**Solidity Imports**

**Introduction:**

In the previous lesson, we integrated the `SimpleStorage` code directly into the `StorageFactory` contract. This allowed `StorageFactory` to have full access to the `SimpleStorage` contract’s functionality. In this lesson, we will explore a more efficient way to arrange and organize the code by using the `import`statement.

**Importing code:**

The `import` keyword enables a contract to utilize code from other files without needing to include the entire codebase directly within the contract. Here are two of the main advantages that the `import` keyword provides:

**1. No cluttering:** it prevents your current file from being cluttered with numerous lines of code, keeping it clean and organized.

**2. Simplified maintenance:** by keeping the code in separate files, it becomes easier to maintain and update individual components without affecting the entire codebase. For example, if we change some lines inside `SimpleStorage`, we would have also to constantly copy-paste the modified content into `StorageFactory`

You can now remove the previously added `SimpleStorage` code and replace it with the `import` shorthand:

import “./SimpleStorage.sol”;

**Warning:** All the solidity contracts should be compiled together using the same compiler version. It’s important to ensure consistency between compiler versions across files since each one will have its own `pragma` statement.

**Named Imports:**

Let’s assume for a moment that `SimpleStorage` would contain multiple contracts, e.g. `SimpleStorage`, `SimpleStorage1`, `SimpleStorage2`, which are quite extensive in size. If we import the whole file as we did before, the statement will replace the `import` directive with all the code contained in `SimpleStorage.sol`. This will result in an unnecessary expensive deployment of the `StorageFactory` contract.

This can be prevented with named imports, which allow you to selectively import only the specific contracts you intend to use:

import { SimpleStorage } from “./SimpleStorage.sol”

You can also use named imports to import multiple contracts:

import { SimpleStorage, SimpleStorage1 } from “./SimpleStorage.sol”;

**Important:** Try to always default to named imports instead of importing the entire file.

**Conclusion:**

The import keyword allows a contract to use code from other files without including the entire codebase. However, it can introduce compilation issues if different compiler versions are used in these files.