技术实现类11-15

面试问答题（中英文）

技术实现类 Technology Realization Category（11-15）

11、在权益证明之前后，block.timestamp 发生了什么变化?

What happened to block.timestamp before and after proof of stake?

答：

在PoW协议中，block.timestamp表示矿工开始挖掘新块的时间戳。在PoS协议中，block.timestamp表示验证器开始验证新块的时间戳。因此，block.timestamp的含义在两种协议中都与块的创建时间有关，但在PoS协议中，它与验证器的行为有关，而不是矿工的行为有关。

In the PoW protocol, block.timestamp represents the timestamp when the miner starts mining the new block. In the PoS protocol, block.timestamp represents the timestamp when the verifier starts verifying the new block. Thus, the meaning of block.timestamp is related to the creation time of the block in both protocols, but in the PoS protocol, it is related to the behavior of the validator, not the behavior of the miner.

12、代理中的函数选择器冲突是什么，它是如何发生的?

What is a function selector conflict in an agent and how does it occur?

答：

函数选择器是一个用于标识函数的哈希值。在Solidity中，当您调用一个合约中的函数时，您需要提供该函数的选择器。函数选择器由函数名称和参数类型组成，并使用Keccak-256哈希算法进行哈希处理。当您在代理合约中调用另一个合约的函数时，您需要将该函数的选择器传递给代理合约。如果您在代理合约中定义了具有相同名称和参数类型的函数，则会发生函数选择器冲突。这意味着当您调用代理合约中的函数时，Solidity无法确定您要调用哪个函数，因为它们具有相同的函数选择器。为了避免函数选择器冲突，您可以使用不同的函数名称或参数类型来定义代理合约中的函数。或者使用管理员校验来调用，只有管理员才调用代理合约定义的函数。

A function selector is a hash value used to identify a function. In Solidity, when you call a function in a contract, you need to provide a selector for that function. The function selector consists of the name of the function and the type of the argument and is hashed using the Keccak-256 hash algorithm. When you call a function from another contract in a proxy contract,

you need to pass the selector for that function to the proxy contract. A function selector conflict occurs if you define functions with the same name and argument types in the proxy contract. This means that when you call a function in the proxy contract, Solidity cannot determine which function you want to call because they have the same function selector. To avoid function selector conflicts, you can define functions in the proxy contract with different function names or parameter types. Or you can use administrator checksums to call them, so that only administrators call the functions defined in the proxy contract.

13、payable 函数对 gas 的影响是什么?

How does the payable function affect gas?

答:

payable函数是一种特殊类型的Solidity函数，它允许合约接受以太币作为支付。当您在Solidity中定义一个payable函数时，您可以在函数调用中包含以太币，并将其存储在合约的余额中。由于以太币是一种有价值的加密货币，因此在调用payable函数时需要支付一定的gas费用。这是因为在以太坊网络中，每个操作都需要消耗一定数量的gas，以保证网络的安全性和可靠性。

当您在Solidity中使用payable函数时，需要注意以下几点：

1.确保您的合约具有足够的余额来处理以太币支付。

2.确保您的合约具有足够的gas来处理以太币支付。

3.确保您的合约具有足够的安全性来处理以太币支付。

The payable function is a special type of Solidity function that allows contracts to accept Ether as payment. When you define a payable function in Solidity, you can include Ether in the function call and store it in the contract's balance. Since Ether is a valuable cryptocurrency, there is a gas fee that needs to be paid when calling a payable function. This is because in the Ether network, each operation consumes a certain amount of gas to keep the network secure and reliable.

When you use the payable function in Solidity, you need to pay attention to the following points:

1. Make sure your contract has enough balance to handle Ether payments.

2. Make sure your contract has enough gas to process Ether payments.

3. Make sure your contract has enough security to handle ethereum payments.

14、函数参数中的 memory 和 calldata 有什么区别?

What is the difference between memory and calldata in function arguments?

答：

memory：用于声明函数参数将被存储在内存中。内存中的数据只在函数执行期间存在，执行完毕后就被销毁。在函数内部，您可以使用memory关键字来创建临时变量，但是不能在函数之外使用它们。在函数调用期间，函数参数的值将从调用方复制到内存中，并在函数执行完毕后被销毁。

calldata：用于声明函数参数将被存储在调用数据区域中。调用数据区域是一个不可修改的区域，用于保存函数参数。在函数内部，您可以使用calldata关键字来访问函数参数，但是不能在函数之外使用它们。在函数调用期间，函数参数的值将从调用方复制到调用数据区域中，并在函数执行完毕后被销毁。

Memory is used to declare that a function parameter will be stored in memory. The data in memory exists only for the duration of the function's execution and is destroyed when execution is complete. You can use the memory keyword to create temporary variables inside a function, but you cannot use them outside the function. During a function call, the values of the function arguments are copied from the caller into memory and destroyed when the function is finished executing.

The calldata is used to declare that the function parameters will be stored in the call data area. The call data area is a non-modifiable area that is used to hold function parameters. You can use the calldata keyword to access function arguments inside a function, but you cannot use them outside the function. During a function call, the values of the function parameters are copied from the caller into the call data area and destroyed when the function is finished executing.

15、UUPS 和 Transparent Upgradeable Proxy 模式之间有什么区别?

What is the difference between UUPS and Transparent Upgradeable Proxy mode?

答：

在Transparent Upgradeable Proxy模式中，代理合约只负责将所有调用转发到实现合约，并将实现合约的地址存储在代理合约的状态变量中。在升级合约时，新的实现合约将被部署到新的地址，然后将新的实现合约的地址存储在代理合约的状态变量中。这种方法的缺点是，每次升级合约时都需要部署一个新的代理合约。

相比之下，UUPS代理模式使用了更加智能的方法。在UUPS代理模式中，代理合约只负责将所有调用转发到实现合约，并将实现合约的地址存储在代理合约的状态变量中。在升级合约时，只需要将新的实现合约的代码上传到现有的代理合约地址，而无需部署新的代理合约。这种方法的优点是，可以在不更改代理合约地址的情况下升级合约，从而避免了每次升级合约时都需要部署一个新的代理合约的问题。

In Transparent Upgradeable Proxy mode, the proxy contract is only responsible for forwarding all calls to the implementation contract and storing the address of the implementation contract in the proxy contract's state variable. At the time of upgrading the contract, the new implementation contract will be deployed to the new address and then the address of the new implementation contract will be stored in the state variable of the proxy contract. The disadvantage of this approach is that a new agent contract needs to be deployed each time the contract is upgraded.

In contrast, the UUPS proxy model uses a smarter approach. In the UUPS proxy model, the proxy contract is only responsible for forwarding all calls to the implementation contract and storing the address of the implementation contract in the proxy contract's state variable. When upgrading a contract, only the code of the new implementation contract needs to be uploaded to the address of the existing proxy contract without deploying a new proxy contract. The advantage of this approach is that the contract can be upgraded without changing the proxy contract address, thus avoiding the problem of deploying a new proxy contract every time the contract is upgraded.