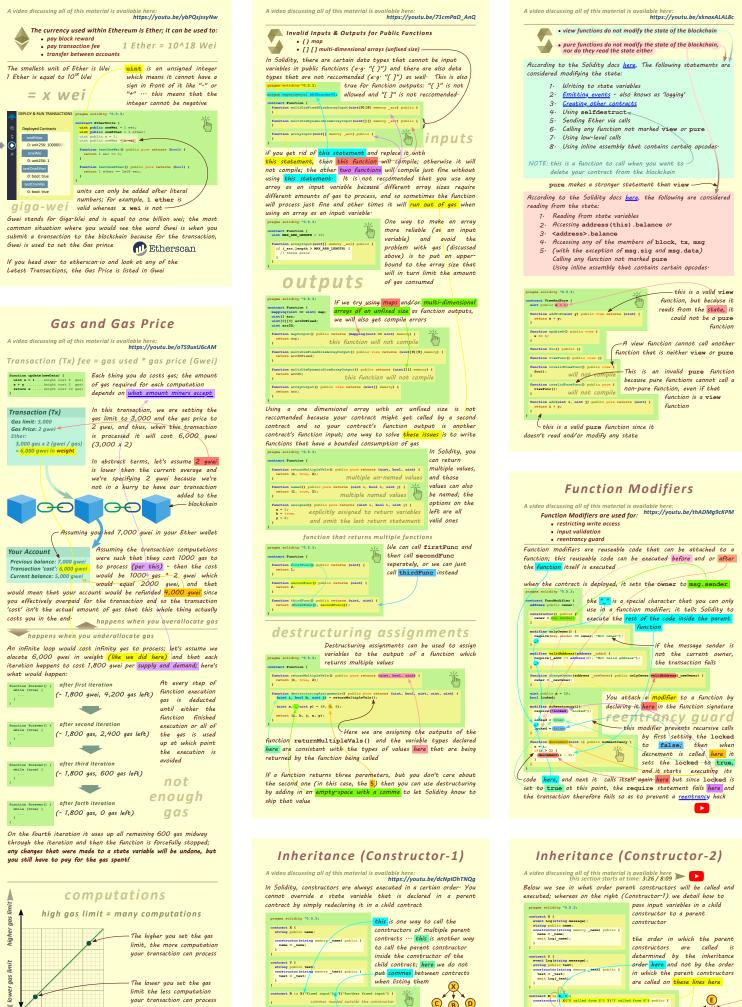


in this file explorer —

contract SimpleStorage {
 string public text;

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

variables; we do this by using the



invocuments

Comparison

Compa

here the constructor is accepting two

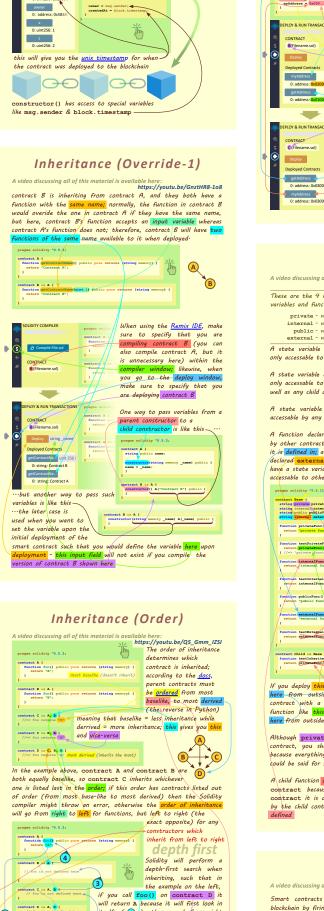
compile and then deploy contract D, and

then upon deployment, set the string_name to "foo" and the string_text to "bar"

— Calling the state variable name returns
the string "foo" and calling the state
variable text returns the string "bax"

Invalid Functions

View and Pure Functions



Constructor

address public owner; sint public createdAt;

onstructor is an optional function

createdAt

owner owner

Deployed Contracts
foo
0: string: A

same order of firing in the logs here-

-click here to see this dropdo

Two ways to call parent contracts

Inheritance (Super)

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

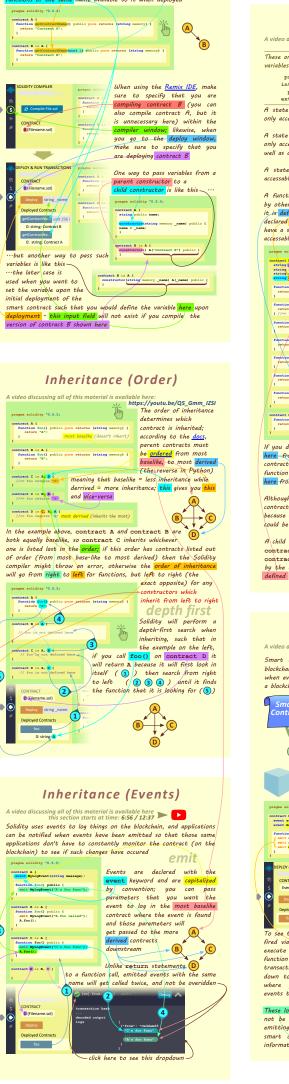
This example on the left shows how the parent (more baselike) functions will be called when using the keyword super; contracts A, B, & C first emit an event then

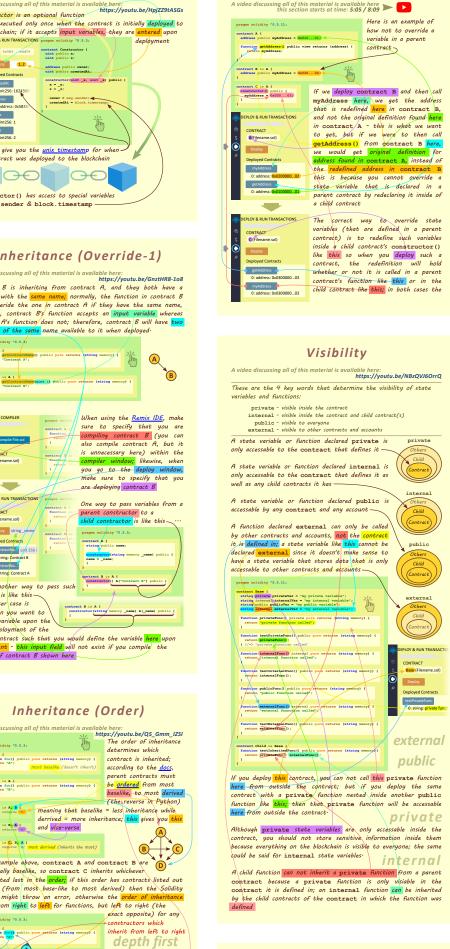
the keyword super

here, super 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

text 0: string: Y was o





Inheritance (Override-2)

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

Loops in Solidity are structured just like they are in JavaScript

ion pay() public (
(uint i = 0; (i < third black line) i++) (
and Ether to such sharsholder

100 iterations here, this i.

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

and the number of elements in it ca

Solidity does not have an unshift()

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

you would have to use a

Mapping

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

In Solidity, there are three

built-in functionalities:

push () is a function that

adds an element to the

end of an array, pop () is

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 _addr) delete myMap[_addr];

0x7E...da42

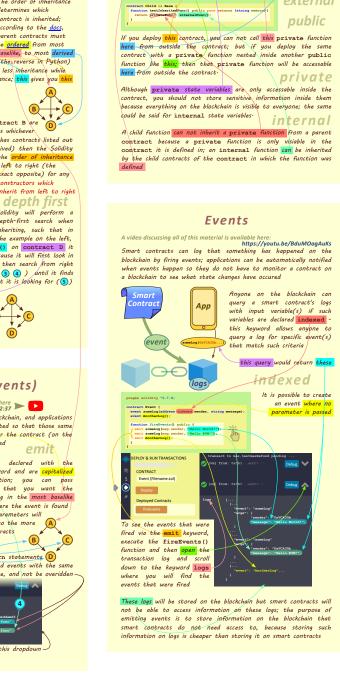
opposed to leaving them

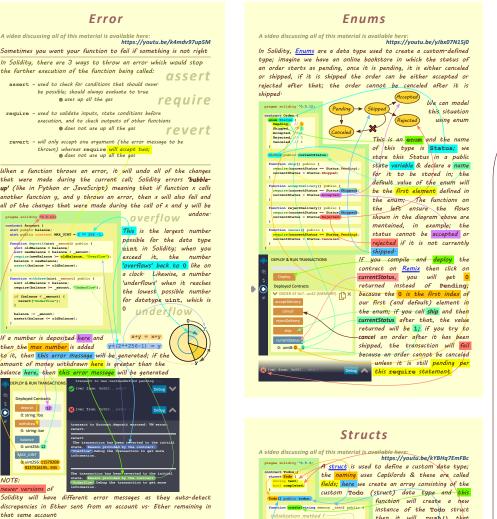
uses up all the gas

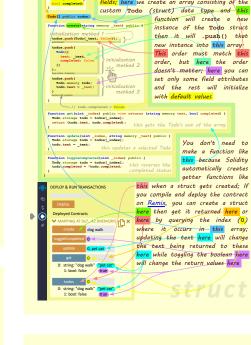
require — used to validate inputs, state conditions before execution, and to check outputs of other functions

thrown) whereas require will accept two;

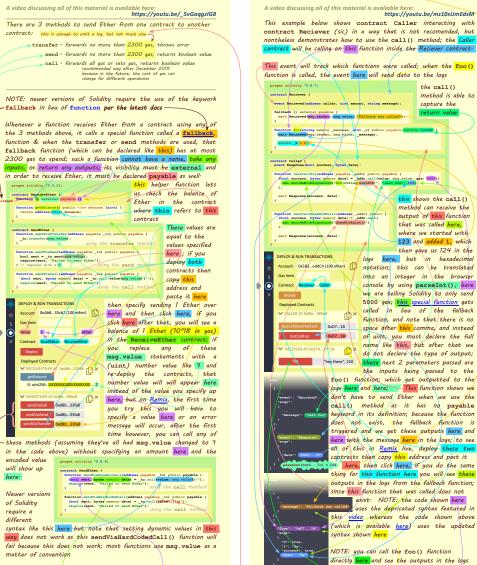
does not use up all the gas











Call -

Sending Ether

transfer - forwards no more than 2300 gas, throws error

-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");

Newer versions of Solidity

function send/salter codedCall (eddress payabla to) public payable (tool sent, byte) smooty data) - to(sall(yabus),142); require (sent, yailed to send they?); using the call me

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

Fallback Function

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

must be declared external; you can write regular code inside the fallback function like this but it is reccomended that you limit the amount of code in fallback functions because the function can fail if

uses too much gas which means you won't be able to send Ether a contract using the send or transfer methods; fallback

7. When you call a function

that does not exist in

2. When you send Ether to

the send, transfer, or call methods

1 Ether here and then

click here, you will see that in th

cauch mare, you will see that in the transaction log that 2255 gas were recorded there; calling this function used the transfer method to send there to the Fallback function and this in turn triggered the fallback districts in incidents callback extracts.

nich logged the gas left here that

300; after doing all that, if

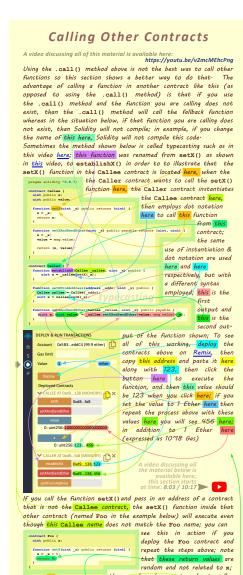
k here you should see 2

matter of convention

enit Log(gasleft());

unction getBalance() public v return address(this).balance

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



they are here because in order to call a function in another contract, the function being called must have a return value or it will not work; this setX()

function located inside this

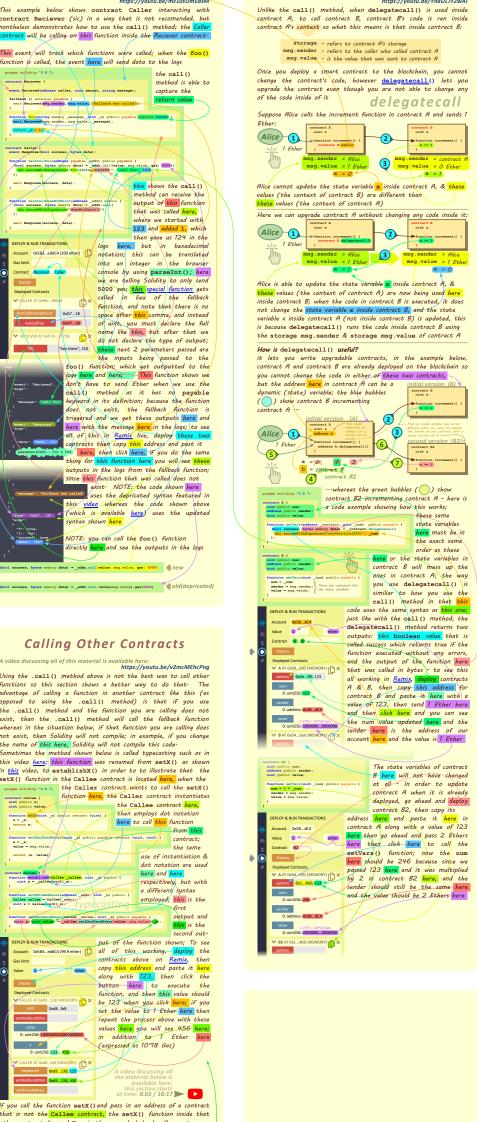
Bar contract for example,

cannot be called by the

Caller function above (if

you repeat the steps in

section) because there is no return value located inside the Bar contract's setX() function



Delegatecall



last updated @1:40pm on 24/August/2021 by Richard Burd rick.a.burd@gmail.com Solidity Illustrated