**Memory Storage And Calldata**

**Introduction:**

In this section, we will explore how Solidity manages data storage, focusing on the differences between storage, memory, and calldata, and why these concepts are crucial for writing optimized and secure smart contracts.

**Data Locations:**

Solidity can store data in six different locations. In this lesson, we will focus on the first three:

1. Calldata
2. Memory
3. Storage
4. Stack
5. Code
6. Logs

**Calldata and Memory:**

In Solidity, calldata and memory are temporary storage locations for variables during function execution. calldata is read-only, used for function inputs that can’t be modified. In contrast, memory allows for read-write access, letting variables be changed within the function. To modify calldata variables, they must first be loaded into memory.

**warning:** Most variable types default to memory automatically. However, for strings, you must specify either memory or calldata due to the way arrays are handled in memory.

string memory variableName = “someValue”;

**Calldata:**

Calldata variables are read-only and cheaper than memory. They are mostly used for input parameters.

In the following example, if we try to replace the keyword memory with calldata, we receive an error because calldata variables can’t be manipulated.

function addPerson(string calldata \_name, uitn256 \_favoriteNumber) public {

\_name = “cat”;

listOfPeople.push(Person(\_favoriteNumber, \_name));

}

**Storage:**

Variables stored in storage are persistent on the blockchain, retaining their values between function calls and transactions.

In our contract, the variable `myFavoriteNumber` is a storage variable. Variables which are declared outside any function are implicitly converted to storage variables.

contract MyContract {

uint256 favoriteNumber; // this is a storage variable

};

**Strings and primitive types:**

If you try to specify the memory keyword for an uint256 variable, you’ll encounter this error:

“Data location can only be specified for array, struct, or mapping type”

In Solidity, a `string` is recognized as an ‘array of bytes’. On the other hand, primitive types, like `uint256` have built-in mechanisms that dictate how and where they are stored, accessed and manipulated.

**warning:** You can’t use the `storage` keyword for variables inside a function. Only `memory` and `calldata` are allowed here, as the variable only exists temporarily.

function addPerson(string memory \_name, uitn256 \_favoriteNumber) public { // cannot use storage as input parameters

uint256 test = 0; // variable here can be stored in memory or stack

listOfPeople.push(Person(\_favoriteNumber, \_name));

}

**Conclusion:**

Well done! You’ve learned the differences between the keywords storage, memory, and calldata in Solidity, enhancing your skills to develop robust Ethereum-based applications.