**Storage Factory Introduction**

**Introduction:**

You can find the code for this section in the Remix Storage Factory Github repository (<https://github.com/cyfrin/remix-storage-factory-f23>). In these nine lessons, we’ll work with three new contracts:

**1.** **SimpleStorage.sol** - the contract we build in the previous section, with some modifications

**2. AddFiveStorage.sol** - a child contract of `SimpleStorage` that leverages inheritance

**3. StorageFactory.sol** - a contract that will deploy a `SimpleStorage` contract and interact with it.

**Section Overview:**

Contract SimpleStorage {

SimpleStorage[] public listOfSimpleStorageContracts;

Function createSimpleStorageContract() public {};

Function sfStore(uint256 \_simpleStorageIndex, uint256 \_simpleStorageNumber) public {};

Function sfGet(uint256 \_simpleStorageIndex) public view returns (uint256) {}

}

After deploying `StorageFactory` and executing its function `createSimpleStorageContract`, we can observe a new transaction appear in the Remix terminal. It’s a deployment transaction of the `SimpleStorage` contract, executed by the `StorageFactory` contract.

It's possible to Interact with this newly deployed `SimpleStorage` via the `store` function. We’ll do this by using the `sfStore` function from the `StorageFactory` contract. This function accepts two parameters: the index of a deployed `SimpleStorage` contract, which will be ‘0’ since we just deployed one contract, and the value of a `favoriteNumber`.

The `sfGet` function, when given the input ‘0’, will indeed return the number provided by the previous function. The \*\*address\*\* of the `SimpleStorage` contract can then be retrieved by clicking on the get function `listOfSimpleStorageContracts`

**Conclusion:**

The `StorageFactory` contract manages numerous instances of an external contract `SimpleStorage`. It provides functionality to deploy new contract instances dynamically and allows for the storage and retrieval of values from each instance. These instances are maintained and organized within an array, enabling efficient tracking and interaction.