Style Guide helps to maintain code layout consistent and make code more readable. Following are the best practices following while writing contracts with Solidity.

Code Layout

* **Indentation** − Use 4 spaces instead of tab to maintain indentation level. Avoid mixing spaces with tabs.
* **Two Blank Lines Rule** − Use 2 Blank lines between two contract definitions.

pragma solidity ^0.5.0;

contract LedgerBalance {

//...

}

contract Updater {

//...

}

* **One Blank Line Rule** − Use 1 Blank line between two functions. In case of only declaration, no need to have blank lines.

pragma solidity ^0.5.0;

contract A {

function balance() public pure;

function account() public pure;

}

contract B is A {

function balance() public pure {

// ...

}

function account() public pure {

// ...

}

}

* **Maximum Line Length** − A single line should not cross 79 characters so that readers can easily parse the code.
* **Wrapping rules** − First argument be in new line without opening parenthesis. Use single indent per argument. Terminating element ); should be the last one.

function\_with\_a\_long\_name(

longArgument1,

longArgument2,

longArgument3

);

variable = function\_with\_a\_long\_name(

longArgument1,

longArgument2,

longArgument3

);

event multipleArguments(

address sender,

address recipient,

uint256 publicKey,

uint256 amount,

bytes32[] options

);

MultipleArguments(

sender,

recipient,

publicKey,

amount,

options

);

* **Source Code Encoding** − UTF-8 or ASCII encoding is to be used preferably.
* **Imports** − Import statements should be placed at the top of the file just after pragma declaration.
* **Order of Functions** − Functions should be grouped as per their visibility.

pragma solidity ^0.5.0;

contract A {

constructor() public {

// ...

}

function() external {

// ...

}

// External functions

// ...

// External view functions

// ...

// External pure functions

// ...

// Public functions

// ...

// Internal functions

// ...

// Private functions

// ...

}

* **Avoid extra whitespaces** − Avoid whitespaces immediately inside parenthesis, brackets or braces.
* **Control structures** − Braces should open on same line as declaration. Close on their own line maintaining the same indentation. Use a space with opening brace.

pragma solidity ^0.5.0;

contract Coin {

struct Bank {

address owner;

uint balance;

}

}

if (x < 3) {

x += 1;

} else if (x > 7) {

x -= 1;

} else {

x = 5;

}

if (x < 3)

x += 1;

else

x -= 1;

* **Function Declaration** − Use the above rule for braces. Always add a visibility label. Visibility label should come first before any custom modifier.

function kill() public onlyowner {

selfdestruct(owner);

}

* **Mappings** − Avoid whitespaces while declaring mapping variables.

mapping(uint => uint) map;

mapping(address => bool) registeredAddresses;

mapping(uint => mapping(bool => Data[])) public data;

mapping(uint => mapping(uint => s)) data;

* **Variable declaration** − Avoid whitespaces while declaring array variables.

uint[] x; // not unit [] x;

* **String declaration** − Use double quotes to declare a string instead of single quote.

str = "foo";

str = "Hamlet says, 'To be or not to be...'";

Order of Layout

Elements should be layout in following order.

* Pragma statements
* Import statements
* Interfaces
* Libraries
* Contracts

Within Interfaces, libraries or contracts the order should be as −

* Type declarations
* State variables
* Events
* Functions

Naming conventions

* Contract and Library should be named using CapWords Style. For example, SmartContract, Owner etc.
* Contract and Library name should match their file names.
* In case of multiple contracts/libraries in a file, use name of core contract/library.

Owned.sol

pragma solidity ^0.5.0;

// Owned.sol

contract Owned {

address public owner;

constructor() public {

owner = msg.sender;

}

modifier onlyOwner {

//....

}

function transferOwnership(address newOwner) public onlyOwner {

//...

}

}

Congress.sol

pragma solidity ^0.5.0;

// Congress.sol

import "./Owned.sol";

contract Congress is Owned, TokenRecipient {

//...

}

* Struct Names

− Use CapWords Style like SmartCoin.

* Event Names

− Use CapWords Style like Deposit, AfterTransfer.

* Function Names

− Use mixedCase Style like initiateSupply.

* Local and State variables

− Use mixedCase Style like creatorAddress, supply.

* Constants

− Use all capital letters with underscore to seperate words like MAX\_BLOCKS.

* Modifier Names

− Use mixCase Style like onlyAfter.

* Enum Names

− Use CapWords Style like TokenGroup.