**Structure of a Contract**[**ℑ**](https://docs.soliditylang.org/en/v0.8.19/structure-of-a-contract.html#structure-of-a-contract)

Contracts in Solidity are similar to classes in object-oriented languages. Each contract can contain declarations of [State Variables](https://docs.soliditylang.org/en/v0.8.19/structure-of-a-contract.html#structure-state-variables), [Functions](https://docs.soliditylang.org/en/v0.8.19/structure-of-a-contract.html#structure-functions), [Function Modifiers](https://docs.soliditylang.org/en/v0.8.19/structure-of-a-contract.html#structure-function-modifiers), [Events](https://docs.soliditylang.org/en/v0.8.19/structure-of-a-contract.html#structure-events), [Errors](https://docs.soliditylang.org/en/v0.8.19/structure-of-a-contract.html#structure-errors), [Struct Types](https://docs.soliditylang.org/en/v0.8.19/structure-of-a-contract.html#structure-struct-types) and [Enum Types](https://docs.soliditylang.org/en/v0.8.19/structure-of-a-contract.html#structure-enum-types). Furthermore, contracts can inherit from other contracts.

There are also special kinds of contracts called [libraries](https://docs.soliditylang.org/en/v0.8.19/contracts.html#libraries) and [interfaces](https://docs.soliditylang.org/en/v0.8.19/contracts.html#interfaces).

The section about [contracts](https://docs.soliditylang.org/en/v0.8.19/contracts.html#contracts) contains more details than this section, which serves to provide a quick overview.

**State Variables**[**ℑ**](https://docs.soliditylang.org/en/v0.8.19/structure-of-a-contract.html#state-variables)

State variables are variables whose values are permanently stored in contract storage.

*// SPDX-License-Identifier: GPL-3.0*

**pragma solidity** >=**0.4.0** <**0.9.0**;

**contract** **SimpleStorage** {

uint storedData; *// State variable*

*// ...*

}

See the [Types](https://docs.soliditylang.org/en/v0.8.19/types.html#types) section for valid state variable types and [Visibility and Getters](https://docs.soliditylang.org/en/v0.8.19/contracts.html#visibility-and-getters) for possible choices for visibility.

**Functions**[**ℑ**](https://docs.soliditylang.org/en/v0.8.19/structure-of-a-contract.html#functions)

Functions are the executable units of code. Functions are usually defined inside a contract, but they can also be defined outside of contracts.

*// SPDX-License-Identifier: GPL-3.0*

**pragma solidity** >=**0.7.1** <**0.9.0**;

**contract** **SimpleAuction** {

function bid() public payable { *// Function*

*// ...*

}

}

*// Helper function defined outside of a contract*

function helper(uint x) pure returns (uint) {

return x \* 2;

}

[Function Calls](https://docs.soliditylang.org/en/v0.8.19/control-structures.html#function-calls) can happen internally or externally and have different levels of [visibility](https://docs.soliditylang.org/en/v0.8.19/contracts.html#visibility-and-getters) towards other contracts. [Functions](https://docs.soliditylang.org/en/v0.8.19/contracts.html#functions) accept [parameters and return variables](https://docs.soliditylang.org/en/v0.8.19/contracts.html#function-parameters-return-variables) to pass parameters and values between them.

**Function Modifiers**[**ℑ**](https://docs.soliditylang.org/en/v0.8.19/structure-of-a-contract.html#function-modifiers)

Function modifiers can be used to amend the semantics of functions in a declarative way (see [Function Modifiers](https://docs.soliditylang.org/en/v0.8.19/contracts.html#modifiers) in the contracts section).

Overloading, that is, having the same modifier name with different parameters, is not possible.

Like functions, modifiers can be [overridden](https://docs.soliditylang.org/en/v0.8.19/contracts.html#modifier-overriding).

*// SPDX-License-Identifier: GPL-3.0*

**pragma solidity** >=**0.4.22** <**0.9.0**;

**contract** **Purchase** {

address **public** seller;

modifier onlySeller() { *// Modifier*

require(

**msg.sender** == seller,

"Only seller can call this."

);

\_;

}

function abort() public view onlySeller { *// Modifier usage*

*// ...*

}

}

**Events**[**ℑ**](https://docs.soliditylang.org/en/v0.8.19/structure-of-a-contract.html#events)

Events are convenience interfaces with the EVM logging facilities.

*// SPDX-License-Identifier: GPL-3.0*

**pragma solidity** >=**0.4.21** <**0.9.0**;

**contract** **SimpleAuction** {

event HighestBidIncreased(address bidder, uint amount); *// Event*

function bid() public payable {

*// ...*

emit HighestBidIncreased(**msg.sender**, **msg.value**); *// Triggering event*

}

}

See [Events](https://docs.soliditylang.org/en/v0.8.19/contracts.html#events) in contracts section for information on how events are declared and can be used from within a dapp.

**Errors**[**ℑ**](https://docs.soliditylang.org/en/v0.8.19/structure-of-a-contract.html#errors)

Errors allow you to define descriptive names and data for failure situations. Errors can be used in [revert statements](https://docs.soliditylang.org/en/v0.8.19/control-structures.html#revert-statement). In comparison to string descriptions, errors are much cheaper and allow you to encode additional data. You can use NatSpec to describe the error to the user.

*// SPDX-License-Identifier: GPL-3.0*

**pragma solidity** ^**0.8.4**;

*/// Not enough funds for transfer. Requested `requested`,*

*/// but only `available` available.*

error NotEnoughFunds(uint requested, uint available);

**contract** **Token** {

mapping(address => uint) balances;

function transfer(address to, uint amount) public {

uint balance = balances[**msg.sender**];

if (balance < amount)

revert NotEnoughFunds(amount, balance);

balances[**msg.sender**] -= amount;

balances[to] += amount;

*// ...*

}

}

See [Errors and the Revert Statement](https://docs.soliditylang.org/en/v0.8.19/contracts.html#errors) in the contracts section for more information.

**Struct Types**[**ℑ**](https://docs.soliditylang.org/en/v0.8.19/structure-of-a-contract.html#struct-types)

Structs are custom defined types that can group several variables (see [Structs](https://docs.soliditylang.org/en/v0.8.19/types.html#structs) in types section).

*// SPDX-License-Identifier: GPL-3.0*

**pragma solidity** >=**0.4.0** <**0.9.0**;

**contract** **Ballot** {

struct Voter { *// Struct*

uint weight;

bool voted;

address delegate;

uint vote;

}

}

**Enum Types**[**ℑ**](https://docs.soliditylang.org/en/v0.8.19/structure-of-a-contract.html#enum-types)

Enums can be used to create custom types with a finite set of ‘constant values’ (see [Enums](https://docs.soliditylang.org/en/v0.8.19/types.html#enums) in types section).

*// SPDX-License-Identifier: GPL-3.0*

**pragma solidity** >=**0.4.0** <**0.9.0**;

**contract** **Purchase** {

enum State { Created, Locked, Inactive } *// Enum*

}