**Strings in Solidity**

1. Any contract can take input of strings.
2. A contract function can output strings—as long as this function is not called in another contract. The function must be used as a helper function. Or it can be called from javascript.

Here are some helper functions:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function stringToBytes32(string memory source) returns (bytes32 result) {

assembly {

result := mload(add(source, 32))

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function bytes32ToString(bytes32 x) constant returns (string typeCasted)

{

bytes memory bytesString = new bytes(32);

uint charCount = 0;

for (uint j = 0; j < 32; j++)

{

byte char = byte(bytes32(uint(x) \* 2 \*\* (8 \* j)));

if (char != 0)

{

bytesString[charCount] = char;

charCount++;

}

}

bytes memory bytesStringTrimmed = new bytes(charCount);

for (j = 0; j < charCount; j++)

{

bytesStringTrimmed[j] = bytesString[j];

}

typeCasted = string(bytesStringTrimmed);

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//helper function for string concatenation

function strConcat(string \_a, string \_b, string \_c) internal returns (string theVal)

{

bytes memory \_ba = bytes(\_a);

bytes memory \_bb = bytes(\_b);

bytes memory \_bc = bytes(\_c);

string memory str = new string(\_ba.length + \_bb.length + \_bc.length);

bytes memory toReturn = bytes(str);

uint k = 0;

for (uint i = 0; i < \_ba.length; i++)

{

toReturn[k++] = \_ba[i];

}

for (i = 0; i < \_bb.length; i++)

{

toReturn[k++] = \_bb[i];

}

for (i = 0; i < \_bc.length; i++)

{

toReturn[k++] = \_bc[i];

}

theVal = string(toReturn);

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_