<Solidity 언어 정리>

1. msg.sender 함수 : 현재 함수를 호출한 사람의 주소를 가리키는 값(컨트랙트를 배포한 사람)
2. 디폴트 가시값은 public 이다.
3. Zombie[] public zombies => public 사용해서 다른 컨트랙트에서도 읽을 수 있게 함
4. 함수선언
    A. Function a(string _name, uint _dna)
    B. Function a(uint _number) private { zombies.push(_number)}; //외부에서 이 함수를 호출 못함
    C. Function a() public returns (string){ return "a"} => returns와 반환값을 적어준다.
5. 배열에 구조체 추가
    A. zombies.push(a);//zombie형태의 값을 배열의 끝에 추가해서 순서를 유지시킴
6. View
    A. 데이터를 읽기만하고 쓰기를 사용하지 않을 때 사용한다.
    B. Function a() public view returns (string) { return "a"} => view 때문에 이 함수 안에서는 쓰기 못함
7. Pure
    A. 함수가 앱에서 어떤 데이터도 접근하지 않을 때 사용 -> (view기능 + 읽는것 금지)
8. keccak256
    A. 내장 해시함수로 값을 해시로 바꿔줌
9. 이벤트
    A. 컨트랙트가 블록체인 상에서 어떤 액션이 발생했을 때 의사소통 하는 방법
    B. event a(uint x, uint y);
10. 솔리디티는 고유의 스트링 비교 기능을 가지고 있지 않아 해시함수 적용 후 비교한다.
11. Require 함수 : 특정 조건이 참이 아닐 때 함수가 에러 메시지를 발생하고 실행을 멈춘다.
    A. require(now >= 60) => now가 60보다 크면 해당 require 함수를 계속 실행, 작으면 에러 발생하고 함수 실행 멈춤
12. 상속 : contract a is b -> 이 형태로 쓰이며, 코드를 쉽게 관리하기 위해 사용된다.
13. Import : 다른 sol파일을 불러온다. -> import "./a.sol" -> './' 의 의미는 해당 컨트렉트와 동일한 폴더임을 의미한다.
14. 변수 저장공간 : 1. Storage(블록체인상 영구적 저장되는 변수로 외부에서 선언된다), 2. Memory(임시적 저장되는 변수)
    B. 상태변수 디폴트 : storage
        i. Uint storage a = 100; // 상태변수로, 영구적 저장됨
    C. 함수 내부 변수 디폴트 : memory => 함수가 끝날 때까지만 존재
        i. Memory 배열은 반드시 길이 인수와 함께 생성되야함

ii. Uint[] memory a = new uint[](3); => 배열길이가 3인 memory생성

15. 접근자 :

D. Internal (private과 비슷하지만 상속하는 컨트랙트에서 접근이 가능) ,

E. External (public과 비슷하지만 컨트랙트 바깥에서만 호출될 수 있다.) => 누구나 외부에서 이 함수를 호출할 수 있음

16. 인터페이스 : contract Interface{ }

17. 컨트랙트 특징 : 컨트랙트는 배포 후 변경할 수 없다. -> 블록체인에 영구적으로 존재하게 됨

18. OpenZeppelin : DApp에서 사용할 수 있는 안전하고 검증받은 스마트 컨트랙트 라이브러리

19. 함수 제어자 : 함수처럼 보이지만 function 키워드 대신 modifier 키워드 사용

F. 소유권이 있는 사용자만이 해당 함수를 호출 할 수 있도록 하는 것

G. 함수 호출하듯 직접 호출 할 수 없다.

H. Function likeA() external onlyOwner{}; -> 여기서 onlyOwner가 함수 제어자이다.

i. Ownable.sol 파일에 " modifier onlyOwner(){ require(msg.sender == owner } "의 형태로 저장되어있다.

I. LikeA() 함수 호출 시 onlyOwner 의 코드가 먼저 실행됨

J. onlyOwner 의 " _; " 부분에서 다시 likeA() 함수로 돌아와 실행됨

20. 가스 : Dapp의 함수를 실행할 때 지불하는 화폐

K. 논리구조에 따라 가스의 요구량이 달라진다.

L. 저장 공간 뿐만 아니라 연산 사용시간에 따라서도 비용을 지불해야 한다.

21. Uint의 크기에 상관없이 256비트의 저장공간을 미리 잡아놓는다.

M. 하지만 구조체 안에서 uint의 크기를 더 작게 만든다면 솔리디티에서 더 작은 공간을 차지하도록 압축한다.

N. 따라서 구조체 안에서는 가능한 한 작은 크기의 정수타입을 사용하는게 좋다.

O. 구조체 안에서 동일 데이터 타입을 모아놓는 것이 저장공간 최소화에 도움된다.

22. 시간 단위(Time units) : now 변수 이용하면 유닉스 타임스탬프 값을 얻을 수 있다.

P. Seconds, minutes, hours, days, weeks, years 같은 시간 단위를 uint 숫자로 자동 변환해준다.

i. Ex) 1 hours : 3600(60초 * 60분)

ii. now는 현재 시간을 저장하고 있는 변수임

iii. now + 5 minutes => 이렇게 사용 가능 => 현재시간 + 5분 으로 됨

23. View 함수는 가스를 소모하지 않는다. -> 블록체인상 어떤것도 수정하지 않기 때문에

Q. 트랜젝션은 모든 개별 노드에서 실행되어야 하고 가스를 소모함

R. 하지만 다른 함수 내부에 있는 view는 가스를 소모한다.

24. 솔리디티에서 비싼 연산은 storage 이다. 그 중 쓰기 연산이 비싸다.