



klaytn

# KLAYTN DEV BOOTCAMP



# Buổi 3: Solidity cho người mới: Cơ bản về hợp đồng thông minh

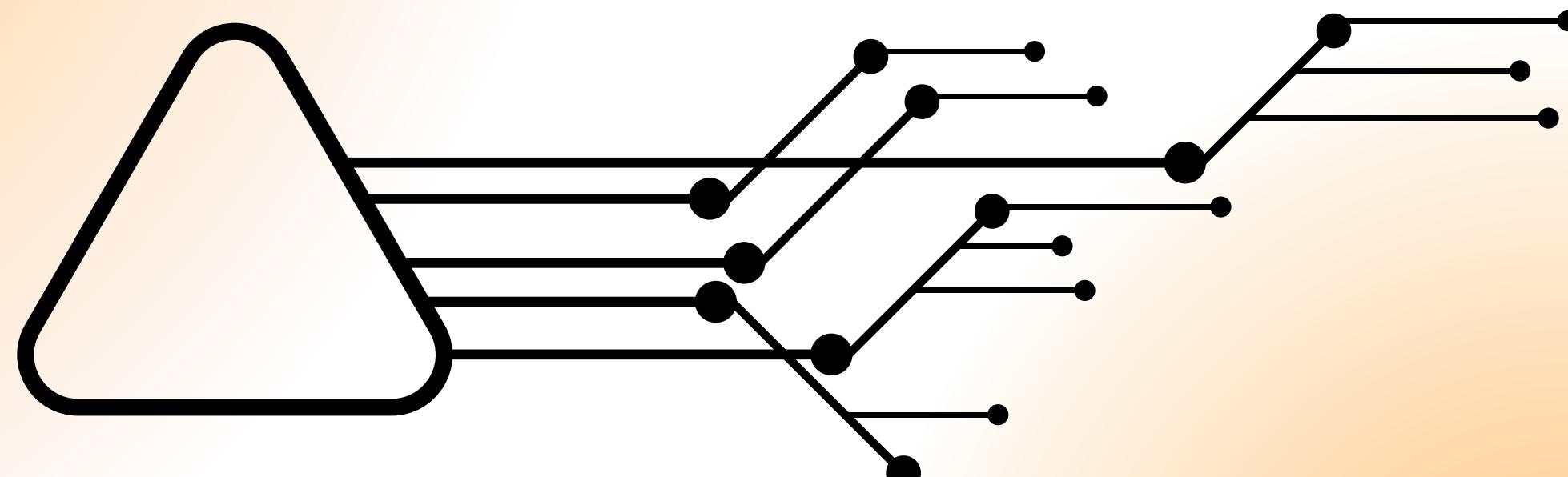


## STRUCT

Struct là một kiểu dữ liệu đặc biệt, cho phép chúng ta tạo một kiểu dữ liệu riêng với thành phần là một danh sách các biến.

### Ví dụ:

```
struct Car {  
    string make;  
    string model;  
    uint16 year;  
    uint16 horsepower;  
}
```



## MAPPING

Kiểu mapping cho phép chúng ta tạo ra các cặp giá trị key-value và lưu thành các danh sách.

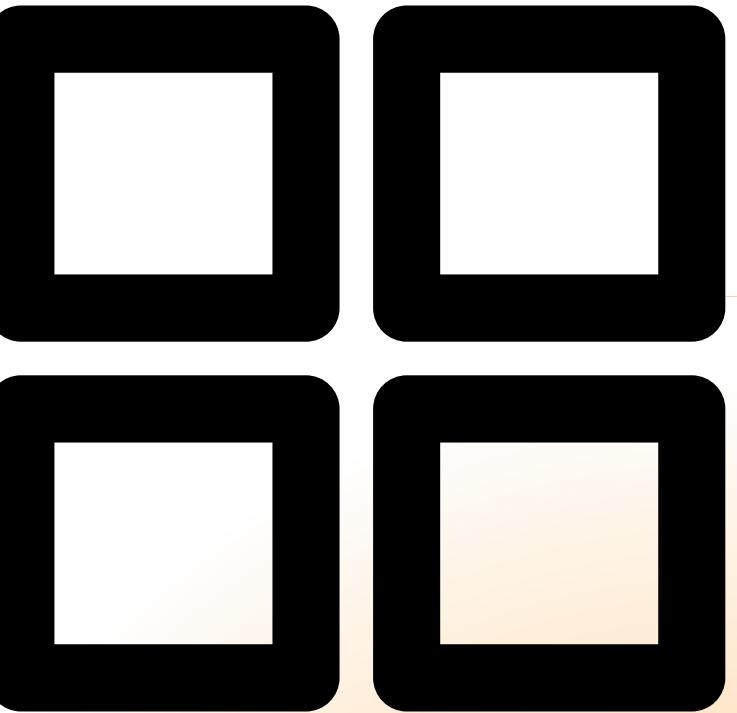
**Cú pháp:**

```
mapping(key_type => value_type) mappingName;
```



## MẢNG

- Mảng là một cách để lưu trữ dữ liệu tương tự trong một cấu trúc dữ liệu tập hợp.
- Mảng có thể có kích thước cố định hoặc động. Các chỉ số của mảng bắt đầu từ 0.



## VÒNG LẶP

- While loop

```
while (expression) {  
    Statement(s) to be executed if expression is true  
}
```

- Do ... while loop

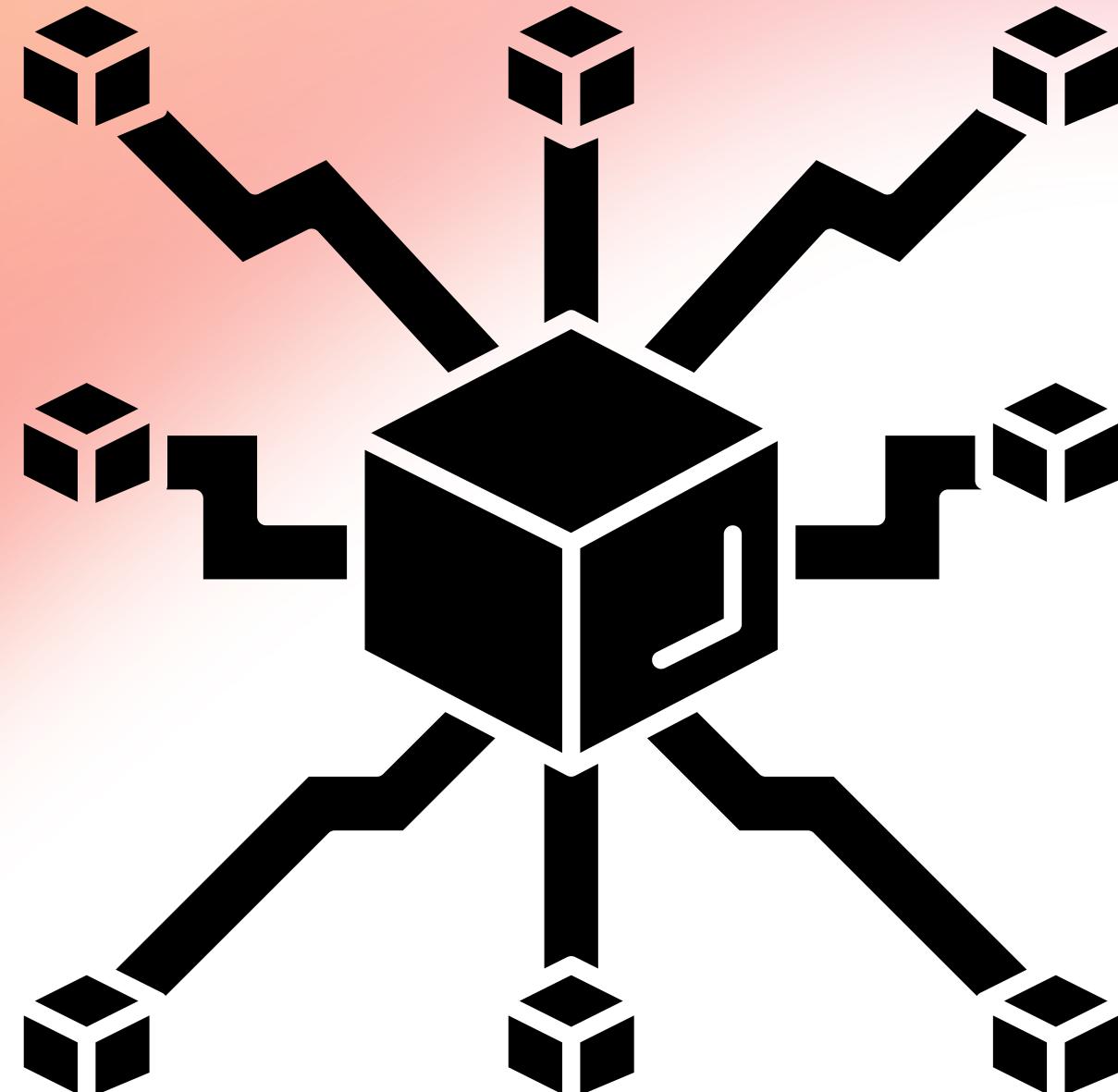
```
do {  
    i<10  
} while (//...);
```

- For loop

```
for (i=0; i< data.length; ++i) {  
    // ...  
}
```

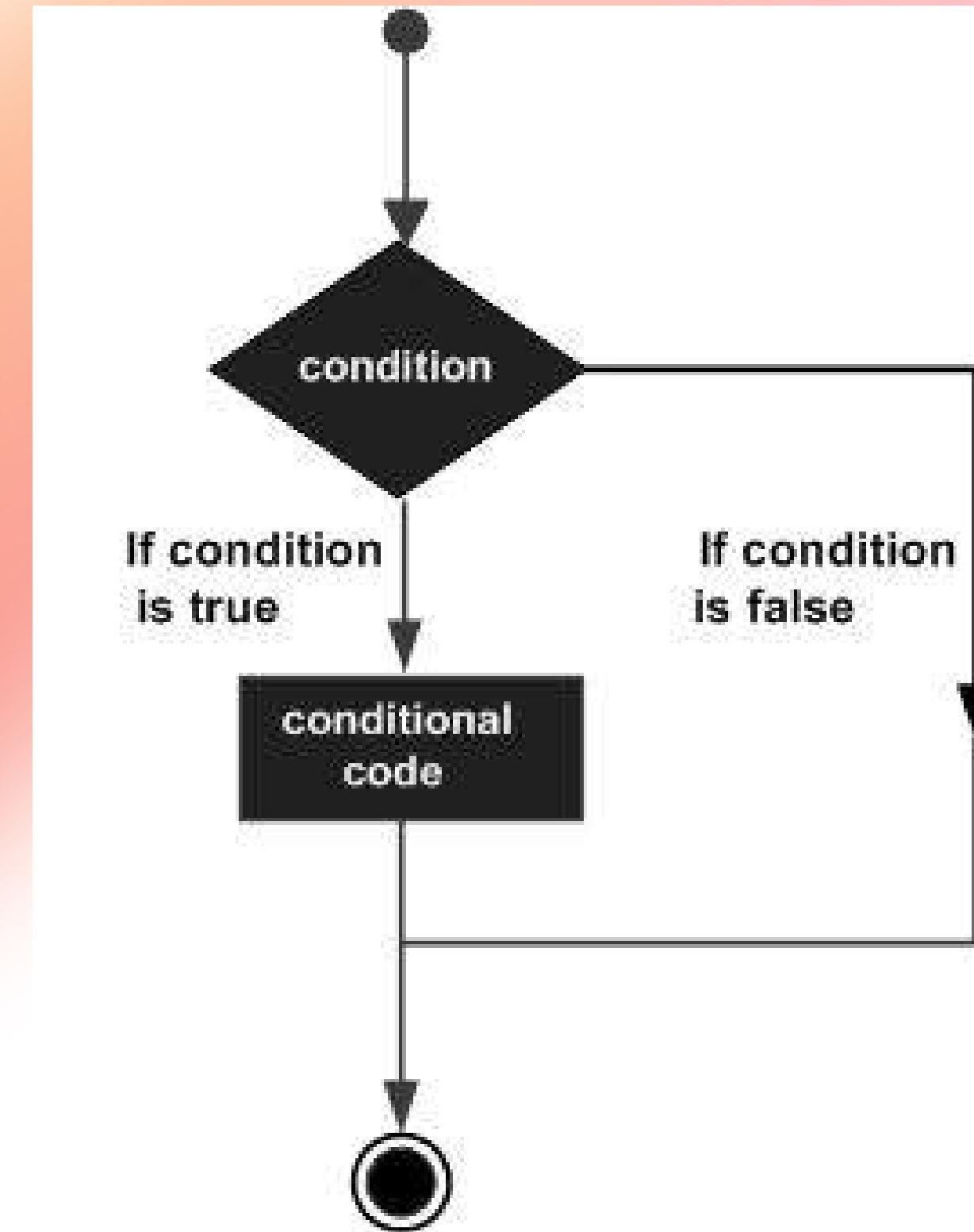
- break

- continue,



## ĐIỀU KIỆN

- if
- if ... else
- if ... else if ...



## CÁC HÀM ĐẶC BIỆT

**constructor:** Hàm khởi tạo của mỗi contract

**require, assert:** kiểm tra các điều kiện và đưa ra một exception nếu điều kiện không được đáp ứng.

**modifier:** kiểm tra các điều kiện trước khi các đoạn mã code trong phương thức đó được thực thi

## HÀM FALLBACK

Hàm fallback được thực thi nếu không có hàm nào khớp với mã định danh hàm hoặc không có dữ liệu nào được cung cấp cùng với lệnh gọi hàm

- Không có tên hoặc tham số.
- Nếu nó không được đánh dấu là payable, contract sẽ đưa ra một ngoại lệ nếu nó nhận được ether mà không có dữ liệu.
- Không thể hoàn trả.
- Chỉ có thể được xác định một lần cho mỗi hợp đồng.
- Nó cũng được thực thi nếu người gọi muốn gọi một hàm không tồn tại
- Bắt buộc để external
- bị giới hạn ở 2300 gas

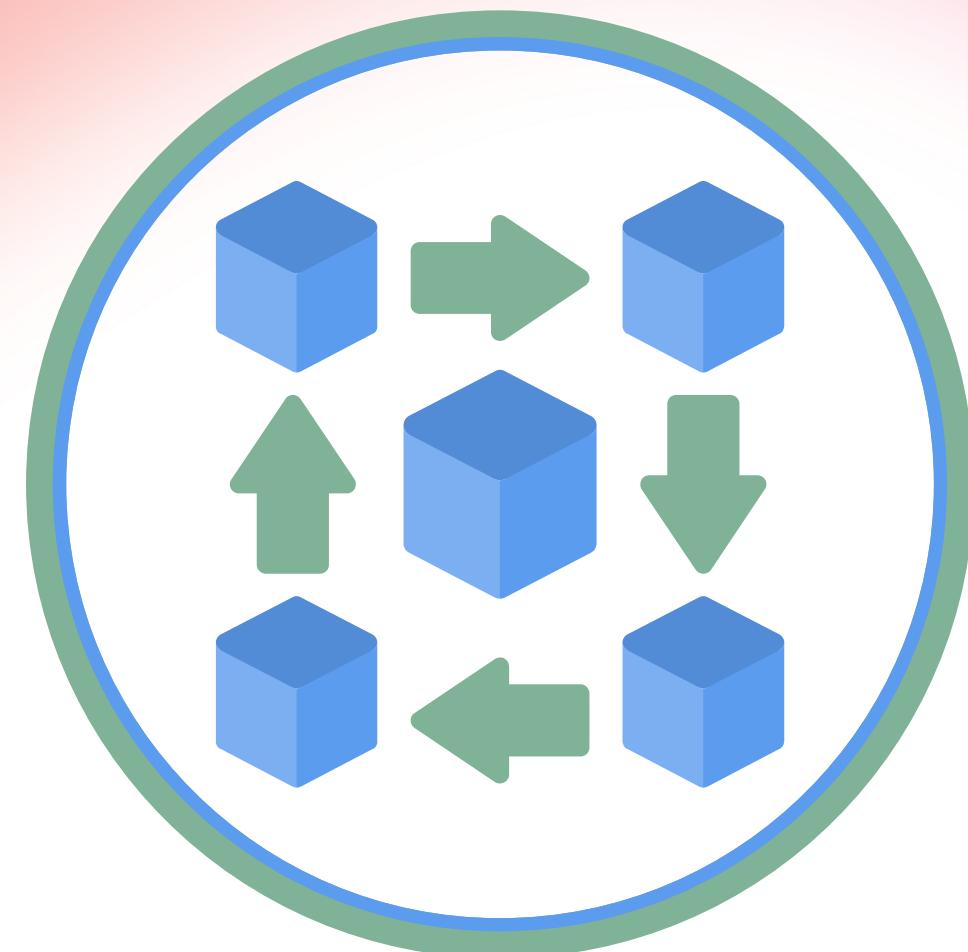
Ví dụ:

```
function () external {}  
receive()
```



## SECURITY PATTERN

- Checks-Effects-Interactions pattern:  
Kiểm tra - Thay đổi - Tương tác, quy trình  
đảm bảo kiểm tra điều kiện an toàn
- ```
function auctionEnd() public {
    // 1. Checks require(now >= auctionEnd);
    require(!ended);
    // 2. Effects
    ended = true;
    // 3. Interaction
    beneficiary.transfer(highestBid);
}
```



## FACTORY PATTERN

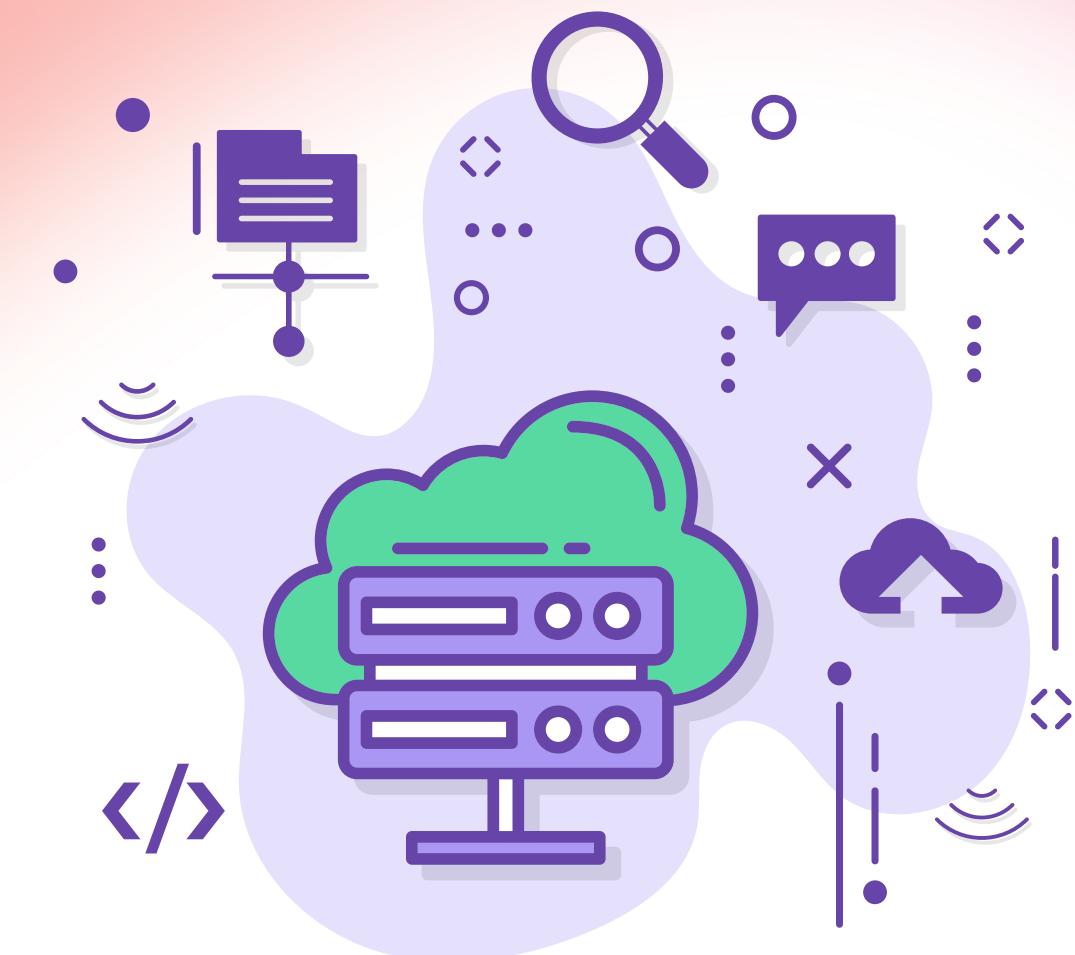
- Ý tưởng của mô hình nhà máy là có một hợp đồng (nhà máy) sẽ thực hiện việc tạo ra các hợp đồng khác. Động lực chính cho mẫu này trong lập trình dựa trên lớp xuất phát từ nguyên tắc chịu trách nhiệm duy nhất (một lớp không cần biết cách tạo các thể hiện của các lớp khác) và mẫu này cung cấp một sự trừu tượng cho hàm tạo.
- **Normal Factory Pattern:** Tạo hợp đồng Factory có chức năng xử lý việc tạo hợp đồng con và có thể thêm các chức năng khác để quản lý hiệu quả các hợp đồng này (ví dụ: tìm kiếm một hợp đồng cụ thể hoặc vô hiệu hóa hợp đồng)

Một số factory pattern khác: **Clone Factory Pattern**, ...

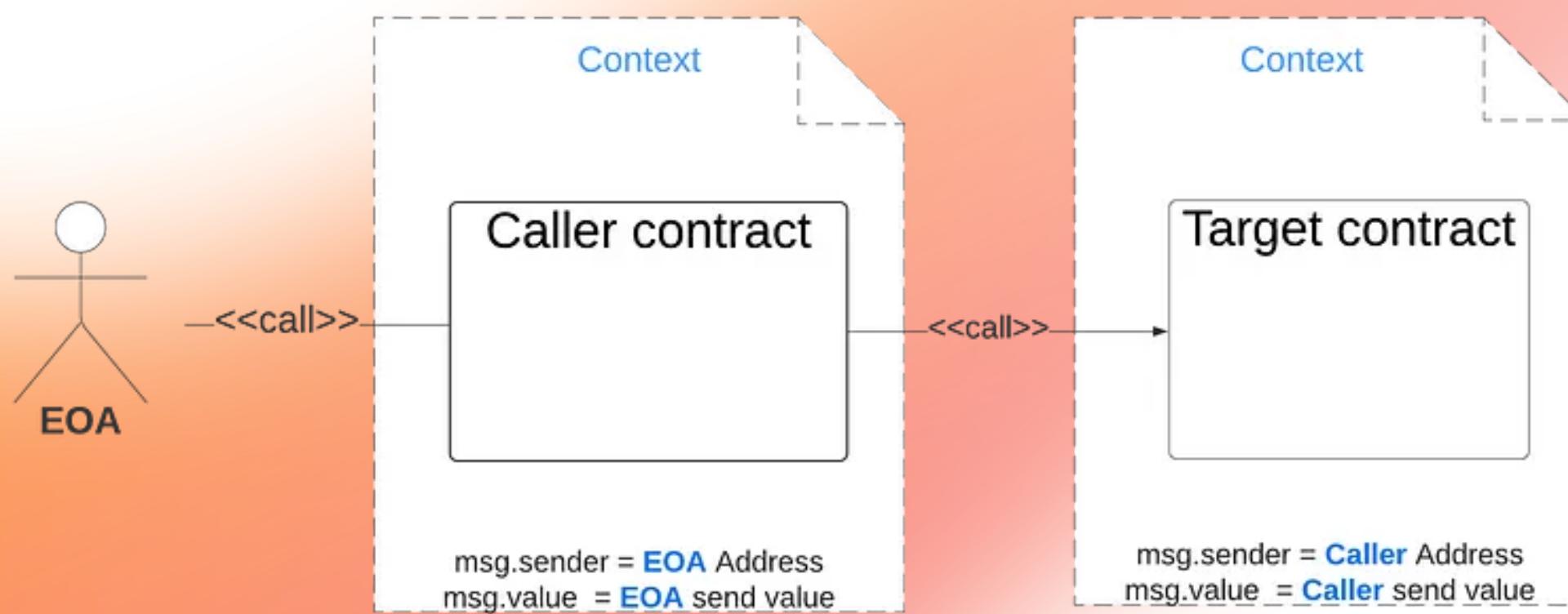


## PROXY

- Proxy contract: hợp đồng chuyển tiếp tất cả các cuộc gọi đến một hợp đồng thông minh khác, được gọi là hợp đồng triển khai
- Sử dụng:
  - Thêm Logic
  - Nâng cấp ảo (các hợp đồng hiện tại vẫn không thể thay đổi): một phiên bản mới có thể được triển khai và địa chỉ của nó thay thế phiên bản cũ trong bộ nhớ
  - Tránh phá vỡ sự phụ thuộc của các hợp đồng khác đang tham chiếu hợp đồng được nâng cấp



# CALL



- giá trị được thay đổi cho dù được truyền vào qua tham số và được thực thi ở contract Caller nhưng trạng thái thay đổi được phản ánh vào contract Target.

## DELEGATE CALL



- `msg.sender` được giữ nguyên như ban đầu không thay đổi và state thay đổi ở context của Caller contract, chỉ mượn code của Target contract rồi thực thi trên context của contract Callet.

# PROXY

## Solidity

Upgradeable smart contract using EIP1967 proxy need 3 smart contract:

- Proxy contract (users interact with)
- Implementation contract (your contract)
- ProxyAdmin contract

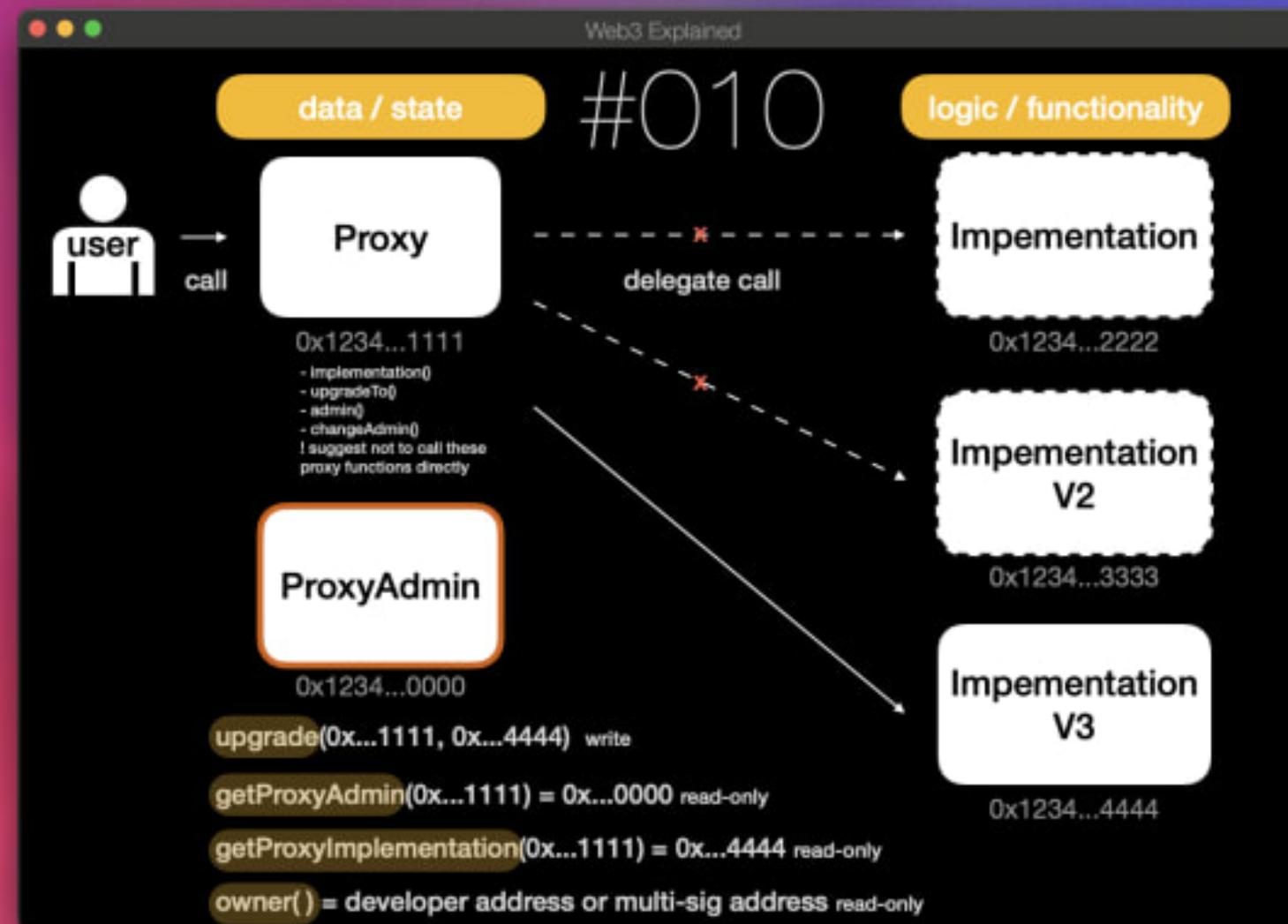
Upgrade in two steps:

1. deploy new implementation
2. upgrade in ProxyAdmin



### How upgradeable smart contract works? How proxy contract works?

Web3 Explained #010



## CÁC THƯ VIỆN

Các thư viện trong solidity tương tự như các hợp đồng chứa các mã có thể tái sử dụng.

```
library <libraryName> {  
    // block of code  
}
```

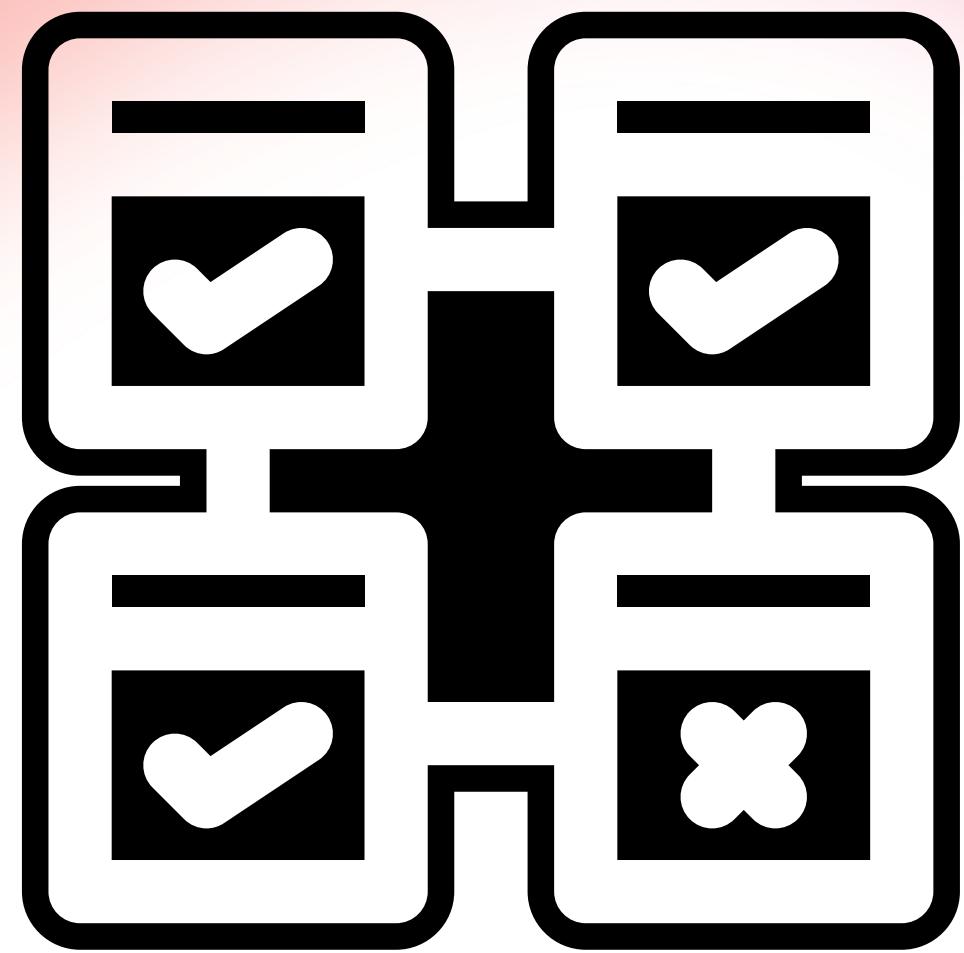
Các thư viện nổi bật:

- Openzeppelin
- DappSys
- HQ20



## UNIT TEST

- Viết unit test để đảm bảo rằng hợp đồng của bạn hoạt động như mong đợi trong các tình huống khác nhau.
- Remix IDE:
  - beforeEach() - Runs before each test
  - beforeAll() - Runs before all tests
  - afterEach() - Runs after each test
  - afterAll() - Runs after all tests



TIẾP THEO

# Tạo Dapp (Phần I): Viết 1 SC bỏ phiếu cộng đồng