encode CLUB

Solidity Foundations

Lesson 2

Introduction to Smart Contracts
Coding HelloWorld.sol

Topics

1. Solidity
2. Remix
3. Storage.sol
4. Coding HelloWorld.sol

encode CLUB

What is Solidity

Solidity is an **object-oriented**, **high-level** language for implementing **smart contracts**. Smart contracts are programs which govern the behaviour of accounts within the Ethereum state.

Solidity is just one of the many programming languages that are able to compile **bytecodes** for Smart Contracts.

What is Solidity

Solidity is a curly-bracket language designed to target the **Ethereum Virtual Machine** (EVM). It is influenced by C++, Python and JavaScript.

Solidity is **statically typed**, supports **inheritance**, **libraries** and complex **user-defined types** among other features.

With Solidity you can create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets.

What is Solidity

Example: **USDT Token**

https://etherscan.io/token/0xdac17f958d2ee523a2206206994597c13d831ec7#code

Remix

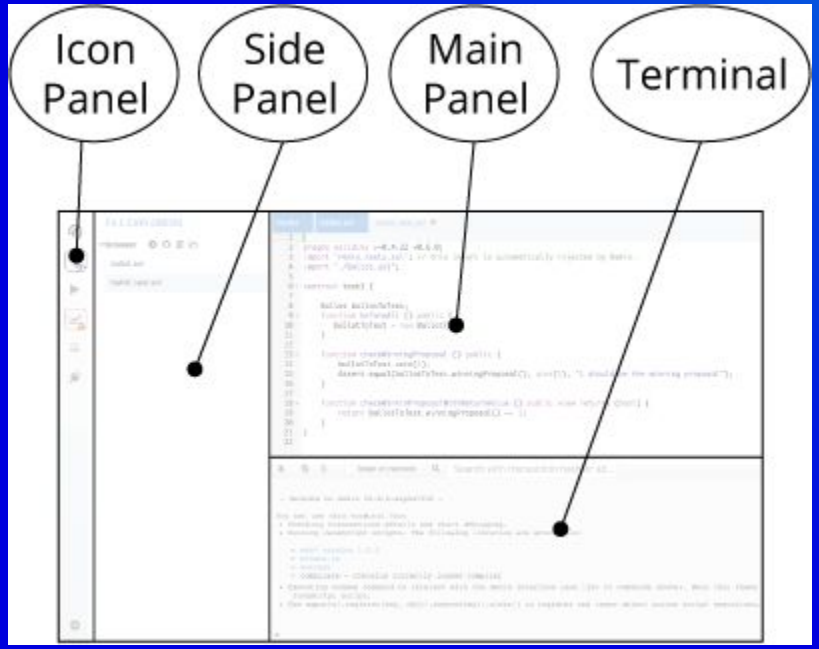
Remix IDE is used for **coding**, **compiling**, **deploying** and **interacting** with smart contracts. It is suitable for users of any knowledge level. It requires no setup, fosters a fast development cycle and has a rich set of plugins with intuitive GUIs.

Remix

Documentation: https://remix-ide.readthedocs.io/en/latest/

Browser IDE: https://remix.ethereum.org

Remix - Layout



Remix - Core plugins

File Explorer 

The File Explorers is where you can see the files.
profile name: **fileManager**

Solidity Compiler 

Compiles Solidity & YUL.
profile name: **solidity**

Deploy & Run 

Deploy & interact with smart contracts on the in-browser chain (JSVM), local nodes, and public networks.
profile name: **udapp**

Remix

encode CLUB

Let’s try it out

Remix example: Storage.sol

- Open file
- Compile
- Using a VM Blockchain
- Deploy
- Interact

Coding HelloWorld.sol

encode CLUB

Layout of a Solidity Source File

Source files can contain an arbitrary number of contract definitions, import , pragma and using for directives and struct, enum, function, error and constant variable definitions.

Important definitions

- SPDX License Identifier
- Version Pragma

Coding HelloWorld.sol

encode CLUB

Structure of a Contract

Contracts in Solidity are similar to classes in object-oriented languages. Each contract can contain declarations of **State Variables**, **Functions**, **Function Modifiers**, **Events**, **Errors**, **Struct Types** and **Enum Types**.

Furthermore, contracts can **inherit** from other contracts.

There are also special kinds of contracts called **libraries** and **interfaces**.

Coding HelloWorld.sol

encode CLUB

- **State Variables**: variables whose values are permanently stored in contract storage.
- **Functions**: the executable units of code.
- **Constructor**: an optional function which is executed upon contract creation, and where you can run contract initialisation code.
- **Visibility**: definition from where state variables and function may be accessed from.
- **State Mutability**: definition of what kinds of state changes are allowed to happen in a function call.

Coding HelloWorld.sol

encode CLUB

Let’s try it out

Understanding Storage.sol

Coding HelloWorld.sol

encode CLUB

- **Value Types**: variables that will always be **passed by value**. For example: they are always copied when they are used as function arguments or in assignments.
- **Reference Types**: Values of reference type can be **modified by reference**. Contrast this with value types where you get an independent copy whenever a variable of value type is used. Because of that, reference types have to be handled more carefully than value types. Currently, reference types comprise **structs**, **arrays** and **mappings**. If you use a reference type, you always have to explicitly provide the **data area** where the type is stored.

Coding HelloWorld.sol

encode CLUB

Data locations

- **Memory**: data lifetime is limited to each function call.
- **Storage**: the location where the state variables are stored, where the lifetime is limited to the lifetime of a contract.
- **Calldata**: special data location that contains the function arguments.

Coding HelloWorld.sol

encode CLUB

- **Special arrays**: variables of type **bytes** and **string** are **special arrays**.

Yes, strings are arrays for Solidity and the EVM.

This means that you need to specify **data location** for using strings in **function parameters** or **return values**.

Coding HelloWorld.sol

encode CLUB

Let’s try it out

Adapting Storage.sol to build HelloWorld.sol

- Modify code
- Compile
- Deploy
- Interact

Coding HelloWorld.sol

encode CLUB

Let’s try it out

Running HelloWorld.sol on a public testnet

- Connect to Metamask
- Deploy
- Interact