

When should I use calldata and when should I use memory?

Asked 2 years, 2 months ago Active 5 months ago Viewed 18k times

59

I have seen people use both `memory` and `calldata` keywords when writing Solidity. Specifically, they are used when declaring function parameters.

When should I use `memory` and when should I use `calldata` ?

★

17

solidity

contract-development

memory

calldata


keyword

🕒

Share Improve this question Follow

edited Sep 1 '19 at 17:42

asked Aug 30 '19 at 23:51

 **Shane Fontaine**
11.6k 12 33 67

1 Answer

Active

Oldest

Votes

68

`memory` and `calldata` (as well as `storage`) are keywords that define the data area where a variable is stored. To answer your question directly, `memory` should be used when declaring variables (both function parameters as well as inside the logic of a function) that you want stored in memory (temporary), and `calldata` *must* be used when declaring an **external** function's **dynamic** parameters.

✓

The easiest way to think about the difference is that `calldata` is a non-modifiable, non-persistent area where function arguments are stored, and behaves mostly like memory.

🕒

Breaking this down, let's first look at `memory` . `memory` 's lifetime is limited to a function call and is meant to be used to temporarily store variables and their values. Values stored in `memory` do not persist on the network after the transaction has been completed. Some notable implementation details about memory are as follows:

- It can be used for both function declaration parameters as well as within the function logic
- It is mutable (it can be overwritten and changed)
- It is non-persistent (the value does not persist after the transaction has completed)

[This](#) answer is a great resource to understand memory.

`calldata` is very similar to memory in that it is a data location where items are stored. It is a special data location that contains the function arguments, only available for external function call parameters. From the Solidity [docs](#):

Calldata is a non-modifiable, non-persistent area where function arguments are stored, and behaves mostly like memory.

This is the cheapest location to use, ~~but it has a limited size~~. In particular, that means that functions may be limited in their number of arguments.¹ Notable implementation details about `calldata` are as follows:

- It can **only** be used for function declaration parameters (and not function logic)
- It is immutable (it can't be overwritten and changed)
- It **must** be used for dynamic parameters of an external function
- It is non-persistent (the value does not persist after the transaction has completed)

The following is a **valid** example of code using `memory` and `calldata` :

```
pragma solidity 0.5.11;

contract Test {

    string stringTest;

    function memoryTest(string memory _exampleString) public returns (string memory) {
        _exampleString = "example"; // You can modify memory
        string memory newString = _exampleString; // You can use memory within a
function's logic
        return newString; // You can return memory
    }

    function calldataTest(string calldata _exampleString) external returns (string
memory) {
        // cannot modify _exampleString
        // but can return it
        return _exampleString;
    }
}
```

Edit Based on the comment by Tjaden Hess

One good way to think about the difference and how they should be used is that `calldata` is allocated by the caller, while `memory` is allocated by the callee.

Share Improve this answer Follow

edited Jun 14 at 5:44



DiveInto
328 2 13

answered Aug 30 '19 at 23:51



Shane Fontaine
11.6k 12 33 67

-
- 9

In what sense does calldata have limited size? Calldata is exactly the size of the data passed to the underlying EVM call, which can be as large as you want (bounded by the gas budget). I'd say that the difference between calldata and memory is that calldata is allocated by the caller, while memory is allocated by the callee – [Tjaden Hess](#) Sep 1 '19 at 21:44
-
- 1

"In what sense does calldata have limited size?" That is an oversight on my part. The limit is, like you said, either gas budget or stack depth limit. I will edit the answer. "calldata is allocated by the caller, while memory is allocated by the callee" This is a great way to put it – [Shane Fontaine](#) Sep 2 '19 at 17:26
-
- 5

Are you sure your "function memoryTest(" example is correct? – [Long Field](#) Jun 8 '20 at 1:28

It has been updated with the correct variables. – [Shane Fontaine](#) Jun 14 at 16:30
-