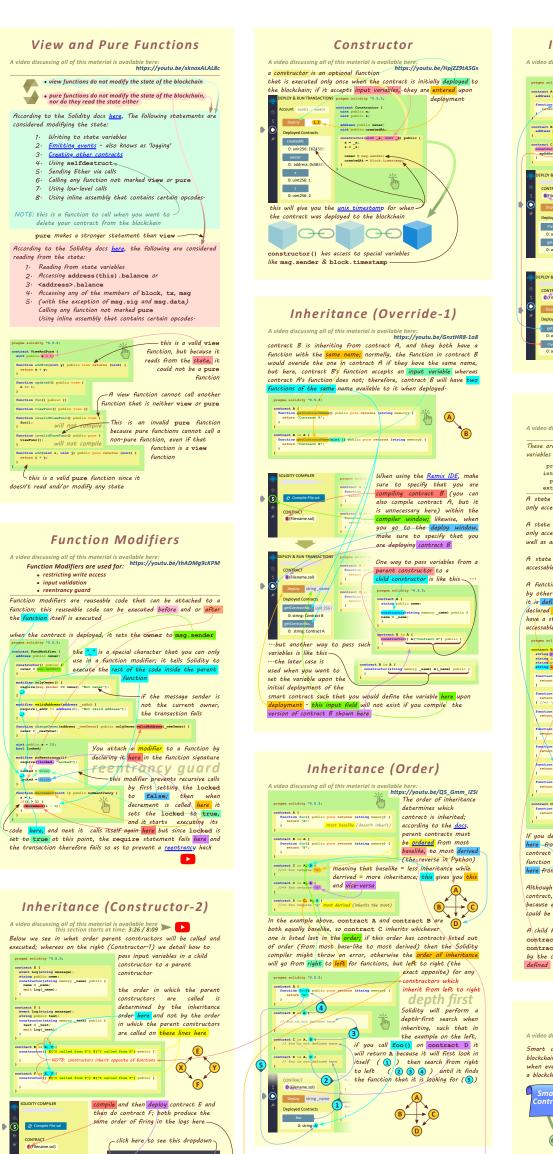


STEP 1: Write a smart contract in this file explorer————

contract SimpleStorage {
 string public text;

function set(string memory \_text) pub text = \_text;

variables; we do this by using the



View and Pure Functions

view functions do not modify the state of the blockchain

pure functions do not modify the state of the blockchain, nor do they read the state either

2. Emitting events - also knows as 'logging'
3. Creating other contracts
4. Using selfdestructs
5. Sending Ether via calls
6. Calling any function not marked view or pure
7. Using low-level calls
8. Using inline assembly that contains cartain accords

8. Using inline assembly that contains certain opcodes.

IOTE: this is a function to call when you want to -

1. Reading from state variables
2. Accessing address (this).balance or

Calling any function not marked pure

this is a valid pure function since it

ne function itself is executed

Function Modifiers

function change/yesr(eddress newOwner) public onlyOwner validaddress (newOwner) ( owder = newOyner;

video discussing all of this material is available here this section starts at time: 3:26 / 8:09

text

O: string: Y was ca

constructor to a parent

are called on these lines here

same order of firing in the logs here-

Inheritance (Super)

this section starts at time: 8:42 / 12:37

This example on the left shows how the parent (more baselike) functions will be called when using the keyword super; contracts A, B, & C first emit an event then

call the parent contract by using the keyword super

contract is called anjecticy wherever here supper is used to call all parent contracts in the order of inheritance because the bar() function in contract A is being overidden by the bar () functions in contracts B and C

here, X, Y, & Z are hypothetical

-click here to see this dropdo

pure makes a stronger statement than view —

<address>.balance
Accessing any of the members of block, tx, msg

Using inline assembly that contains certain opcodes.

5. (with the exception of msg.sig and msg.data)

considered modifying the state:

reading from the state:

allowed and "[ ]" is not reccomended.

more reliable (as an input variable) and avoid the

we will also get compile errors

problem with gas (discussed above) is to put an upper-bound to the array size that will in turn limit the amount

We can call firstFunc and

then call secondFunc seperately, or we can just call thirdFunc instead

Here we are assigning the outputs of the

constructors of multiple parent
contracts ··· this is another way

to call the parent constructor inside the constructor of the child contract; here we do not put commas between contracts when listing them

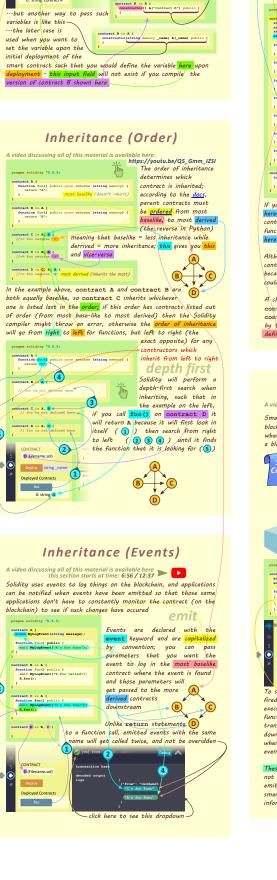
here the constructor is accepting two

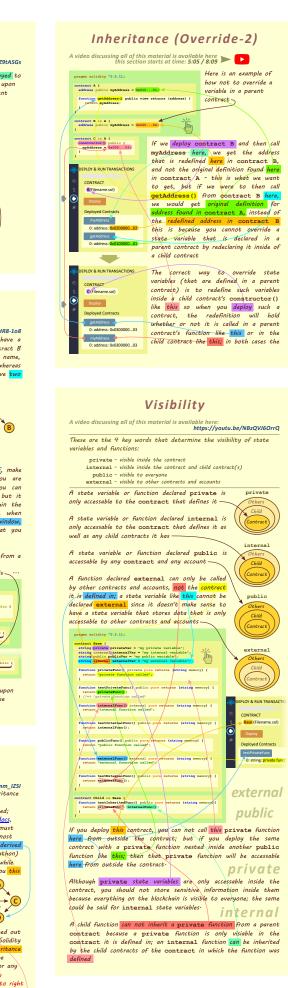
compile and then deploy contract D, and

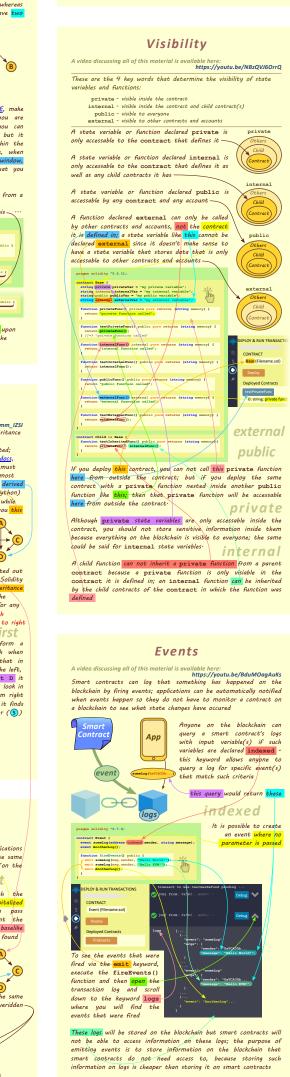
then upon deployment, set the string\_name to "foo" and the string\_text to "bar"

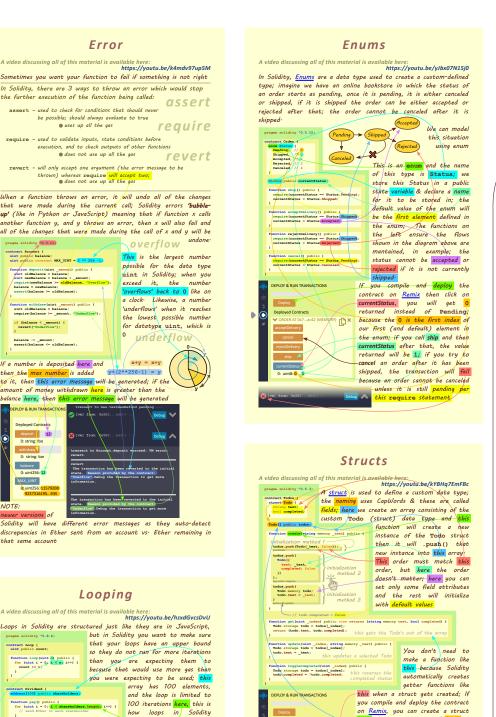
Calling the state variable name returns the string "£00" and calling the state variable text returns the string "bax"

· Writing to state variables









address payable public owner;

owner

Sending Ether

There are 3 methods to send Ether from one contract to another

transfer - forwards no more than 2300 gas, throws error

call - forwards all gas or sets gas, returns boolean value recommended way after December 2019 because in the future, the cost of gas can change for different operations

NOTE: newer versions of Solidity require the use of the keywork

where this refers to this

These values are

Whenever a function receives Ether from a contract using any

-fallback in lieu of function per the latest docs

nction getBalance() public view return address(this).balance;

unction sendVisSend(address payable bool sent = \_to.send(msg.value); require(sent, "Failed to send Ether

unction sendViaCall(address payable \_to) public payable (
(bool sent, bytes memory data) = \_to.cali.value(msg.value)("")
require(sent, "Failed to send Ether");

unt OxAbs. 35cb/(100 ether) here and then click here, if you will see a

in the code above) without specifying an amount here and the

yntax like this <mark>here but note that setting dynamic values in th</mark>i

way does not work as this sendViaHardCodedCall() function will fail because this does not work; most functions use msg.value as a

alue since the fallback cannot return anything (i·e· 0 bytes)

Fallback Function

Fallback functions like this have no names, inputs, or outputs and

7. When you call a function

2. When you send Ether to

If you copy the address of the Fallback function here and the

paste it <mark>here then set the transfer amount to 1 Ether here and then</mark>

click here, you will see that in th

transaction log that 2255 gas were recorded here; calling this function was the transfer method to send there to the Fallback function and this in turn, triggeted the fallback function inside the Fallback contract

which logged the gas left here tha

the send, transfer, or call methods

that does not exist in

Newer versions of Solidity (colors psychia to) public payable to Solidity (color seat, byte memory data) - to call (galors 1/4"); require (seat, 'yelled to seed Ether'); with the call medit

matter of convention

here, but on Remix, the first time you try this you will have to specify a value here or an error message will occur, after the first time however, you can call any of the world and the standard was value, channed to 7

Error

the further execution of the function being called:

assert - used to check for conditions that should never

function withdraw(uint \_amount) public {
 uint oldmalance = balance;
 require(balance >= \_amount, "Underflow");

if (balance < \_amount) {
 revert("Underflow");</pre>

balance -- \_amount; assert(balance <- oldBalance);

that same account

Looping

open like this where you can have unlimited iterations

return myArray length;

assert(myArray.length = assert(myArray[0] == 1) assert(myArray[1] == 4) assert(myArray[2] == 3)

contract Mapping (
mapping (address -> uint) public myMap;

function remove(address \_add delete myMap[\_addr];

0x7E...da42

Arrays

arrays are declared with type, accessability, and name; they can be either dynamically sized like this or fixed sized like this; the actual

, size = 10) once the size of the array has been declared, i

opposed to leaving them

In Solidity, there are three

built-in functionalities:

push () is a function that

adds an element to the

end of an array, pop () is

Solidity does not have an unshift()

needed to delete an element at a specified index, but this would set the element's value to 0, the keep arrays

compact, we can grab the <mark>l</mark>ast element in an array and set it to the index we want deleted, then delete the <mark>last</mark> element after that

Mapping

ing all of this material is available here: https://youtu.be/t03vVMCOts8

in Solidity, you cannot iterate through a mapping and

in order to create a mapping that you could interate on,

you would have to use a

combination of mapping and arrays wherein you store the keys in an array; you also cannot get the size of a mapping; the format is:

or a mapping temetormae is:

mapping (keytype, Vouluetype)

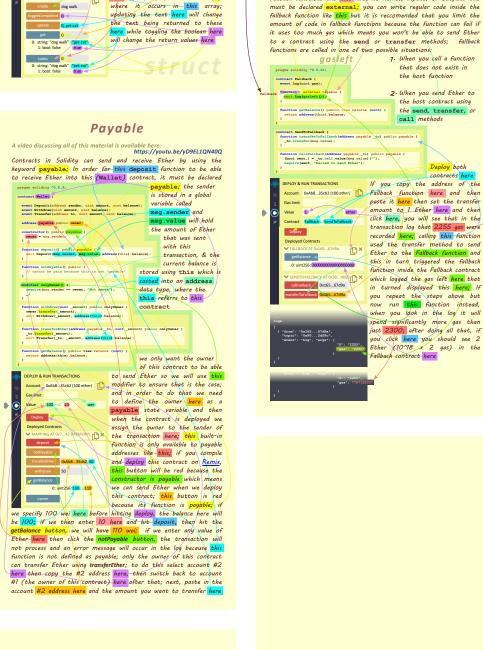
the Keytype for a mapping has to
be a built-in value type, a string,
or bytes; the valuetype can be any
type including an array or another
mapping! Unlike arrays, mappings
will will not return arrors when you
about visit incheal the desult value of 0

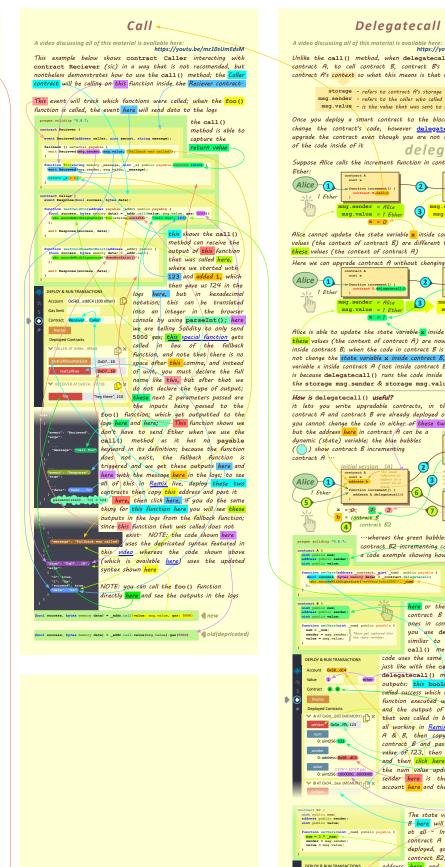
to access keys that do not exist, instead the default value of 0 will be returned; as with arrays, the delete keyword does not will be returned; as with arrays, the delete keyword aces not remove an element in a mapping, but instead sets its value to 0; To see all this, compile and deploy this contract on Remix and then copy the contract address here and paste it here with a unit value here, then hit the set button. After that, hit the set button and you will see this wint value here, but if you copy the address and

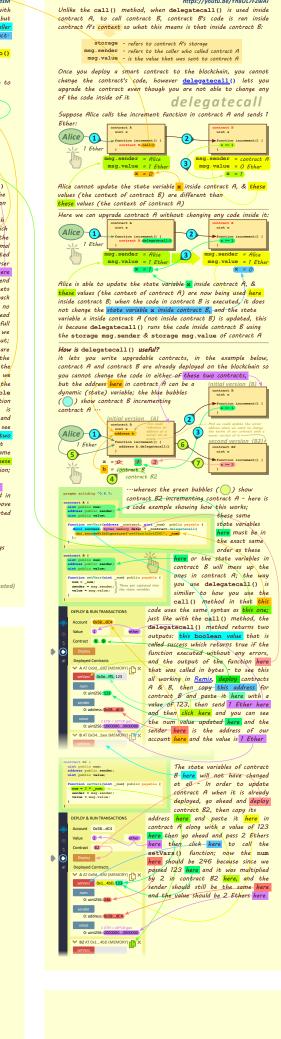
paste it here, then hit the remove button after that, when you hit the get button the value will be 0 instead of 123

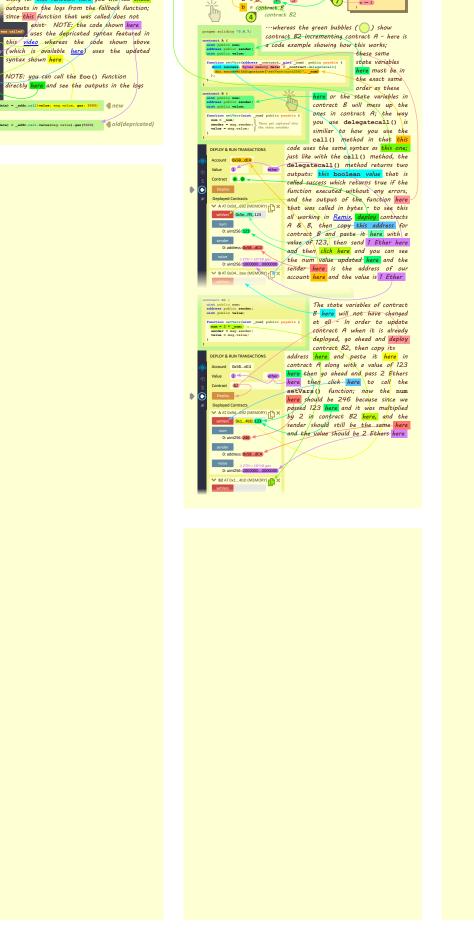
or splice() method like

uses up all the gas











github.com/Richard-Burd/solidity-sandbox last updated @11:25am on 18/August/2021 by Richard Burd rick.a.burd@gmail.com

