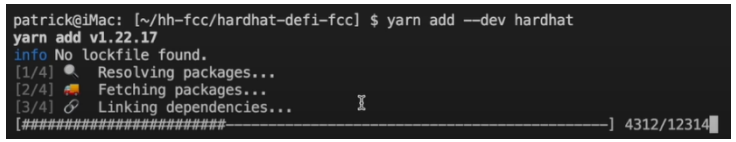
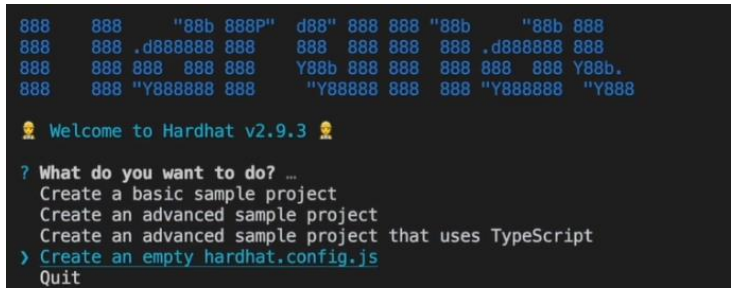
Lesson 13

HARDHAT DEFI & AAVE

Hikmah Nisya - 1103184094
Radzis Araaf Jaya Jamaludin - 1103184234
Raudhatul Rafiqah Assyahiddini - 1103180225

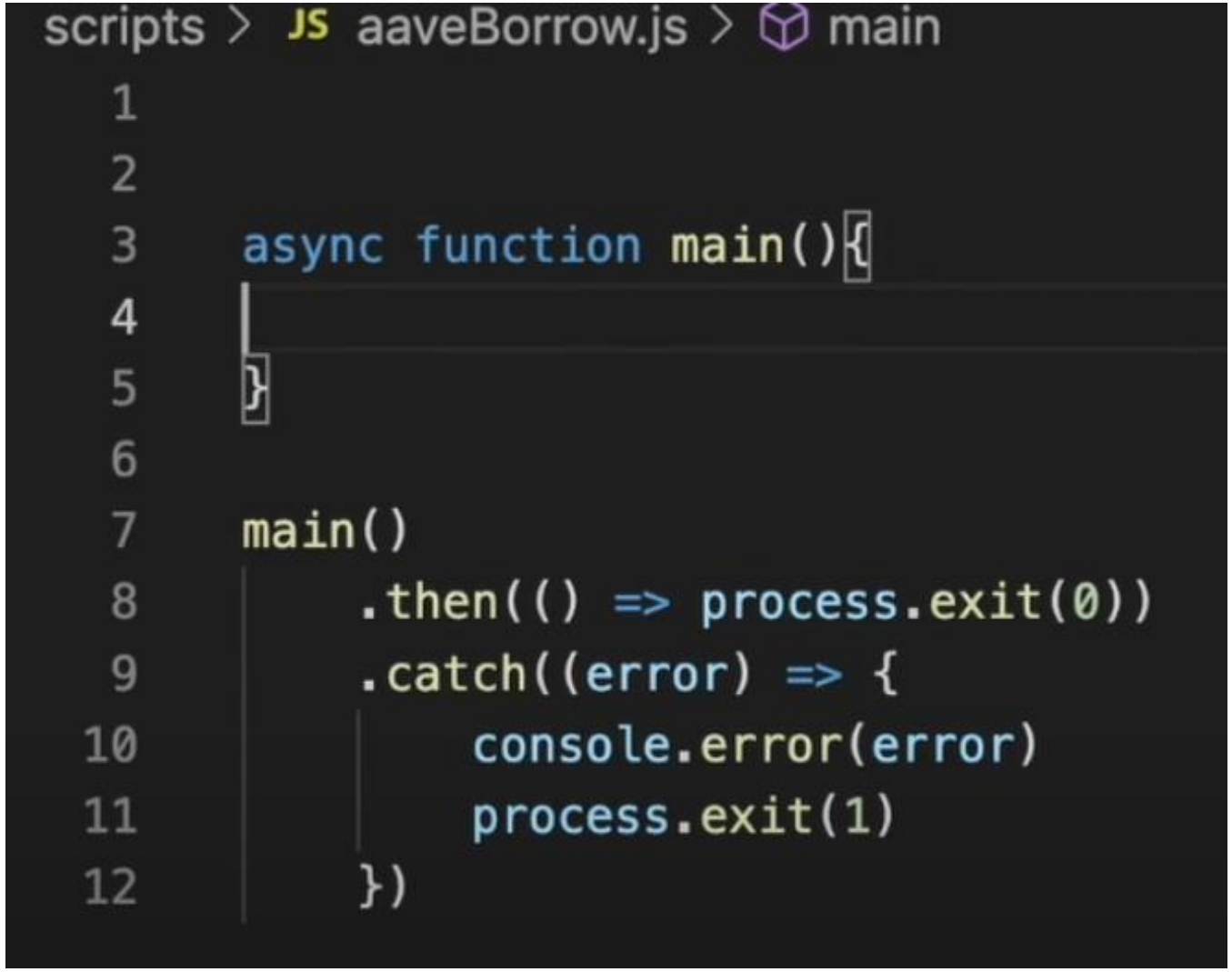


Buatlah Hardhat Terlebih dahulu dengan code “ yarn add –dev hardhat” lalu enter, dan tunggu sampai installasi nya selesai.



Setelah itu jalan kan hardhatnya denga code “ yarn hardhat”, tunggu sampai muncul pilihan, lalu pilih “ buat hardhat.config.js” setelah itu enter

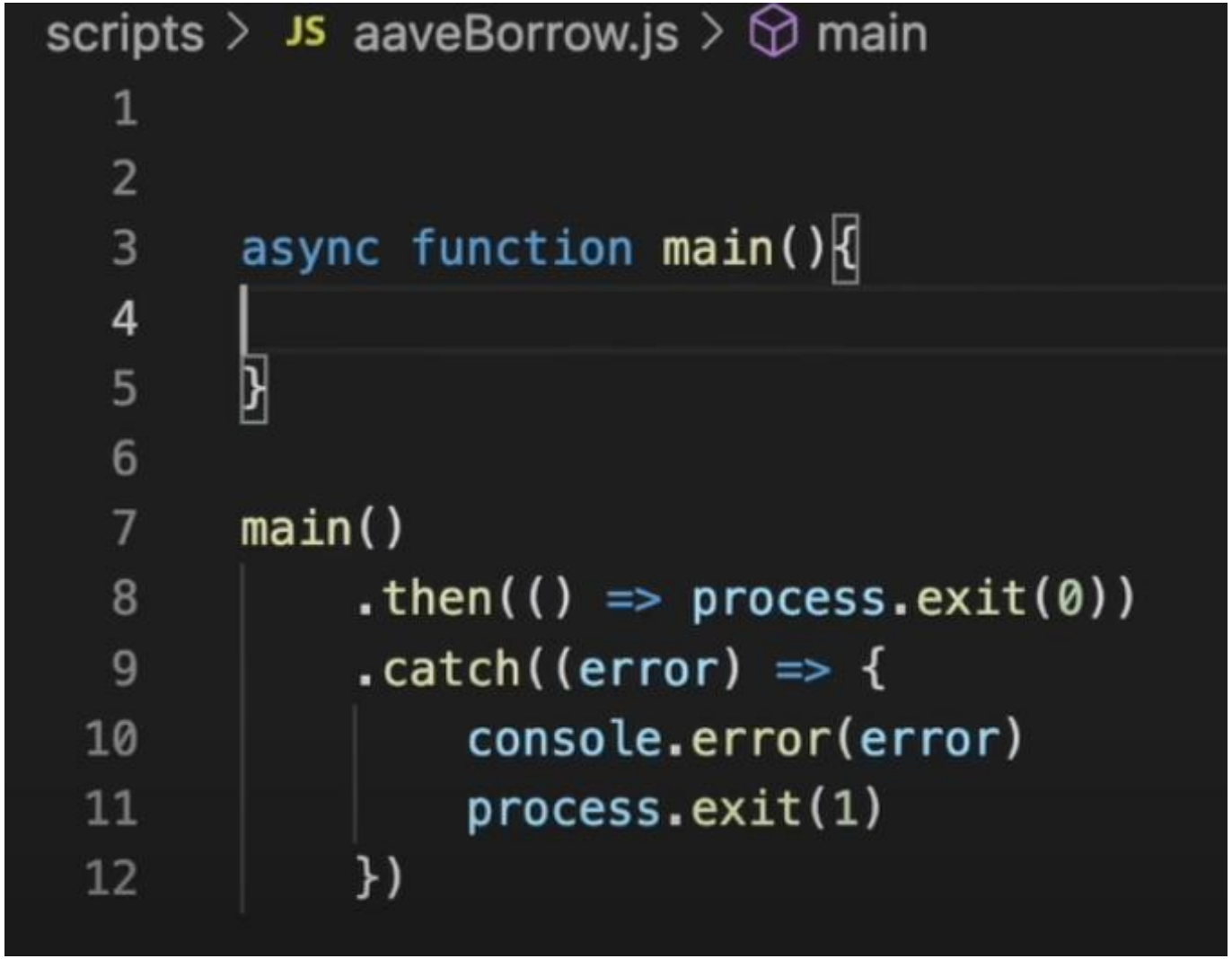
Lalu akan muncul tampilan seperti gambar di samping, dan selanjutnya buat project yang sama seperti Lesson 9.



selanjutnya membuat Scripts baru dengan nama “ aaveBorrows.js” dan membuat function main seperti gambar di ini.

WETH

Wrapped ETH

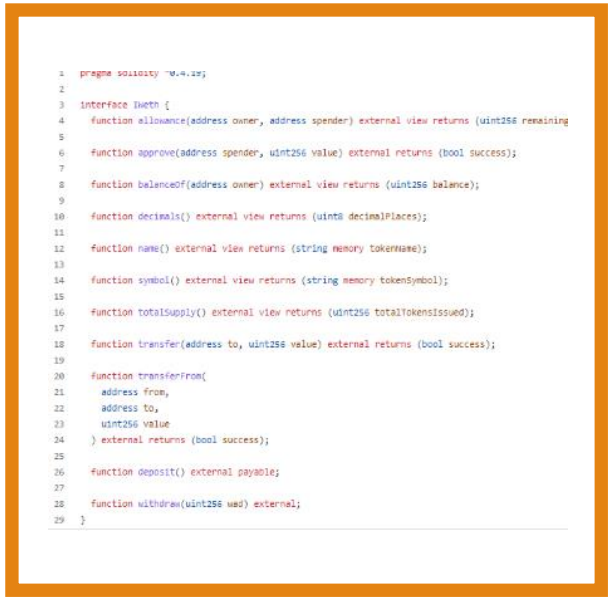


Langkah selanjutnya adalah membuat Scripts baru dengan nama “ aaveBorrows.js” dan membuat function main seperti gambar di ini.

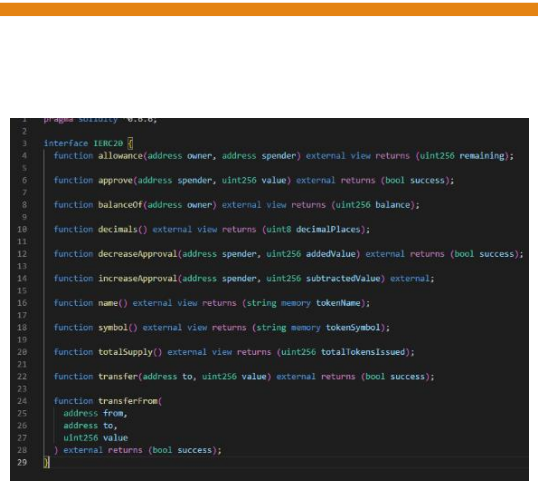
```
1   const { ethers, getNamedAccounts, network } = require("hardha
2   const { networkConfig } = require("../helper-hardhat-config")
3
4   const AMOUNT = ethers.utils.parseEther("0.1")
5
6   async function getWeth() {
7       const { deployer } = await getNamedAccounts()
8       const iWeth = await ethers.getContractAt(
9           "IWeth",
10          networkConfig[network.config.chainId].wethToken,
11          deployer
12      )
13      const txResponse = await iWeth.deposit({
14          value: AMOUNT,
15      })
16      await txResponse.wait(1)
17      const wethBalance = await iWeth.balanceOf(deployer)
18      console.log(`Got ${wethBalance.toString()} WETH`)
19  }
20
21  module.exports = { getWeth, AMOUNT }
```

Lalu buat script baru dengan nama “getWeth.js” , buat async function getWeth untuk membuat token yang di masukan kedalam data atau web aave

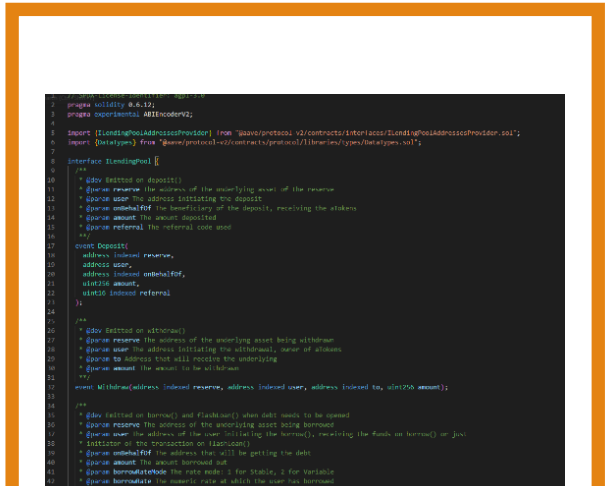
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.6.0;

interface AggregatorV3Interface {
  function decimals() external view returns (uint8);

  function description() external view returns (string memory);

  function version() external view returns (uint256);

  // getRoundData and latestRoundData should both raise "No data present"
  // if they do not have data to report, instead of returning unset values
  // which could be misinterpreted as actual reported values.
  function getRoundData(uint80 _roundId)
    external
    view
    returns (
      uint80 roundId,
      int256 answer,
      uint256 startedAt,
      uint256 updatedAt,
      uint80 answeredInRound
    );

  function latestRoundData()
    external
    view
    returns (
      uint80 roundId,
      int256 answer,
      uint256 startedAt,
      uint256 updatedAt,
      uint80 answeredInRound
    );
}
Footer
```



```
pragma solidity ^0.4.19;

interface IWeth {
  function allowance(address owner, address spender) external view returns (uint256 remaining

  function approve(address spender, uint256 value) external returns (bool success);

  function balanceOf(address owner) external view returns (uint256 balance);

  function decimals() external view returns (uint8 decimalPlaces);

  function name() external view returns (string memory tokenName);

  function symbol() external view returns (string memory tokenSymbol);

  function totalSupply() external view returns (uint256 totalTokensIssued);

  function transfer(address to, uint256 value) external returns (bool success);

  function transferFrom(
    address from,
    address to,
    uint256 value
  ) external returns (bool success);

  function deposit() external payable;

  function withdraw(uint256 wad) external;
}
```



```
pragma solidity ^0.8.0;

interface IERC20 {
  function allowance(address owner, address spender) external view returns (uint256 remaining);

  function approve(address spender, uint256 value) external returns (bool success);

  function balanceOf(address owner) external view returns (uint256 balance);

  function decimals() external view returns (uint8 decimalPlaces);

  function decreaseApproval(address spender, uint256 addedValue) external returns (bool success);

  function increaseApproval(address spender, uint256 subtractedValue) external;

  function name() external view returns (string memory tokenName);

  function symbol() external view returns (string memory tokenSymbol);

  function totalSupply() external view returns (uint256 totalTokensIssued);

  function transfer(address to, uint256 value) external returns (bool success);

  function transferFrom(
    address from,
    address to,
    uint256 value
  ) external returns (bool success);
}
```



```
// SPDX-License-Identifier: AGPL-3.0
pragma solidity 0.6.12;
pragma experimental ABIEncoderV2;

import {ILendingPoolAddressesProvider} from "@aave/protocol-v2/contracts/interfaces/ILendingPoolAddressesProvider.sol";
import {DataTypes} from "@aave/protocol-v2/contracts/protocol/libraries/types/DataTypes.sol";

interface ILendingPool {
  /**
   * @dev Emitted on deposit()
   * @param reserve The address of the underlying asset of the reserve
   * @param user The address initiating the deposit
   * @param onBehalfOf The beneficiary of the deposit, receiving the aTokens
   * @param amount The amount deposited
   * @param referral The referral code used
   **/
  event Deposit(
    address indexed reserve,
    address user,
    address indexed onBehalfOf,
    uint256 amount,
    uint16 indexed referral
  );

  /**
   * @dev Emitted on withdraw()
   * @param reserve The address of the underlying asset being withdrawn
   * @param user The address initiating the withdrawal, owner of aTokens
   * @param to Address that will receive the underlying
   * @param amount The amount to be withdrawn
   **/
  event Withdraw(address indexed reserve, address indexed user, address indexed to, uint256 amount);

  /**
   * @dev Emitted on borrow() and flashLoan() when debt needs to be opened
   * @param reserve The address of the underlying asset being borrowed
   * @param user The address of the user initiating the borrow(), receiving the funds on borrow() or flas
   * initiator of the transaction on flashLoan()
   * @param onBehalfOf The address that will be getting the debt
   * @param amount The amount borrowed out
   * @param borrowRateMode The rate mode: 1 for Stable, 2 for Variable
   * @param borrowRate The numeric rate at which the user has borrowed
   * @param referral The referral code used
   **/
```

Selanjutnya membuat “AggregatorV3Interface.sol”, ERC20, Ilendingpool,IWeth untuk interfacenya dalam contrakctions.

Dengan begitu DeFi bias di gunakan ke dalam Aave

```solidity
pragma solidity ^0.6.6;

interface IERC20 {
  function allowance(address owner, address spender) external view returns (uint256 rema

  function approve(address spender, uint256 value) external returns (bool success);

  function balanceOf(address owner) external view returns (uint256 balance);

  function decimals() external view returns (uint8 decimalPlaces);

  function decreaseApproval(address spender, uint256 addedValue) external returns (bool

  function increaseApproval(address spender, uint256 subtractedValue) external;

  function name() external view returns (string memory tokenName);

  function symbol() external view returns (string memory tokenSymbol);

  function totalSupply() external view returns (uint256 totalTokensIssued);

  function transfer(address to, uint256 value) external returns (bool success);

  function transferFrom(
    address from,
    address to,
    uint256 value
  ) external returns (bool success);
}
```

Selanjutnya kita akan membuat “IERC20.sol”