**Importing libraries from NPM and GitHub**

**Introduction:**

As we delve into smart contract development, interacting with external smart contracts will become increasingly frequent. This can involve the use of multiple interfaces, which can clog up the source code and make it difficult to read.

**Interfaces:**

Let’s take a look at the `SmartContract` interface as an example:

Interface SmartContract {

Function someFunction() external view returns(uint, uint){};

}

To include this contract, we simply use the `import “./SimpleStorage.sol”` statement at the top of our file. Instead, if the project is not stored locally, we can use imports from Github.

**Direct Imports from GitHub:**

Smart Contracts hosted on GitHub can be imported directly into your project. For instance, consider the `AggregatorV3Interface` contract from Chainlink, which is located in Chainlink’s GitHub repository and not in our project’s directory.

Instead of manually copying all its code into your project and then importing it like this:

Import { AggregatorV3Interface } from “./AggregatorV3Interface.sol”;

We can import it more efficiently, as specified in the Chainlink documentation(<https://docs.chain.link/docs/using-chainlink-reference-contracts>):

Import { AggregatorV3Interface } from “@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol”;

This import statement includes the path to the AggregatorV3Interface.sol file in the GitHub repository, allowing you to directly import the contract from GitHub or NPM (Node Package Manager).

**Importing from NPM:**

The `@chainlink/contracts` package, available on NPM, follows Semantic Versioning (SemVer), which allows you to download and use specific versions in your contracts (e.g., `npm install @chainlink/contracts@1.2.3`) while being directly synchronized with Chainlink’s GitHub repository. The rest of the import path specifies the exact file that Remix should use.

Remix interprets `@chainlink/contracts` as a reference to the [NPM package] (<https://www.npmjs.com/package/@chainlink/contracts>), and downloads all the necessary code from it.

Pragma solidity ^0.8.18;

Import {AggregatorV3Interface} from “@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol”;

Contract FundMe {}

**Conclusion:**

Efficiently managing external smart contracts is crucial in smart contract development to maintain clean and readable source code. Utilizing direct imports from GitHub and NPM, as shown with Chainlink’s AggregatorV3Interface, simplifies this process.

**Updates:**

Correct import for Chainlink VRF

Last updated on November 12, 2024

Note that as mentioned on the video for the 2024 update, the import for Chainlink VRF changed from:

Import {AggregatorV3Interface} from “@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol”;

To

Import {AggregatorV3Interface} from “@chainlink/contracts/src/v0.8/shared/interfaces/AggregatorV3Interface.sol”;

So be sure to add the shared on your code.

Take into consideration that this might change with time, so be sure to check Chainlink VRF documentation to be sure you are using the correct import.