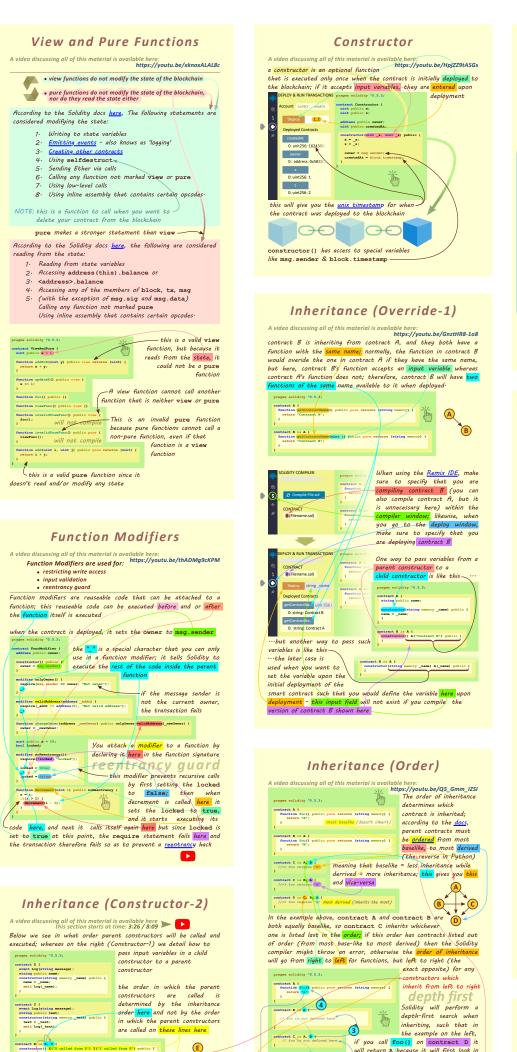


STEP 1: Write a smart contract in this file explorer————

contract SimpleStorage {
 string public text;

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

variables; we do this by using the



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

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

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

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

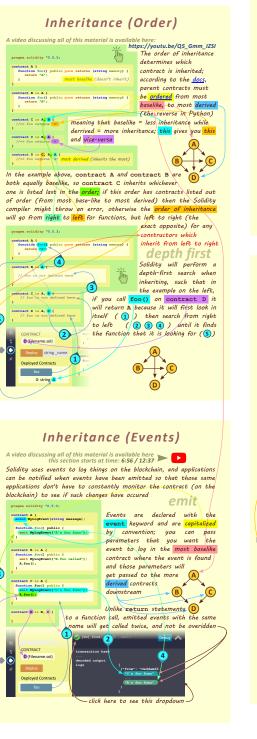
-click here to see this dropdo

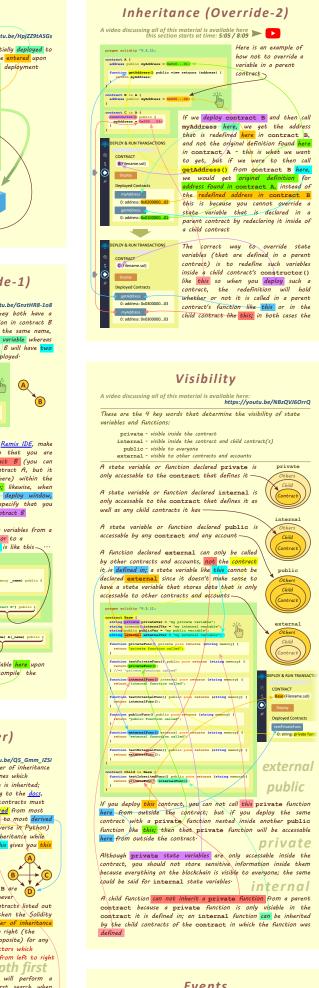
pure makes a stronger statement than view —

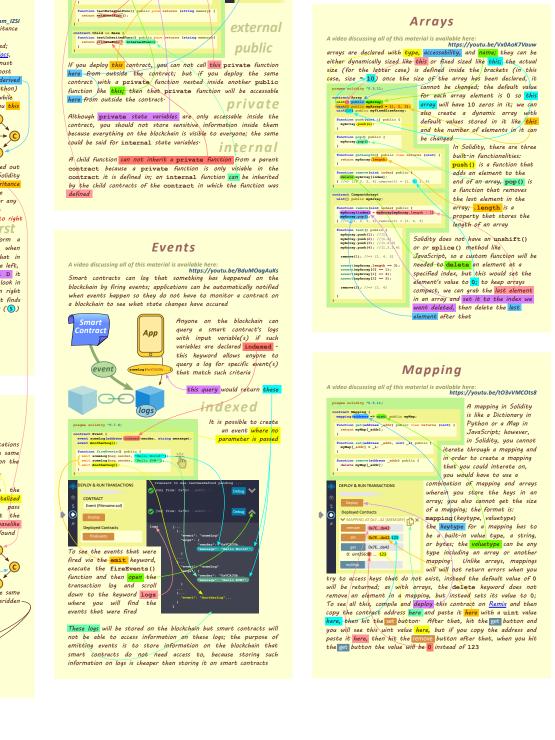
<address>.balance
Accessing any of the members of block, tx, msg

Using inline assembly that contains certain opcodes.

5. (with the exception of msq.siq and msq.data)







Enums Sometimes you want your function to fail if something is not right In Solidity, Enums are a data type used to create a custom-defined type; imagine we have an online bookstore in which the status of an order starts as pending, once it is pending, it is either cancele or shipped, if it is shipped the order can be either accepted Pending Shipped Rejected using enum Rejected This is an appearance of the second seco • does not use up all the gas rever This is an enum and the name of this type is Status; w revert - will only accept one argument (the error message to be function ship() public (require (currentStatus -- Status.Pending); currentStatus -- Status.Shipped; When a function throws an error, it will undo all of the changes at were made during the current call; Solidity errors **'bubble** default value of the enum w definition acceptibilities of public incorporation acceptibilities of Status (Mapped): Description of Status (Mapped): Description of Status (Mapped): The functions on up' (like in Python or JavaScript) meaning that if function x cal another function y, and y throws an error, then x will also fail and all of the changes that were made during the call of x and y will be The second ly palie | markatined, in example; the markatined of the second or rejected or rejected or shipments are second or rejected fit is not currently shipped. Shipped: According () Sylve balance () This is the largest number possible for the data type of the largest possible for the largest 'underflows' when it reaches he lowest possible numbe for datatype uint, which is the enum; if you call ship and the currentStatus after that, the val Structs T<u>struct</u> is used to define a custom data type; the <mark>naming</mark> uses CapWords & these are called <mark>Telds; here</mark> we create an array consisting of the The we create an array consisting of the (struct) data type—and this function will create a new instance of the Todo struct then it will push () that new instance into this array This order must match this order, but here the order doesn't-matter; here you can set only some field attributes discrepancies in Ether sent from an account vs· Ether remaining in tion create string m

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 < *harebolders.length) i++) (
and Ether to such sharebolder

100 iterations | here, this i.

open like this where you can have unlimited iterations

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

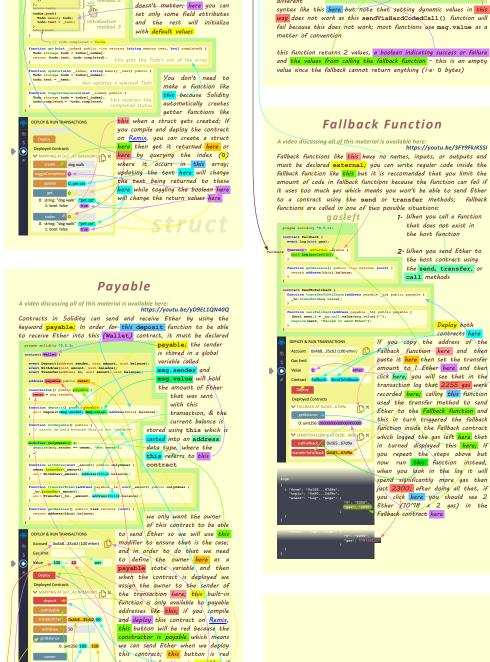
thrown) whereas require will accept two;

does not use up all the gas

Here is an example of how not to override a

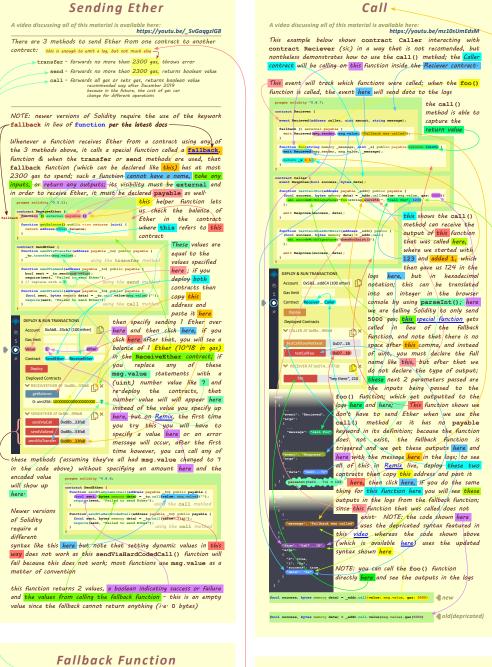
ent contract by redeclaring it inside of

vailable here: https://youtu.be/NBzQVJ6OrrQ



not process and an error message will occur in the log because this function is not defined as payable; only the owner of this contract can transfer Ether using transferther; to de this select account #2 here then copy the #2 address here, then switch back to account #1 (the owner of this contract) here after that; next, paste in the

account #2 address here and the amount you want to transfer he



Sending Ether

-fallback in lieu of function per the latest docs

tract ReceiveEther (

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");

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>

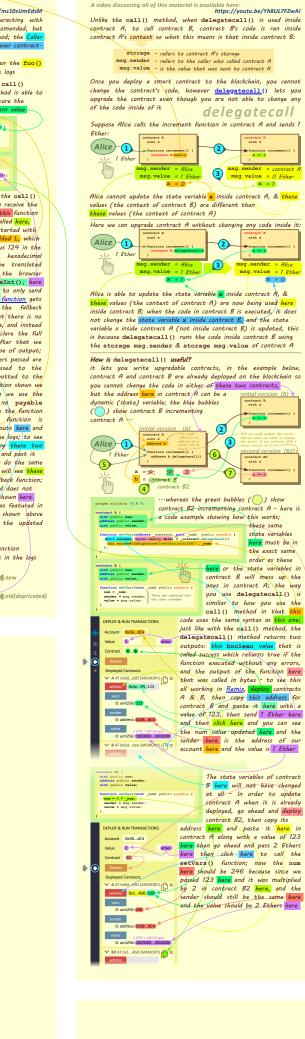
click here, you will see that in th

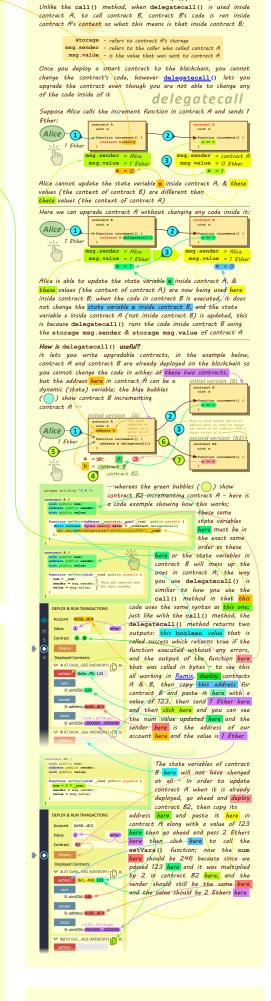
transaction log that 2255 gas were recorded here; calling this function used the transfer method to send this function and this in turn triggered the fallback function inside the Fallback contract

which logged the gas left here tha

the send, transfer, or call methods

that does not exist in





Delegatecall



github.com/Richard-Burd/solidity-sandbox last updated @9:33am on 22/August/2021 by Richard Burd rick.a.burd@gmail.com

