Advanced Functions
Function Visibility
Visibility Overview
Function Modifiers
Modifiers Guide
Structs
Structs
Step by Step Guide
Exercise
Inheritance
Inheritance Overview
Step by Step Guide
Multiple Inheritance
Multiple Inheritance Guide
Abstract Contracts
Abstract Contracts Guide

Inheritance

Inheritance

Copy page

Learn how to use inheritance to bring functionality from one contract into another.

Solidity is an object-oriented language. Contracts can inherit from one another, allowing efficient reuse of code.

Objectives

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

- Write a smart contract that inherits from another contract
- Describe the impact inheritance has on the byte code size limit

Ask a question... Ctrl+I

Function Visibility
Visibility Overview
Function Modifiers
Modifiers Guide
Structs
Step by Step Guide
Exercise
Inheritance Overview
Step by Step Guide
Multiple Inheritance
Multiple Inheritance Guide
Abstract Contracts
Abstract Contracts Guide

Inheritance

Create a new contract file in Remix called `Inheritance.sol` and add two simple contracts, each with a function identifying which contract called it:

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.17;

contract ContractB {
    function whoAmI() external pure returns (string memory) {
        return "contract B";
    }
}

contract ContractA {
    function whoAmI() external pure returns (string memory) {
        return "contract A";
    }
}
```

`ContractA` says that it is “contract A” and `ContractB` says that it is “contract B”.

Inheriting from Another Contract

Inheritance between contracts is indicated by the `is` keyword in the contract declaration. Update `ContractA` so that it `is` `ContractB`, and delete the `whoAmI`

Function Visibility
Visibility Overview
Function Modifiers
Modifiers Guide
Structs
Step by Step Guide
Exercise
Inheritance Overview
Step by Step Guide
Multiple Inheritance
Multiple Inheritance Guide
Abstract Contracts
Abstract Contracts Guide

function from `ContractA`.

Reveal code

Deploy and test again. Even though `ContractA` doesn't have any functions in it, the deployment still shows the button to call `whoAmI`. Call it. `ContractA` now reports that it is "contract B", due to the inheritance of the function from `Contract B`.

Internal Functions and Inheritance

Contracts can call the `internal` functions from contracts they inherit from. Add an `internal` function to `ContractB` called `whoAmIInternal` that returns "contract B".

Add an external function called `whoAmIExternal` that returns the results of a call to `whoAmIInternal`.

Reveal code

Deploy and test. Note that in the deployment for `ContractB`, the `whoAmIInternal` function is **not** available, as it is `internal`. However, calling `whoAmIExternal` can call the `internal` function and return the expected result of "contract B".

Internal vs. Private

Function Visibility
Visibility Overview
Function Modifiers
Modifiers Guide
Structs
Step by Step Guide
Exercise
Inheritance Overview
Step by Step Guide
Multiple Inheritance
Multiple Inheritance Guide
Abstract Contracts
Abstract Contracts Guide

You cannot call a `private` function from a contract that inherits from the contract containing that function.

```
// Bad code example, do not use
contract ContractB {
    function whoAmIPrivate() private pure returns (string memory) {
        return "contract B";
    }
}

contract ContractA is ContractB {
    function whoAmExternal() external pure returns (string memory) {
        return whoAmIPrivate();
    }
}
```

The compiler will raise an error:

```
from solidity:
DeclarationError: Undeclared identifier.
  --> contracts/Inheritance.sol:17:16:
   |
17 |         return whoAmIPrivate();
   |                ^^^^^^^^^^^^^
```

Function Visibility
Visibility Overview
Function Modifiers
Modifiers Guide
Structs
Step by Step Guide
Exercise
Inheritance Overview
Step by Step Guide
Multiple Inheritance
Multiple Inheritance Guide
Abstract Contracts
Abstract Contracts Guide

Inheritance and Contract Size

A contract that inherits from another contract will have that contract's bytecode included within its own. You can view this by opening settings in Remix and turning *Artifact Generation* back on. The bytecode for each compiled contract will be present in the JSON file matching that contract's name within the `artifacts` folder.

Any empty contract:

```
contract EmptyContract {

}
```

Will compile into something similar to this:

```
6080604052600080fdfea2646970667358221220df894b82f904e22617d7e4015030̶6e2̶2e8
```

A slightly more complex contract:

```
contract notEmptyContract {
    function sayHello() public pure returns (string memory) {
        return "To whom it may concern, I write you after a long period of
    }
}
```

Function Visibility
Visibility Overview
Function Modifiers
Modifiers Guide
Structs
Step by Step Guide
Exercise
Inheritance Overview
Step by Step Guide
Multiple Inheritance
Multiple Inheritance Guide
Abstract Contracts
Abstract Contracts Guide

Will have more complex bytecode. In this case, mostly to store the long string present in the return:

```
6080604052348015610010576000080fd5b5061020180610020600039600 0f3fe6080604052 3
```

However, if the empty contract inherits from the not empty contract:

```
contract EmptyContract is notEmptyContract {

}
```

The resulting bytecode will include that of the contract inherited from:

```
6080604052348015610010576000080fd5b5061020180610020600039600 0f3fe6080604052 3
```

Conclusion

In this lesson, you've learned how to use inheritance to include the functionality of one contract in another. You've also learned that inheriting contracts can call `internal` functions, but they cannot call `private` functions. You've also learned that inheriting from a contract adds the size of that contract's bytecode to the total deployed size.

Was this page helpful? Yes No Suggest edits Raise issue

Inheritance Overview Multiple Inheritance

Function Visibility
Visibility Overview
Function Modifiers
Modifiers Guide
Structs
Step by Step Guide
Exercise
Inheritance Overview
Step by Step Guide
Multiple Inheritance
Multiple Inheritance Guide
Abstract Contracts
Abstract Contracts Guide