

integer cannot be negative

gas limit to 3,000 and the gas price to

execution or all of

up at which point

not

enough

your transaction can process

— The lower you set the gas limit the less comp

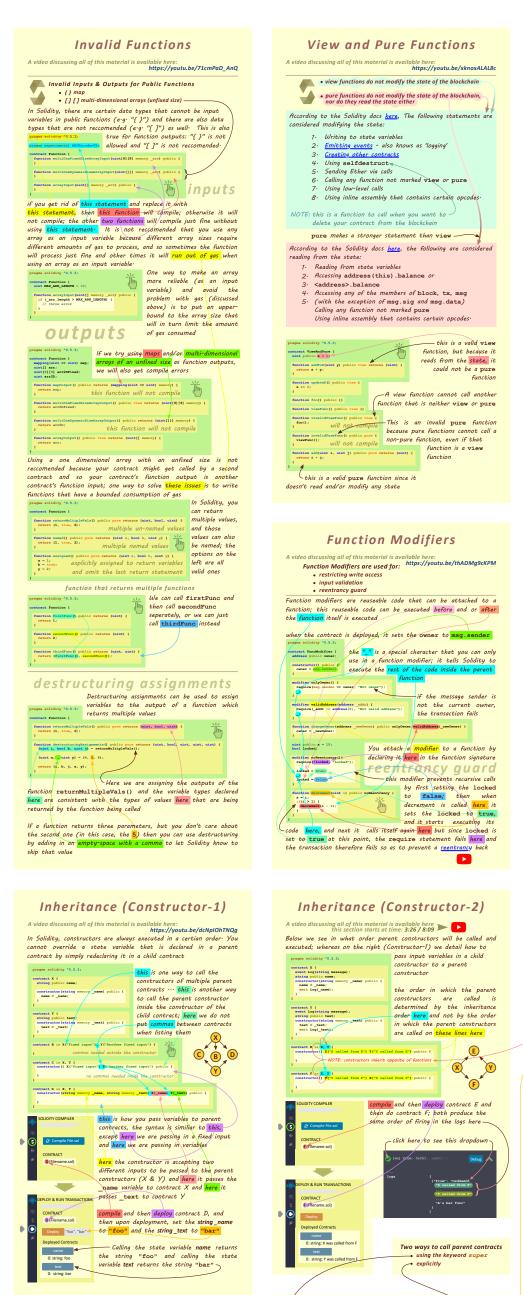
set, the more Ether you

your transaction will be processed faster

will have to spend, bu

After your transaction is sent, and included in a block, your account will

be refunded for any unspent gas

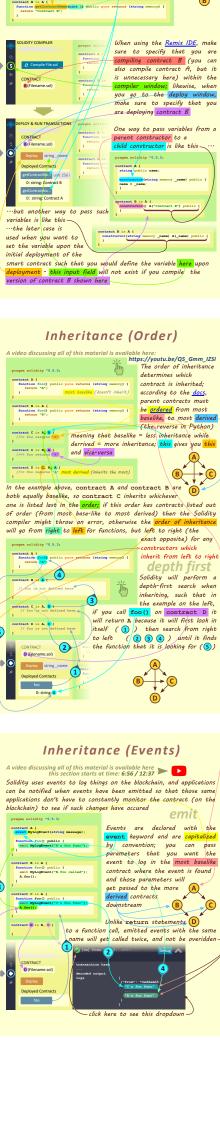


Inheritance (Super)

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

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



Constructor

blockchain; if it accepts input variables, they are entere

owner = msg.sender; createdAt = block.timestamp;

constructor is an optional function

the contract was deployed to the blockchain

ctor() has access to special variable: like msg.sender & block.timestamp -----

Inheritance (Override-1)

contract B is inheriting from contract A, and they both have a

function with the same name; normally, the function in contract 8 would overide the one in contract A if they have the same name, but here, contract B's function accepts an input variable whereas contract A's function/does not; therefore, contract B will have two

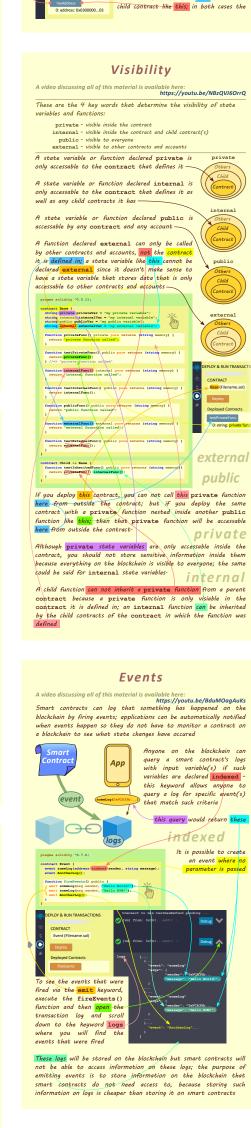
ion (estions/contractions)) public pure returns (string memory)

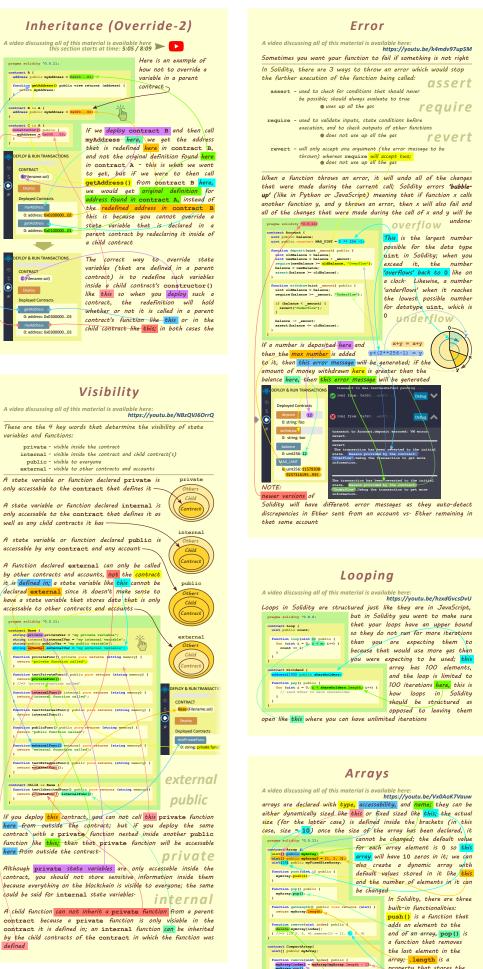
of the same name available to it when deployed

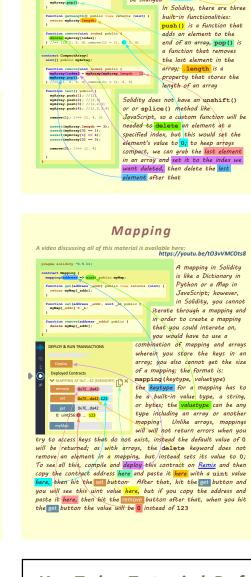
createdAt

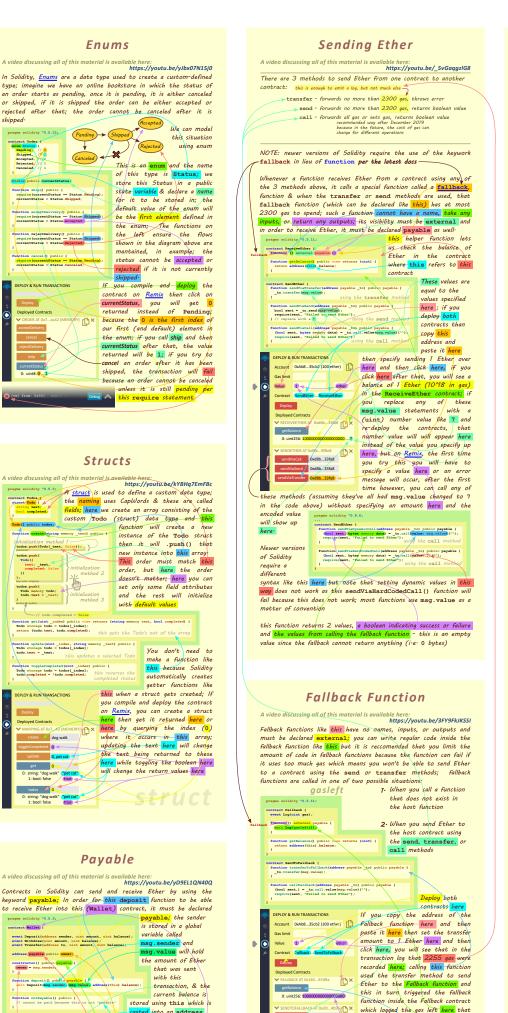
owner owner

0: uint256: 1









Enums

Structs

nction create string

function toggleCompleted(uint \_index)
Todo storage todo = todos[\_index];
todo.completed = 'todo.completed;

create dog walk

0: string: "dog walk" "pet cat"
1: bool: false true

0: string: "dog walk" "pet cat"
1: bool: false true

address payable public owner;

unction notPayable() public (
// cannot be paid because this is not "payab

ndifier onlyOwner() 4(
require(msg.sender -- owner, "Not owner");

unction withdraw(uint\_amount) public onlycomer (
owner\_transfer(amount);
out Withdraw(amount, address(this).belance);

we only want the owner of this contract to be able

payable state variable and the when the contract is deployed we

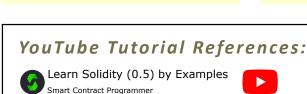
the transaction here; this built-in

function is only available to payable addresses like this; if you compile and deploy this contract on Remix,

constructor is paganie with means of the who we can send Ether when we deploy this contract; this button is red because its function is payable; if we specify 100 wei here before hitting deploy, the balance here will be 100; if we than enter 10 here and hit deposit, then hit the

getBalance button, we will have 110 wei; if we enter any value of Ether here then click the <mark>notPayable button,</mark> the transaction will

terner there then click the notroyable buccon, the transaction will 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 do 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 here



Smart Contract Programmer

github.com/Richard-Burd/solidity-sandbox last updated @12:58am on 12/August/2021 by Richard Burd rick.a.burd@gmail.com



turned displayed this here;

2300; after doing all that, if