基础知识类1-5

面试问答题（中英文）

基础知识类（1-5）

1.difference between private, internal, public and external functions?

私有、内部、公共和外部函数之间的区别?

答:

Private can only be used within the current contract, subcontract inheritance can not be called; internal can be called by the current contract and subcontracts; public contract internal and external and subcontracts can be called; external only provided to the external call, the contract can not be called within the contract, the contract interface functions should be declared as external.

私有只能当前合约内部使用，子合约继承也不能调用；内部可以当前合约和子合约调用；公共合约内部外部和子合约都可以调用；外部只提供给外部调用，合约内部不能调用，合约接口的函数都要声明成external。

2.Approximately how big can the smart contract size be?

智能合约大小大约可以有多大?

答:

After Shanghai upgraded, the realisation of EIP-3860 is 48KB, the original 24KB, expanding the amount of code to solve the problem of high gas fees between contracts calling each other after splitting the previous complex contract.

上海升级后，实现EIP-3860是48KB，原来24KB，扩大代码量解决以前复杂合约拆分后合约之间互相调用的高gas费问题。

3.What is the difference between create and create2?

create 和 create2 之间有什么区别?

答：

CREATE and CREATE2 are two important opcodes in Solidity that can be used to deploy smart contracts, but there are some differences between them. The CREATE opcode calculates the address of a new contract by hashing the sender's address and the nonce value, while the CREATE2 opcode uses a more complex formula that includes parameters such as the sender's address, a random number, and a bytecode to calculate the address of a new contract. The main advantage of the CREATE2 opcode is that it can predict the address of a contract before it is deployed, which is useful in scenarios where the address of the contract is known in advance.

CREATE和CREATE2是Solidity中的两个重要的操作码，它们都可以用于部署智能合约，但是它们之间有一些区别. CREATE操作码通过对发送者地址和nonce值进行哈希运算来计算新合约的地址，而CREATE2操作码则使用了更复杂的公式来计算新合约的地址，这个公式包括发送者地址、随机数、字节码等参数. CREATE2操作码的主要优点是，它可以在部署合约之前预测合约的地址，这对于需要提前知道合约地址的场景非常有用。

4.What are the major changes to arithmetic operations in Solidity version 0.8.0?

Solidity 0.8.0 版本对算术运算有什么重大变化?

答：

As of Solidity version 0.8.0, arithmetic operations are rolled back on underflow and overflow. This means that if the result of an arithmetic operation is out of range for its datatype, the operation will fail and roll back. Prior to Solidity 0.8.0, integer underflows and overflows were allowed and did not result in an error. To avoid this, Solidity 0.8.0 turned on arithmetic operation checking by default, and if you need to use the old way of doing arithmetic operations, you can use the unchecked{...} statement block

从Solidity 0.8.0版本开始，算术运算会在下溢和上溢时回滚。这意味着，如果一个算术运算的结果超出了其数据类型的范围，那么这个运算将会失败并回滚。在Solidity 0.8.0之前，整数下溢和上溢是允许的，不会导致错误，为了避免这种情况，Solidity 0.8.0默认开启了算术运算检查，如果您需要使用旧的算术运算方式，可以使用unchecked{...}语句块。

5.Is it better to use a map or an array for an address allowlist? Why?

对于地址 allowlist，使用映射还是数组更好？为什么?

答：

An address allowlist is a list of stored addresses that restricts certain operations to only those addresses in the list. In Solidity, an address allowlist can be implemented as either a map or an array, with the advantage of using a map to quickly find out if an address is in the allowlist, and the advantage of using an array to more easily traverse the entire allowlist.

If you need to quickly find out if an address is in an allowlist, then it may be better to use a mapping. Mapping is implemented using a hash table and can find out if an address is in an allowlist in O(1) time. This is especially useful for large allowlists, as it avoids having to traverse the entire list to find an address.

If you need to traverse the entire allowlist, it may be better to use an array. Arrays make it easier to traverse the entire list because they are contiguous chunks of memory. This is especially useful for small allowlists, as it avoids the extra overhead of using a hash table.

地址 allowlist 是一个存储地址的列表，用于限制只有在列表中的地址才能执行某些操作。在 Solidity 中，可以使用映射或数组来实现地址 allowlist。使用映射的优点是可以快速查找地址是否在 allowlist 中，而使用数组的优点是可以更容易地遍历整个 allowlist。

如果您需要快速查找地址是否在 allowlist 中，那么使用映射可能更好。映射使用哈希表实现，可以在 O(1) 的时间内查找地址是否在 allowlist 中。这对于大型 allowlist 尤其有用，因为它可以避免遍历整个列表来查找地址。

如果您需要遍历整个 allowlist，那么使用数组可能更好。数组可以更容易地遍历整个列表，因为它们是连续的内存块。这对于小型 allowlist 尤其有用，因为它可以避免使用哈希表的额外开销。