10 câu hỏi cơ bản của blockchain developer

1. Ethereum smart contract là gì?
   * Là một chương trình máy tính được viết bằng các ngôn ngữ như solidity, được deploy lên blockchain và tự động thực hiện khi đáp ứng được các điều kiện.
2. Điều đặc biệt nhất của smart contract là gì?
   * Điều đặc biệt là nó sẽ không thể thay đổi sau khi đã deploy và hoạt động không cần bên trung gian nào cả.
3. Có thể gọi một smart contract từ một smart contract khác không?
   * Có hoàn toàn có thể gọi một smart contract từ một smart contract khác trong Ethereum. Đây là một trong những tính năng mạnh mẽ nhất của EVM (ethereum virtual machine)
4. Từ một smart contract có thể gọi được 1 API bên ngoài được không?
   * Không thể gọi API bên ngoài blockchain từ một smart contract bởi vì smart contract được chạy trên môi trường deterministic (các node phải giống nhau mất sự đồng nhất) nên dữ liệu thay đổi theo thời gian sẽ làm mất tính toàn vẹn của blockchain
   * Giải pháp dùng: Oracle pattern như chainlink (dùng để giới thiệu dữ liệu vào trong chứ không thể gọi từ smart contract)
5. Smart contract có lưu nhiều dữ liệu không?
   * Thường là không vì khi lưu dữ liệu trên blockchain mình phải trả phí gas. Dữ liệu càng lớn thì phí gas càng cao
6. Ngôn ngữ nào được dùng để code smart contract?
   * Ngoài solidity ra thì Vyper,… dù code ngôn ngữ nào thì cũng sẽ compile ra EVM bytecode để EVM có thể hiểu được
7. Có thể code nhiều smart contract trong cũng một file không?
   * Được mình chỉ cần định nghĩa từ khóa contract nhiều lần trong cùng 1 file solidity
8. Solidity là dynamic-type hay static-type?
   * Solidity là static-type có nghĩa là mình phải định nghĩa kiểu dữ liệu của một biến trước khi mình sài đó
9. Bạn dùng công cụ nào để xem và lấy dữ liệu blockchain?
   * Mình sẽ xem tất cả thông tin thông qua cái block explorer (đa phần các blockchain đều có block explorer riêng trong đó bao gồm các thông tin của các transaction trên blockchain đó).
10. ABI là gì?
    * Application binary interfact (được sinh ra sau khi compile 1 smart contract) là một signature của smart contract sẽ chứa những thông tin của các hàm (kiểu dữ liệu, tham số nhận vào là gì,…)
    * Được dùng bởi bên thứ 3 như Web3, eitherjs để giao tiếp với smart contract.

**Solidity**

**🔒 1. Bảo mật**

Đây là ưu tiên hàng đầu:

|  |  |
| --- | --- |
| **Vấn đề** | **Mô tả** |
| **Reentrancy Attack** | Khi gọi một hợp đồng khác, nếu không cẩn thận sẽ bị gọi ngược và mất tiền. Dùng checks-effects-interactions để tránh. |
| **Integer Overflow / Underflow** | Sử dụng SafeMath hoặc Solidity ≥ 0.8.0 (đã tự xử lý). |
| **Visibility** | Luôn khai báo rõ public, private, internal, external. Mặc định là public. |
| **DoS (Denial of Service)** | Không lặp quá dài hoặc gọi contract không kiểm soát được. |
| **Access Control** | Dùng require(msg.sender == owner) hoặc Ownable để giới hạn quyền. |

**🧠 2. Logic & Cấu trúc**

|  |  |
| --- | --- |
| **Khái niệm** | **Mô tả** |
| **Constructor** | Chạy một lần khi deploy. Dùng để khởi tạo biến. |
| **Modifier** | Dùng để kiểm tra điều kiện trước khi chạy hàm. |
| **Events** | Ghi lại log trên blockchain (giao tiếp frontend). |
| **Fallback / Receive** | Nhận ETH khi không gọi được hàm nào cụ thể. |
| **Gas Optimization** | Sắp xếp storage hợp lý, dùng uint8/uint16 nếu có thể. |

**🧱 3. Storage vs Memory vs Calldata**

|  |  |
| --- | --- |
| **Loại** | **Mô tả** |
| storage | Lưu trên blockchain, tốn gas, dùng cho biến trạng thái. |
| memory | Chỉ tồn tại khi gọi hàm, dùng ít gas hơn. |
| calldata | Dữ liệu input không thay đổi được (dùng trong external). |

**🔧 4. Các từ khóa thường dùng**

|  |  |
| --- | --- |
| **Từ khóa** | **Ý nghĩa** |
| require() | Dừng hàm nếu điều kiện sai. |
| assert() | Kiểm tra logic nội bộ, dừng nếu sai (tốn gas hơn). |
| revert() | Hủy bỏ transaction kèm thông báo. |
| msg.sender | Địa chỉ người gọi hàm. |
| msg.value | Số ETH gửi kèm. |
| address(this) | Địa chỉ của contract hiện tại. |