


Solidity Smart Contract

How to build Smart Contracts with Solidity

Daniel (Sơn) PHAM 

Solidity Developer Bootcamp



Ngày 22 tháng 7 năm 2023

1. **Workspaces for Solidity projects**
2. **Solidity**
3. **QnA**

1. Workspaces for Solidity projects

- 1.1 IDE (Integrated Development Environment)
- 1.2 Solidity framework/library

2. Solidity

3. QnA

Hiện nay có nhiều IDE phục vụ cho coding, và 1 số IDE hỗ trợ ngôn ngữ Solidity, hoặc thiết kế dành riêng để code Solidity.

- VS Code
- Remix IDE
- Truffle Suite

Khuyến khích dùng vscode vì tính linh hoạt, và có nhiều extensions hỗ trợ.

- solidity-visual-auditor
- hardhat-solidity
- foundry-snippets

Remix IDE.

Tiện lợi, phù hợp khi muốn kiểm tra nhanh 1 smart contract.

Foundry

Framework mạnh mẽ, thiết kế dành riêng cho solidity. Viết test và debug bằng ngôn ngữ solidity, tính tương thích rất cao với EVM. Forking chain, simulate transactions cực kì mạnh mẽ. Được đa số projects lớn sử dụng, và auditors ưa dùng.

Install

```
curl -L https://foundry.paradigm.xyz | bash  
foundryup
```

Hardhat

Được built dựa trên nodejs. Được đa số developer dùng.

Viết test bằng ts/js.

Dễ dàng customize cho mục đích sử dụng.

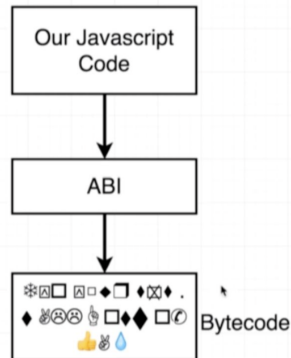
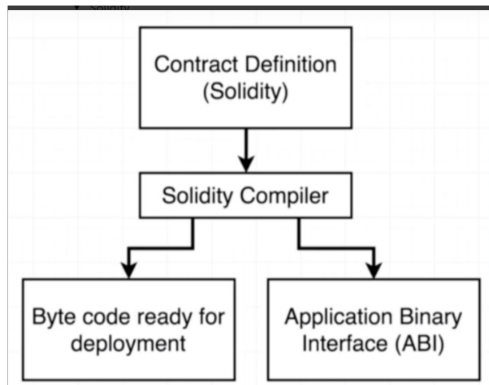
1. Workspaces for Solidity projects

2. Solidity

- 2.1 Solidity introduction
- 2.2 Smart Contract layout
- 2.3 Structure of a Contract
- 2.4 Types

3. QnA

Solidity introduction



- SPDX License Identifier
- Pragma
- Importing other Source Files
- Smart Contract code

Để tăng tính tin cậy, và tránh tranh chấp về bản quyền.

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

Sử dụng để bật các tính năng nhất định của compiler, hoặc các checks.

```
pragma solidity ^0.8.19;
```

Importing other Source Files

Import files khác

```
import "filename";
```

```
import * as symbolName from "filename";
```

```
import "filename" as symbolName;
```

```
import symbol1 as alias, symbol2 from "filename";
```

Structure of a Contract



Tương tự như các ngôn ngữ OOP (hướng đối tượng). Mỗi Contract có thể khai báo:

- State Variables
- Functions
- Function Modifiers
- Events
- Errors
- Struct Types
- Enum Types

Basic types



Basic Types

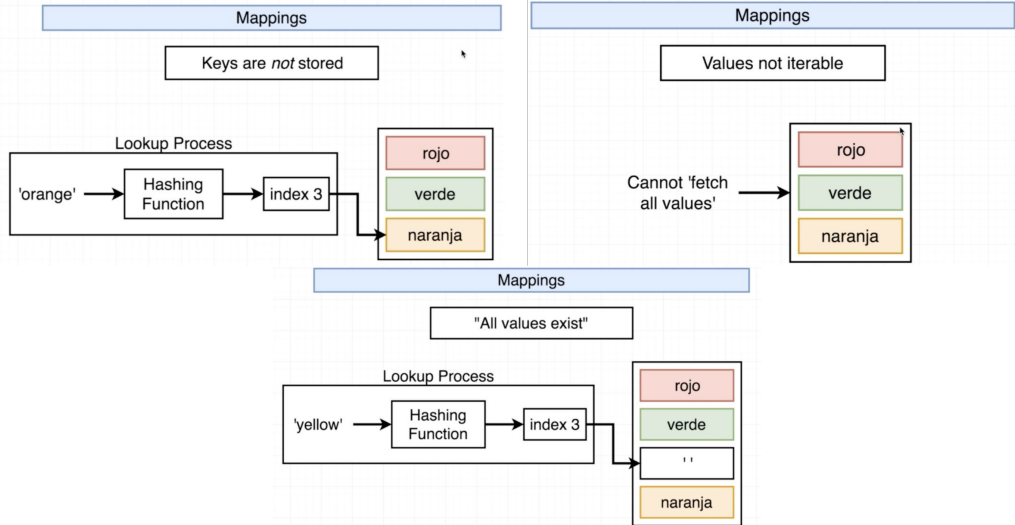
| Name | Notes | Examples | | |
|--------------|--|------------------------|-------------|--------|
| string | Sequence of characters | "Hi there!" | "Chocolate" | |
| bool | Boolean value | true | false | |
| int | Integer, positive or negative. Has no decimal | 0 | -30000 | 59158 |
| uint | 'Unsigned' integer, positive number. Has no decimal | 0 | 30000 | 999910 |
| fixed/ufixed | 'Fixed' point number. Number with a decimal after it | 20.001 | -42.4242 | 3.14 |
| address | Has methods tied to it for sending money | 0x18bae199c8dbae199c8d | | |

Arrays



| Reference Types | | |
|-----------------|---|--|
| Name | Notes | Examples |
| fixed array | Array that contains a <i>single type</i> of element. Has an unchanging length | <code>int[3] --> [1, 2, 3]</code> <code>bool[2] --> [true, false]</code> |
| dynamic array | Array that contains a <i>single type</i> of element. Can change in size over time | <code>int[] --> [1,2,3]</code> <code>bool[] --> [true, false]</code> |
| mapping | Collection of key value pairs. Think of Javascript objects, Ruby hashes, or Python dictionary. All keys must be of the same type, and all values must be of the same type | <code>mapping(string => string)</code> <code>mapping(int => bool)</code> |
| struct | Collection of key value pairs that can have different types. | <pre>struct Car { string make; string model; uint value; }</pre> |

Mappings



1. Workspaces for Solidity projects
2. Solidity
- 3. QnA**

