**Advanced Solidity Concepts**

Advanced Solidity Concepts is the third course on cryptozombies.io, and as the name suggests it aims to delve deeper into the language than the previous courses and teach us more blockchain related concepts implemented by Solidity.

Once deployed, contracts in Solidity are immutable and can not be updated. This breaks the normal workflow of other programming languages where you release version 1, hunt for bugs and fix as you go. For this reason security is the highest concern of smart contracts, as once you discover a vulnerability in the contract it becomes unusable.

However, it cannot be guaranteed the bugs won’t be encountered and at some point you will really need to change the address of a contract that you rely on, so that this current contract you defined now will still work if some dependent contract breaks. A very common contract you can inherit to help you in this cause is Ownable. Ownable contains a modifier function onlyOwner which can be used to ensure functions called with that modifier are only called by the owner of the contract. The owner is set when the contract is created and can even be transferred.

Modifiers such as these can give certain users special powers, which if you notice can be used maliciously. If a backdoor is implemented, then the contract can be abused, and it actually no longer is decentralized. For this reason it is important that you read the source code of less trusted (or popular) contracts you use, for you may be subjected to a rug pull.

Another concept that is very important to the Ethereum blockchain is gas. Gas is a fee you have to pay each time you call a method, this way you ensure you won’t clog up the network with precious runtime, as you will be disincentivized to call complex methods due to high fees. Gas also incentivizes programmers to write as efficient code as possible, so that it won’t be costly for users, which is another priority that is higher than in other programming languages (you’re unlikely to want to do weird hacks in python, as the aim of python is to be readable, not the most efficient).

Struct packing is a good way you can save on gas, this is the practice of using the smallest int data types possible since these will be grouped into chunks of 256 bits of memory. So 3 uints will occupy 3\*256 bits of memory, while 3 uint32 will occupy just one 256 bit of memory! (3\*32=96 which is less than 256, but in blockchain the smallest reservable space is 256). Another way to save gas is by using view functions as often as possible. Calling view functions externally doesn’t cost gas since it doesn’t modify the storage, but calling a view function from a non-view function will still cost gas, that computation still has to be done by many nodes.

Since storage is very expensive gas-wise, you might also find yourself wanting to implement what would be considered in other programming languages anti-patterns. You would almost always prefer to rebuild an array in memory instead of using the one in storage for example, even though you waste more time on that, storage is usually more expensive.

Finally this course also teaches up for loops, which are almost identical to those in C. Wrapping up this course with this simple keyword makes for another very well explained and useful course, where we delve deeper and deeper into blockchain concepts and dos and don’ts that every blockchain programmer has to know!