[`external` vs `public` best practices](https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices)

A simple example demonstrating this effect looks like this:

pragma solidity^0.4.12;

contract Test {

function test(uint[20] a) public returns (uint){

return a[10]\*2;

}

function test2(uint[20] a) external returns (uint){

return a[10]\*2;

}

}

Calling each function, we can see that the public function uses 496 gas, while the external function uses only 261.

The difference is because in public functions, Solidity immediately copies array arguments to memory, while external functions can read directly from calldata. Memory allocation is expensive, whereas reading from calldata is cheap.

The reason that public functions need to write all of the arguments to memory is that public functions may be called internally, which is actually an entirely different process than external calls. Internal calls are executed via jumps in the code, and array arguments are passed internally by pointers to memory. Thus, when the compiler generates the code for an internal function, that function expects its arguments to be located in memory.

For external functions, the compiler doesn't need to allow internal calls, and so it allows arguments to be read directly from calldata, saving the copying step.

As for best practices, you should use external if you expect that the function will only ever be called externally, and use public if you need to call the function internally. It almost never makes sense to use the this.f() pattern, as this requires a real CALL to be executed, which is expensive. Also, passing arrays via this method would be far more expensive than passing them internally.

You will essentially see performance benefits with external any time you are only calling a function externally, and passing in a lot of calldata (eg, large arrays).

Examples to differentiate:

**public -** all can access

**external -** Cannot be accessed internally, only externally

**internal -** only this contract and contracts deriving from it can access

**private -** can be accessed only from this contract