# Free TON Solidity support for Visual Studio Code

## Author

Telegram: @podlodkin

Twitter: <https://twitter.com/podlodkin82>

GitHub: <https://github.com/podlodkin>

Free TON address: 0:424c3fdf6ea1bff797767c3989e275e3f0ffdaf3a95c67cc3006b08c22923ac0

## Description

Free TON Solidity is the language used in Free TON to create smart contracts.

This extension provides:

* Syntax highlighting (keywords, variables, literals, comments and other things from the language specification and Free TON additions).
* Code completion (keywords, variable names, classes names, method names, interfaces)
* Intellisense (commands, contracts, methods, interfaces)
* Extension is fully customizable by VS Code native settings. For example, for colors of syntax highlighting.

## Solution

Visual Code marketplace (compiled and published extension)

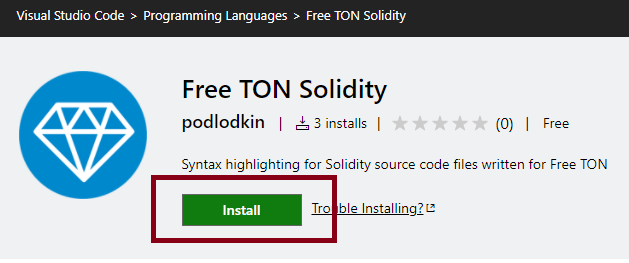
<https://marketplace.visualstudio.com/items?itemName=podlodkin.podlodkin-freeton-vscode-solidity>

GitHub (source code)

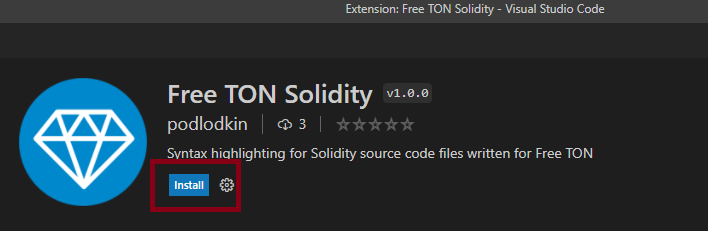
<https://github.com/podlodkin/podlodkin-freeton-vscode-solidity>

## How to install

1. Open VS Code marketplace URL - <https://marketplace.visualstudio.com/items?itemName=podlodkin.podlodkin-freeton-vscode-solidity>
2. Press the button “Install”:

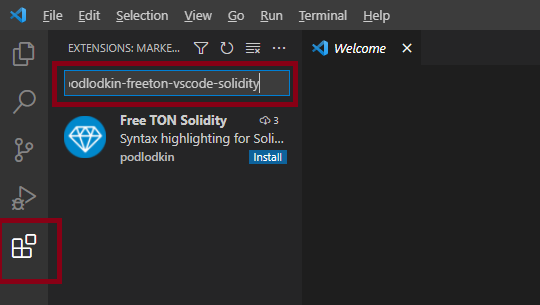


1. This extension must will be opening in VS Code application. After then press the button "Install" in VS Code application:



1. Enjoy – open any .ton file.

Alternative – you can use embedded Visual Code Marketplace – find extension by name “podlodkin-freeton-vscode-solidity”:



# Examples

## 10\_Wallet

<https://raw.githubusercontent.com/podlodkin/podlodkin-freeton-vscode-solidity/main/examples/10_Wallet.sol> or <https://raw.githubusercontent.com/tonlabs/samples/master/solidity/10_Wallet.sol>

pragma ton-solidity >= 0.35.0;

pragma AbiHeader expire;

/// @title Simple wallet

/// @author Tonlabs

contract Wallet {

/\*

Exception codes:

100 - message sender is not a wallet owner.

101 - invalid transfer value.

\*/

/// @dev Contract constructor.

constructor() public {

// check that contract's public key is set

require(tvm.pubkey() != 0, 101);

// Check that message has signature (msg.pubkey() is not zero) and message is signed with the owner's private key

require(msg.pubkey() == tvm.pubkey(), 102);

tvm.accept();

}

// Modifier that allows function to accept external call only if it was signed

// with contract owner's public key.

modifier checkOwnerAndAccept {

// Check that inbound message was signed with owner's public key.

// Runtime function that obtains sender's public key.

require(msg.pubkey() == tvm.pubkey(), 100);

// Runtime function that allows contract to process inbound messages spending

// its own resources (it's necessary if contract should process all inbound messages,

// not only those that carry value with them).

tvm.accept();

\_;

}

/// @dev Allows to transfer tons to the destination account.

/// @param dest Transfer target address.

/// @param value Nanotons value to transfer.

/// @param bounce Flag that enables bounce message in case of target contract error.

function sendTransaction(address dest, uint128 value, bool bounce) public pure checkOwnerAndAccept {

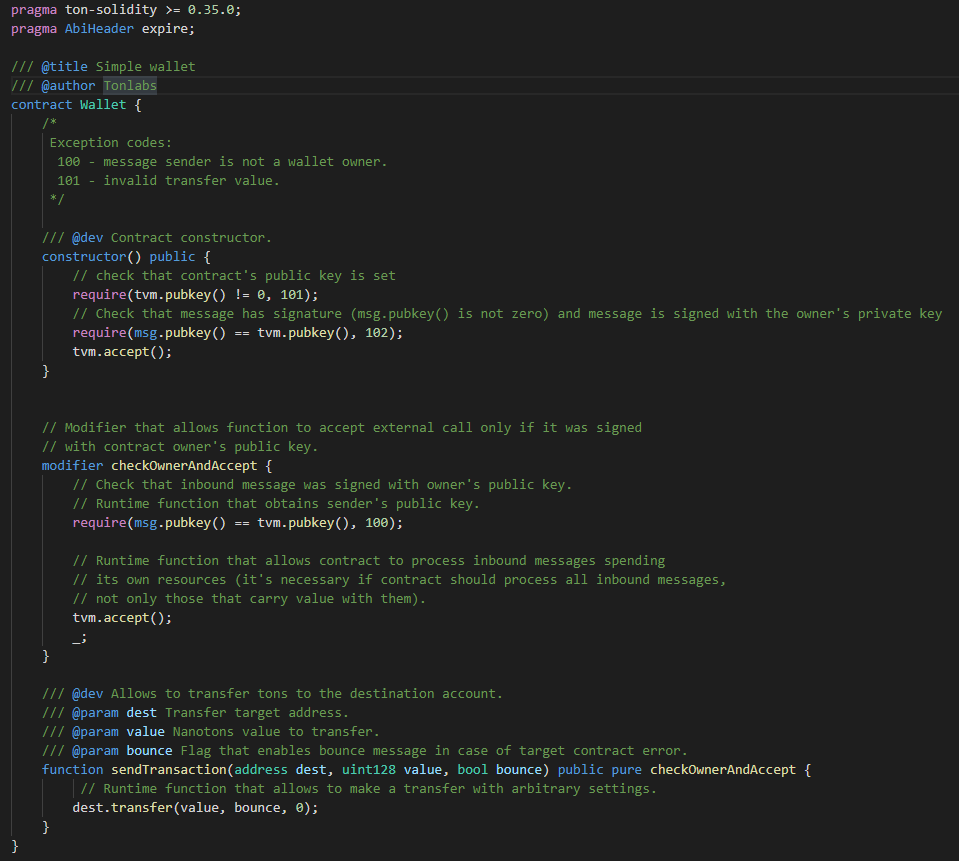
// Runtime function that allows to make a transfer with arbitrary settings.

dest.transfer(value, bounce, 0);

}

}

Results:



## 5\_Bank

<https://raw.githubusercontent.com/podlodkin/podlodkin-freeton-vscode-solidity/main/examples/5_Bank.sol> or <https://raw.githubusercontent.com/tonlabs/samples/master/solidity/5_Bank.sol>

pragma ton-solidity >= 0.35.0;

pragma AbiHeader expire;

// Import the interface file

import "5\_BankClientInterfaces.sol";

// This contract implements 'IBank' interface.

// The contract allows to store credit limits in mapping and give to the caller it's credit limits.

contract Bank is IBank {

// Struct for storing the credit information.

struct CreditInfo {

uint allowed;

uint used;

}

// State variable storing a credit information for addresses.

mapping(address => CreditInfo) clientDB;

constructor() public {

// check that contract's public key is set

require(tvm.pubkey() != 0, 101);

// Check that message has signature (msg.pubkey() is not zero) and message is signed with the owner's private key

require(msg.pubkey() == tvm.pubkey(), 102);

tvm.accept();

}

modifier checkOwnerAndAccept {

// Check that message was signed with contracts key.

require(msg.pubkey() == tvm.pubkey(), 102);

tvm.accept();

\_;

}

// Set credit limit for the address.

function setAllowance(address bankClientAddress, uint amount) public checkOwnerAndAccept {

// Store allowed credit limit for the address in state variable mapping.

clientDB[bankClientAddress].allowed = amount;

}

// Get allowed credit limit for the caller.

function getCreditLimit() public override {

// Cast caller to IMyContractCallback and invoke callback function

// with value obtained from state variable mapping.

CreditInfo borrowerInfo = clientDB[msg.sender];

IBankClient(msg.sender).setCreditLimit(borrowerInfo.allowed - borrowerInfo.used);

}

// This function checks whether message sender's available limit could be loaned

// and sends currency.

function loan(uint amount) public override {

CreditInfo borrowerInfo = clientDB[msg.sender];

if (borrowerInfo.used + amount > borrowerInfo.allowed) {

IBankClient(msg.sender).refusalCallback(borrowerInfo.allowed - borrowerInfo.used);

} else {

// '{value: amount}' allows to attach arbitrary amount of currency to the message

// if it is not set amount would be set to 10 000 000 nanoton

IBankClient(msg.sender).receiveLoan{value: uint128(amount)}(borrowerInfo.used + amount);

clientDB[msg.sender].used += amount;

}

}

}

Results:

