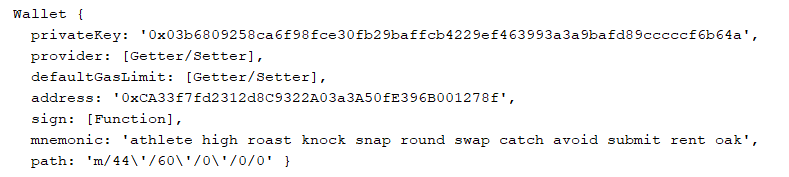


## Load HD Wallet from JSON

Decrypt and load a HD node from **JSON** document by given **password**.

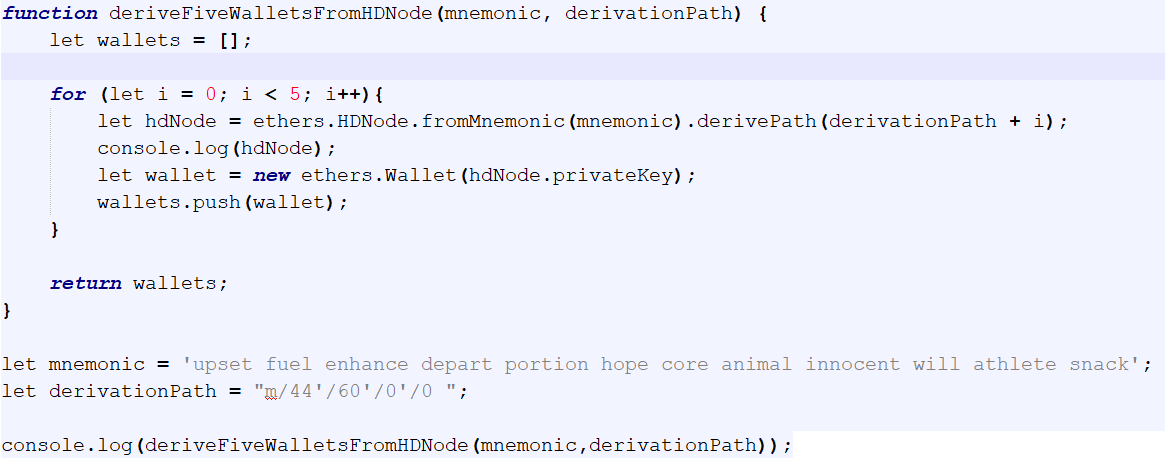
Use **ethers.Wallet.fromEncryptedMnemonic(json, password)**

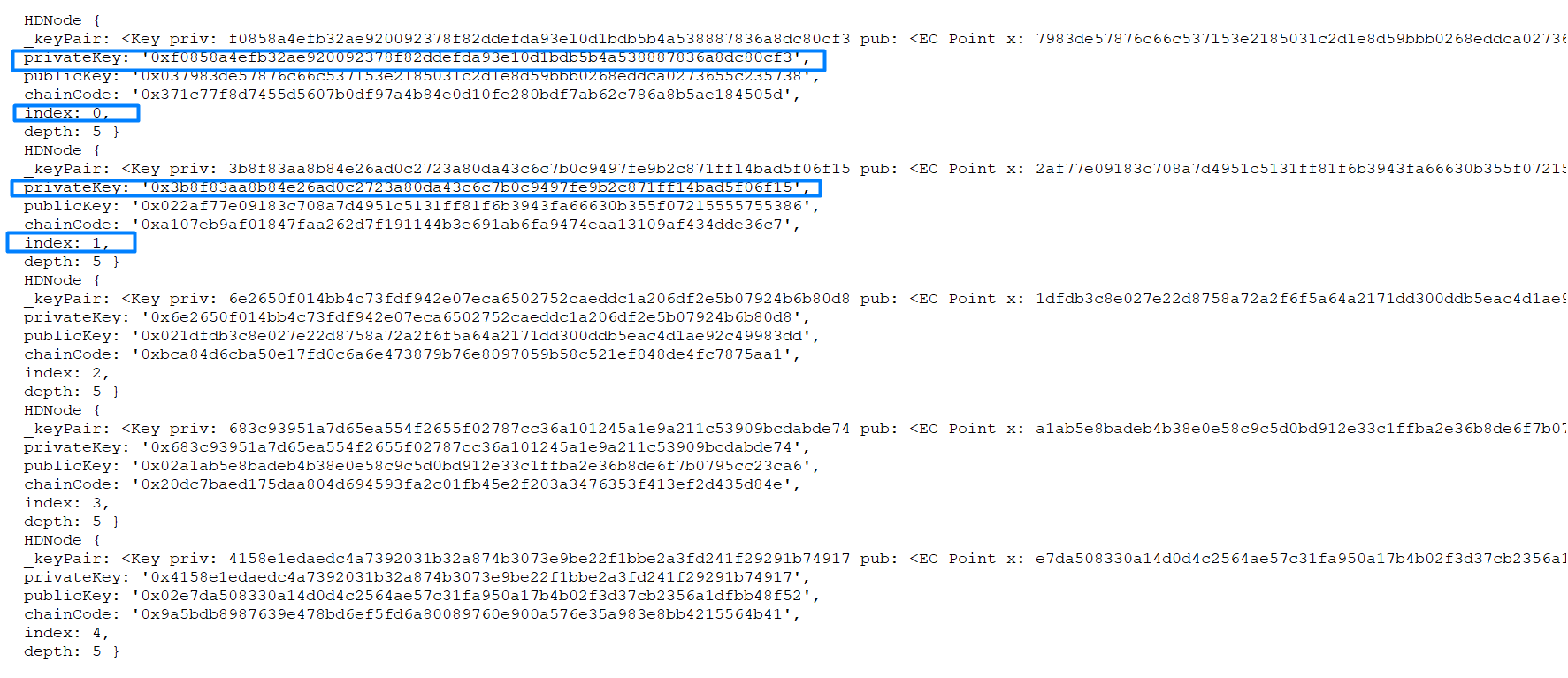




## Derive Keys from HD Wallet

**Derive keys** (and their associated addresses) from HD Wallet by given derivation path. Derivation path is **m/44'/60'/0'/0**

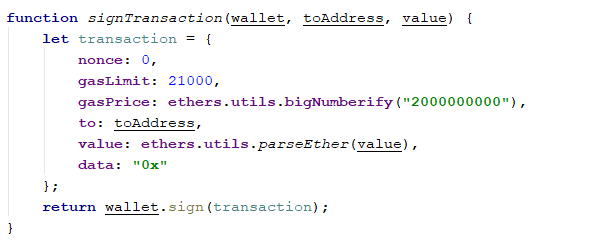






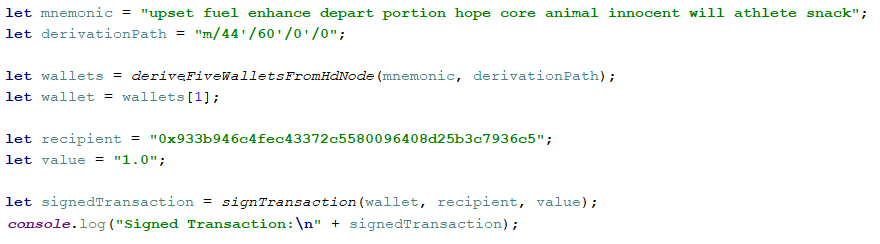
## Sign a Transaction

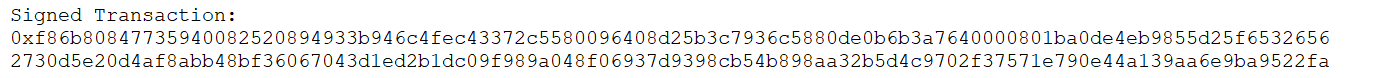
Take the second of the derived wallets and sign a transaction with given recipient address and value of ethers.



Ethereum Address :

0x933b946c4fec43372c5580096408d25b3c7936c5





# What to Submit?

Create a **ZIP file** (e.g. your-name-ethereum-hd-wallet-exercise.zip) holding your source code for all problems. Submit your ZIP file as **homework** at the course Web site.