Advanced Functions

Function Visibility and State Mutability

Copy page

A quick reference for all your function declaring needs.

You've seen much of this before, but this document outlines and highlights the options for *function visibility* and *state mutability* all in one document.

Objectives

By the end of this lesson you should be able to:

- Categorize functions as public, private, internal, or external based on their usage
- Describe how pure and view functions are different than functions that modify storage

Ask a question... Ctrl+I

Function Visibility

There are four types of **visibility** for functions in Solidity: `external` , `public` , `internal` , and `private` . These labels represent a further division of the *public* and *private* labels you might use in another language.

External

Functions with `external` visibility are **only** callable from other contracts and cannot be called within their own contract. You may see older references stating you should use `external` over `public` because it forces the function to use `calldata` . This is no longer correct, because both function visibilities can now use `calldata` and `memory` for parameters. However, using `calldata` for either will cost less gas.

```
contract Foo {
    constructor() {
        // Bad code example, will not work
        uint bar = foo(3);
        // ... other code
    }

    function foo(uint _number) external pure returns (uint) {
         return _number*2;
    }
}
```

Public

Public functions work the same as external, except they may also be called within the contract that contains them.

```
contract Foo {
    constructor() {
        // Public functions may be called within the contract
        uint bar = foo(3);
        // ... other code
    }

    function foo(uint _number) public pure returns (uint) {
        return _number*2;
    }
}
```

Private and Internal

Functions visible as `private` and `internal` operate nearly identically. Beyond writing hygienic code, these have a very important effect. Because they are not a part of the contract's ABI, you can use `mapping` s and `storage` variable references as parameters.

The difference is that `private` functions can't be called from derived contracts. You'll learn more about that when we cover inheritance.

Some developers prepend an underscore to `private` and `internal` functions.

```
function _foo(uint _number) private returns (uint) {
    return _number*2;
}
```

⚠ All data on a blockchain is public. Don’t mistake hiding visibility while coding for hiding information from the world!

Function State Mutability

State mutability labels are relatively unique to Solidity. They determine how a function can interact with state, which has a substantial impact on gas costs.

Pure

`pure` functions promise to neither read nor write state. They’re usually used for helper functions that support other functionality.

```
function abs(int x) public pure returns (int) {
    return x >= 0 ? x : -x;
}
```

`pure` functions can be called from outside the blockchain without using gas, if they are also `public` or `external` .

View

`view` functions access state, but don't modify it. You've used these for tasks such as returning all the values in an array.

```
function getArr() public view returns (uint[] memory) {
    return arr;
}
```

`view` functions can be called from outside the blockchain without using gas, if they are also `public` or `external` .

Unlabeled Functions

Functions that are not labeled `view` or `pure` can modify state and the compiler will generate a warning if they do not.

```
function addToArr(uint _number) public {
    arr.push(_number);
}
```

They can have any visibility and will always cost gas when called.

Conclusion

The visibility and mutability keywords in Solidity help you organize your code and alert other developers to the properties of each of your functions. Use them to keep your code organized and readable.

Was this page helpful? Yes No Suggest edits Raise issue

‹ Function Visibility Function Modifiers ›

B e.org Blog Privacy Policy Terms of Service Cookie Policy