Solidity Contracts :

- Contract Layout

raj@acloudfan.com

@acloudfan

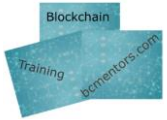
http://ACloudFan.com

Discount Coupon Links to UDEMY courses:

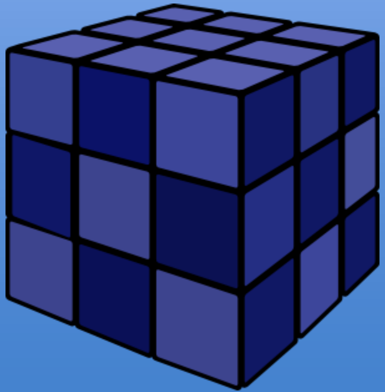
https://www.udemy.com/hyperledger/?couponCode=DKHLF1099

https://www.udemy.com/ethereum-dapp/?couponCode=DKETH1099

https://www.udemy.com/rest-api/?couponCode=DKRST1099



mentoring, seeking Blockchain part time work, project guidance, advice … …
http://www.bcmentors.com



This deck is part of a online course on "Ethereum: Design and Development of Decentralized Apps.

http://solidity.readthedocs.io/en/develop/index.html

OLIDITY

- Statically typed language
- Similar to object oriented languages

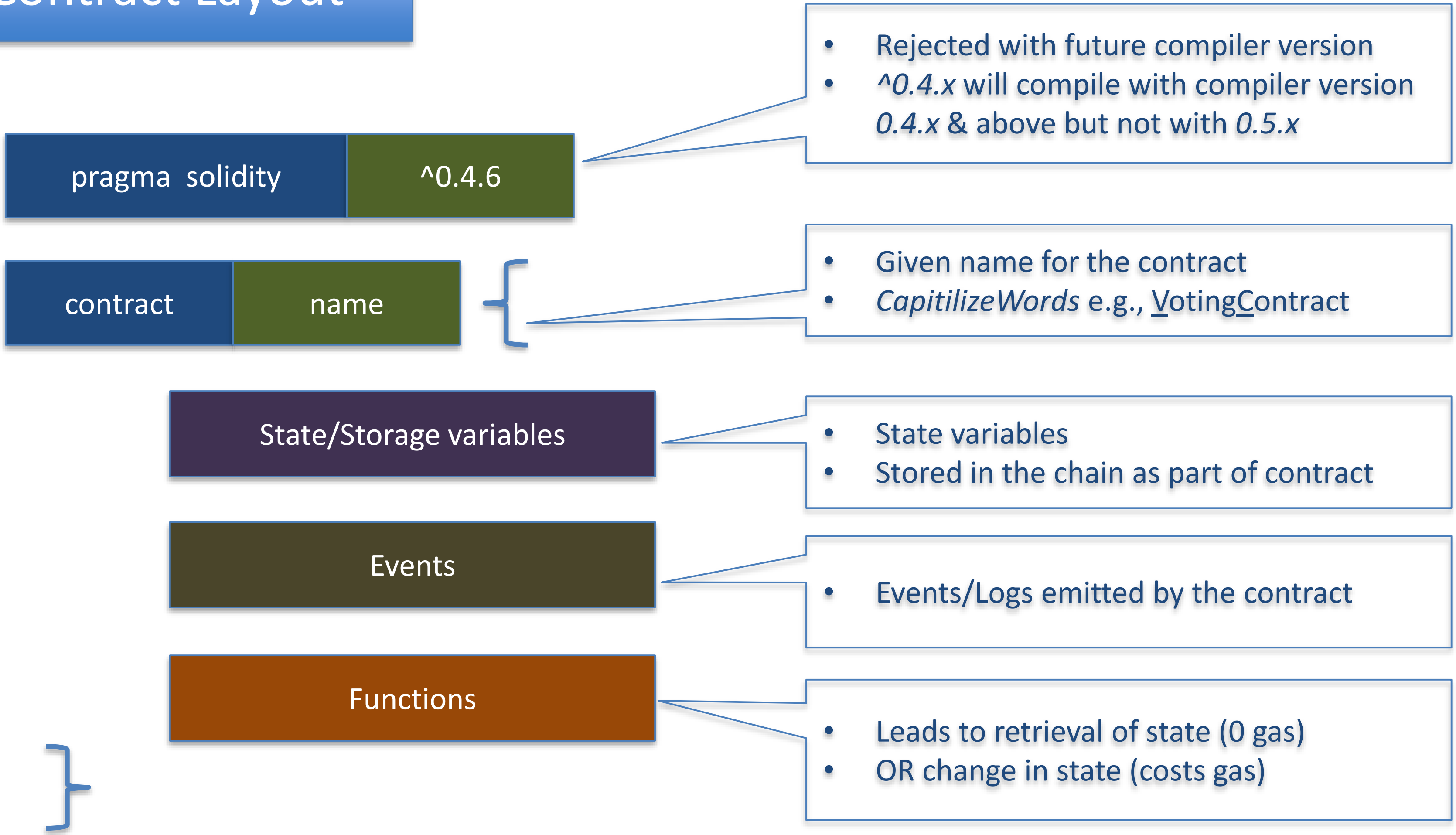
Java™

Contract = Class

Object Instance = Deployed contract on EVM

Many differences between JAVA & Solidity e.g., multiple inheritance, no overloading

Contract Layout



pragma solidity ^0.4.6

- Rejected with future compiler version
- *^0.4.x* will compile with compiler version *0.4.x* & above but not with *0.5.x*

contract name {

- Given name for the contract
- *CapitilizeWords* e.g., VotingContract

State/Storage variables

- State variables
- Stored in the chain as part of contract

Events

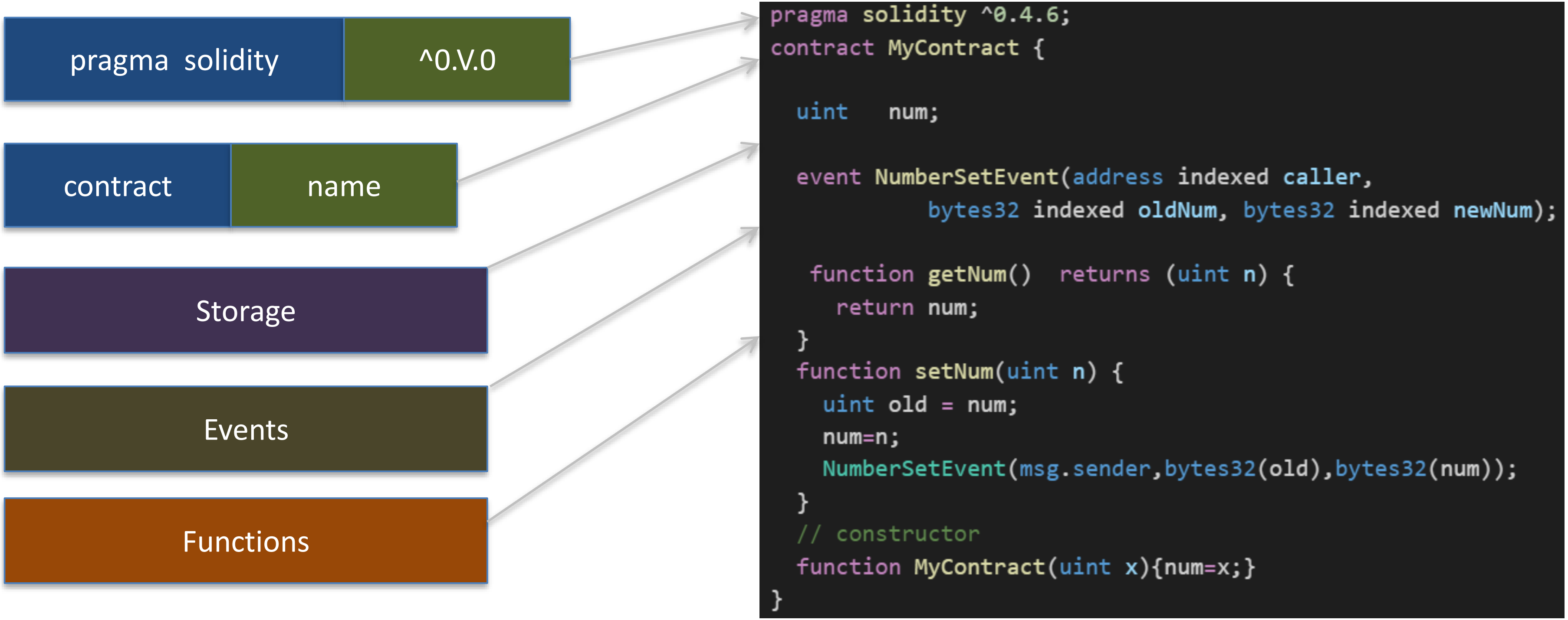
- Events/Logs emitted by the contract

Functions

- Leads to retrieval of state (0 gas)
- OR change in state (costs gas)

}

Walkthrough



pragma solidity | ^0.V.0

contract | name

Storage

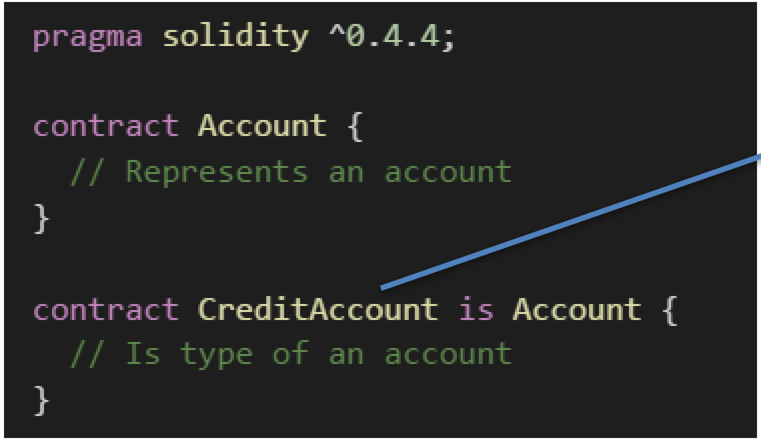
Events

Functions

```
pragma solidity ^0.4.6;
contract MyContract {

  uint   num;

  event NumberSetEvent(address indexed caller,
             bytes32 indexed oldNum, bytes32 indexed newNum);

   function getNum()  returns (uint n) {
     return num;
  }
  function setNum(uint n) {
    uint old = num;
    num=n;
    NumberSetEvent(msg.sender,bytes32(old),bytes32(num));
  }
  // constructor
  function MyContract(uint x){num=x;}
}
```

Multiple Contracts

- Source files can contain multiple contracts
  - Invocation
  - Inheritance
  - Creation

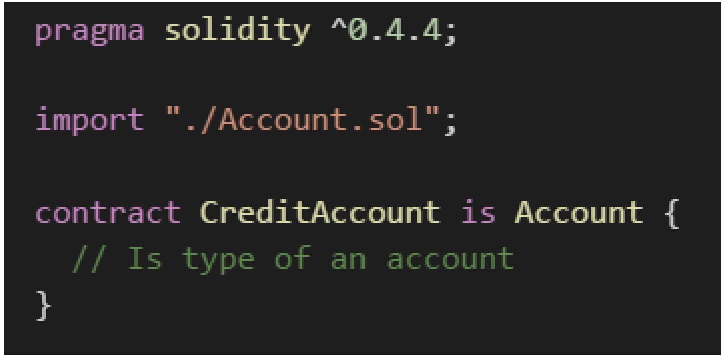


```
pragma solidity ^0.4.4;

contract Account {
  // Represents an account
}

contract CreditAccount is Account {
  // Is type of an account
}
```

Last contract in file gets deployed

Import Statement

- Allows contracts code to be managed across multiple files



```
pragma solidity ^0.4.4;

import "./Account.sol";

contract CreditAccount is Account {
  // Is type of an account
}
```

- Direct import possible over

  HTTP, Github

- Support depends on compiler

Solidity Contracts :

- Basic Types

raj@acloudfan.com

@acloudfan

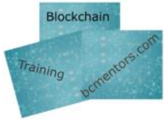
http://ACloudFan.com

Discount Coupon Links to UDEMY courses:

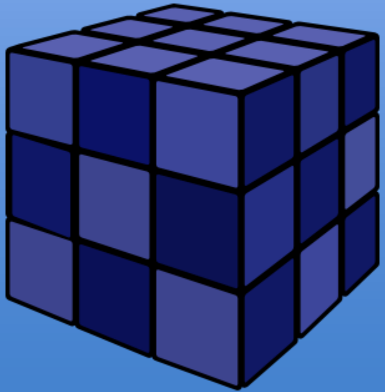
https://www.udemy.com/hyperledger/?couponCode=DKHLF1099

https://www.udemy.com/ethereum-dapp/?couponCode=DKETH1099

https://www.udemy.com/rest-api/?couponCode=DKRST1099



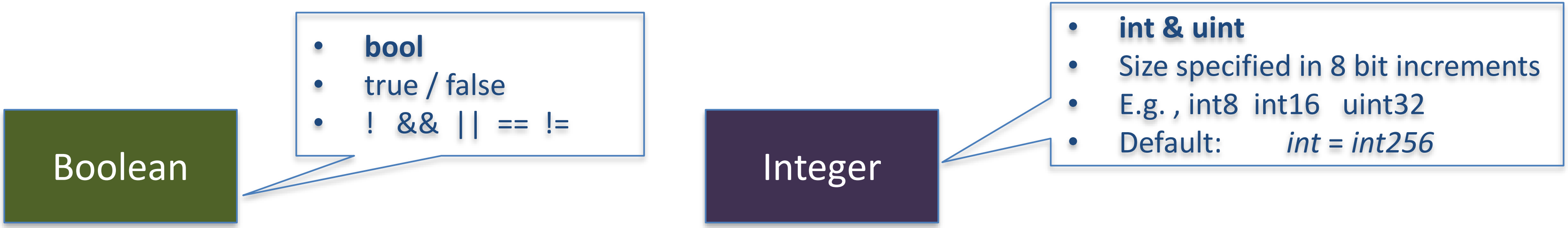
mentoring, seeking Blockchain part time work, project guidance, advice … …
http://www.bcmentors.com



This deck is part of a online course on "Ethereum: Design and Development of Decentralized Apps.

Boolean & Number

- Value types = Always passed by value

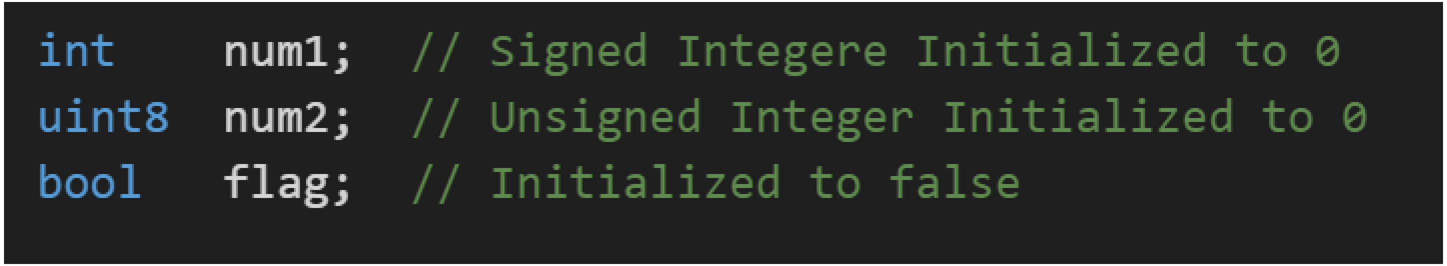


Boolean

- **bool**
- true / false
- ! && || == !=

Integer

- **int & uint**
- Size specified in 8 bit increments
- E.g. , int8 int16   uint32
- Default: *int = int256*



```
int     num1;  // Signed Integere Initialized to 0
uint8   num2;  // Unsigned Integer Initialized to 0
bool    flag;  // Initialized to false
```

Address

- Represents the 20 byte *Ethereum* address
- Value Type

balance

- address.balance Returns balance in *wei*

transfer( ) send( )

- address.transfer(10) Sends 10 Wei from to the *address*

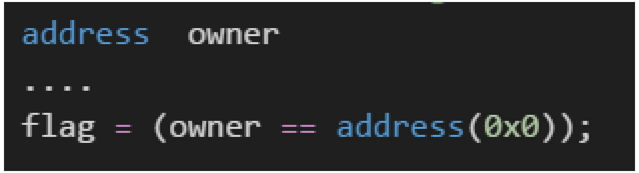
Variable Initialization

CheckNulls.sol

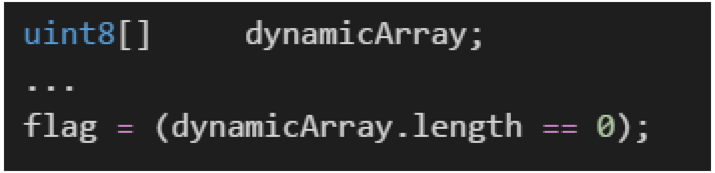
- An un-initialized variable is set to 0s

- NO special keyword to check for validity of variable
  - null/undefined NOT valid in Solidity

- Check for 0 values depend on type of data



```
address  owner
....
flag = (owner == address(0x0));
```



```
uint8[]     dynamicArray;
...
flag = (dynamicArray.length == 0);
```

Type Conversions

Implicit

- Compiler allows if no loss of information
-
```
If (1) { /** code **/}
```

Explicit

- Potential loss of information

```
uint32   x32 = 20;                uint24    x24 = x32;

                                  uint24    x24 = uint24(x32);
```

Deduction

- Compiler can automatically infers type

```
var    someVar = x32;
```

Solidity Contracts :

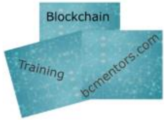
- Memory Management

Discount Coupon Links to UDEMY courses:

https://www.udemy.com/hyperledger/?couponCode=DKHLF1099

https://www.udemy.com/ethereum-dapp/?couponCode=DKETH1099

https://www.udemy.com/rest-api/?couponCode=DKRST1099



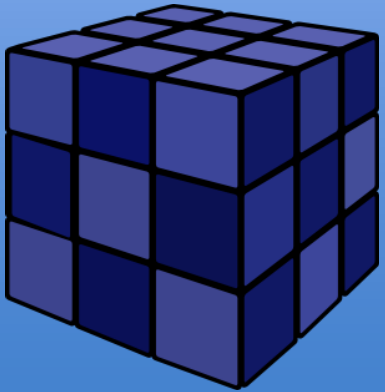
mentoring, seeking Blockchain part time work, project guidance, advice … …
http://www.bcmentors.com

This deck is part of a online course on “Ethereum: Design and Development of Decentralized Apps.

raj@acloudfan.com

@acloudfan

http://ACloudFan.com

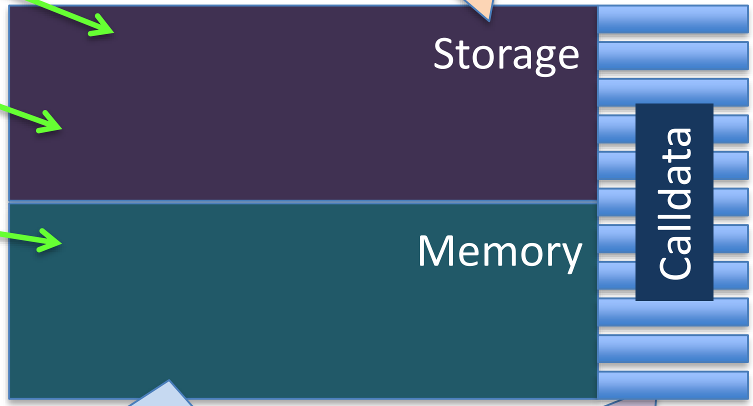


Data Location

- Default: State variables

- Default: Local variable

- Function(args)

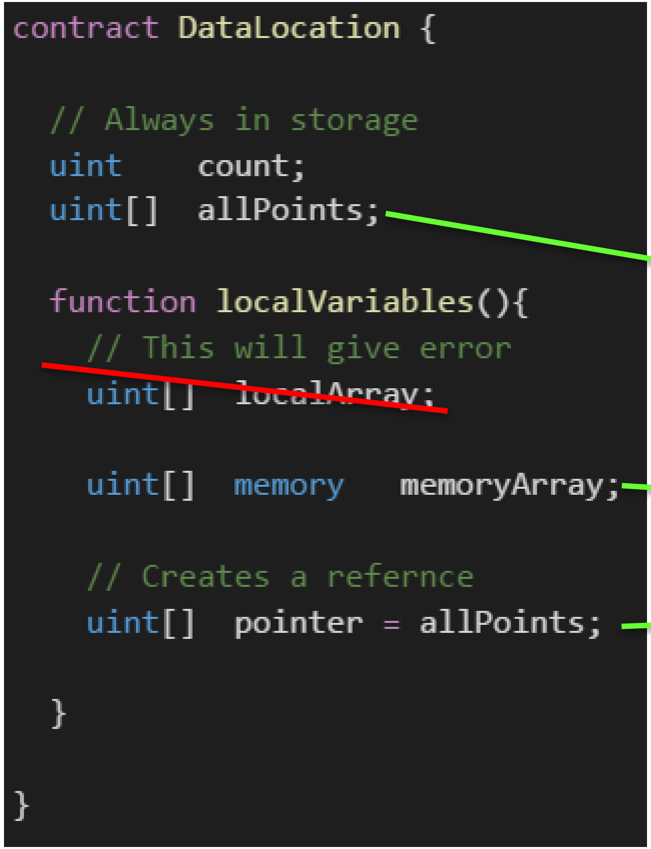


- **Persistence** *(it's a database)*
- Key-Value Store (256 bit key & value)
- Read/write are costly
- Contract can manage only its own

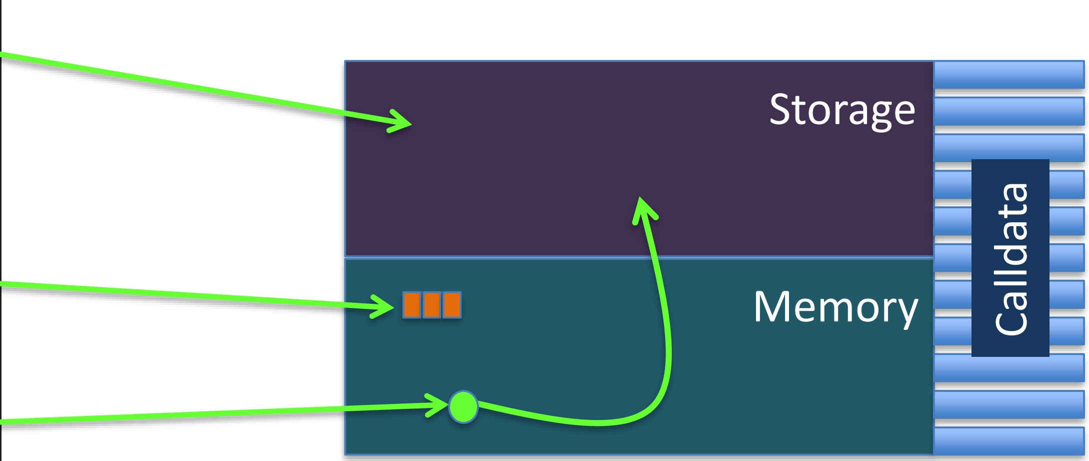
- **Temporary**
- Arrays & structs
- Addressable at byte level

- **Temporary**
- EVM code execution
- Non-modifiable
- Max size 1024, Word 256 bit

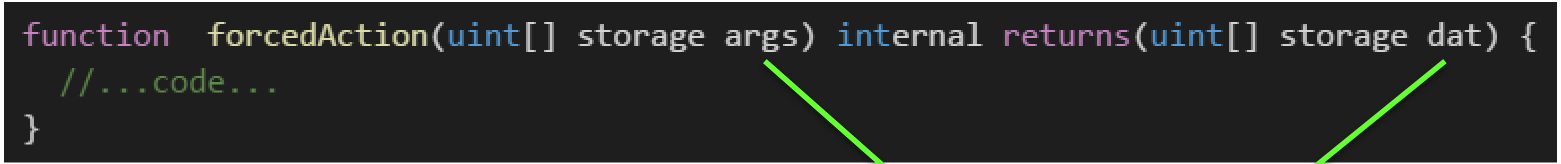
Local & Storage Variables



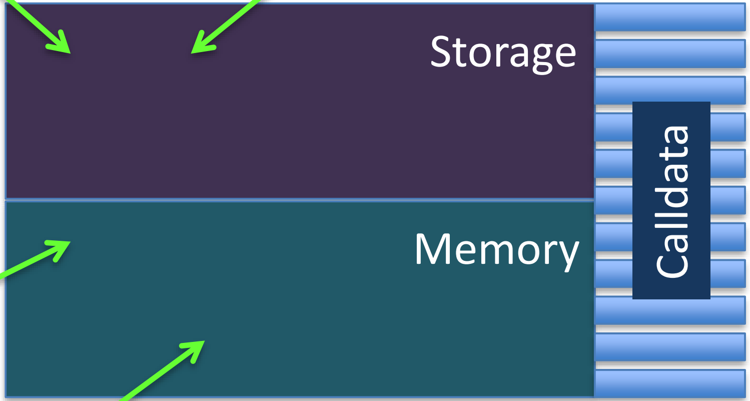
```
contract DataLocation {

  // Always in storage
  uint     count;
  uint[]   allPoints;

  function localVariables(){
    // This will give error
    ~~uint[]  localArray;~~

    uint[]  memory   memoryArray;

    // Creates a refernce
    uint[]  pointer = allPoints;

  }

}
```

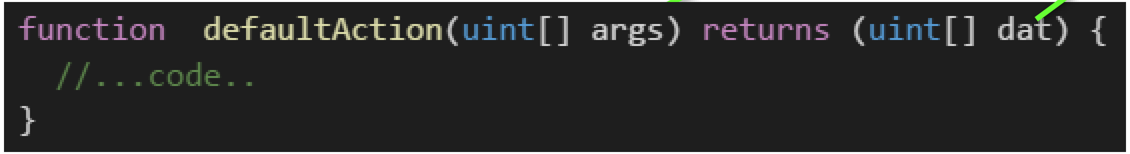


Function args



```
function  forcedAction(uint[] storage args) internal returns(uint[] storage dat) {
  //...code...
}
```





```
function  defaultAction(uint[] args) returns (uint[] dat) {
  //...code..
}
```

Solidity Contracts :

- Arrays

raj@acloudfan.com

@acloudfan

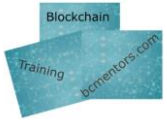
http://ACloudFan.com

Discount Coupon Links to UDEMY courses:

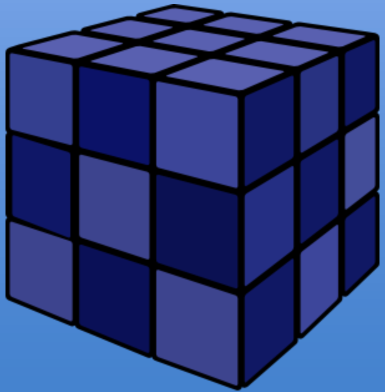
https://www.udemy.com/hyperledger/?couponCode=DKHLF1099

https://www.udemy.com/ethereum-dapp/?couponCode=DKETH1099

https://www.udemy.com/rest-api/?couponCode=DKRST1099



mentoring, seeking Blockchain part time work, project guidance, advice … …
http://www.bcmentors.com



This deck is part of a online course on "Ethereum: Design and Development of Decentralized Apps.

Static Arrays

- Fixed sized arrays

```
bool[10]  array;

bool element = array[4]


uint    len = array.length;

array.length = 6;
```

Dynamic Arrays

- Size can be changed at runtime

```
bool[ ]  array;

bool element = array[4]


uint    len = array.length;

array.length = 6;  // Storage
```

Initialization & Assignment

PS: **Storage** arrays only

```
uint8[3]   arr = [1,2,3]          // Implicit conversion


int8[3]    arr = [1,2,3]          // Compilation fails  elements interpreted as uint8


int8[ ]    arr = [int8(1),2,3]   // Gets compiled
```

Creating

Static Arrays

- bool   bool[10] array;
- uint   uint[10] array;

Dynamic Arrays

```
int8[ ]  array;   //Storage

array = new  int8[]( 10 );


array.push(5);

array = [1,2,3];
```

```
int8[ ] memory  array;

array = new  int8[]( 10 );

// Compiler errors
array.push(5);

array = [1,2,3];
```

Solidity Contracts :

- Special Arrays
  - Bytes
  - String type

raj@acloudfan.com

@acloudfan

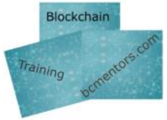
http://ACloudFan.com

Discount Coupon Links to UDEMY courses:

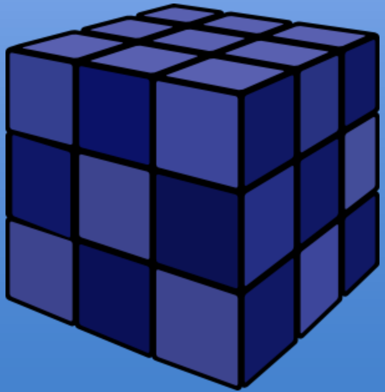
https://www.udemy.com/hyperledger/?couponCode=DKHLF1099

https://www.udemy.com/ethereum-dapp/?couponCode=DKETH1099

https://www.udemy.com/rest-api/?couponCode=DKRST1099



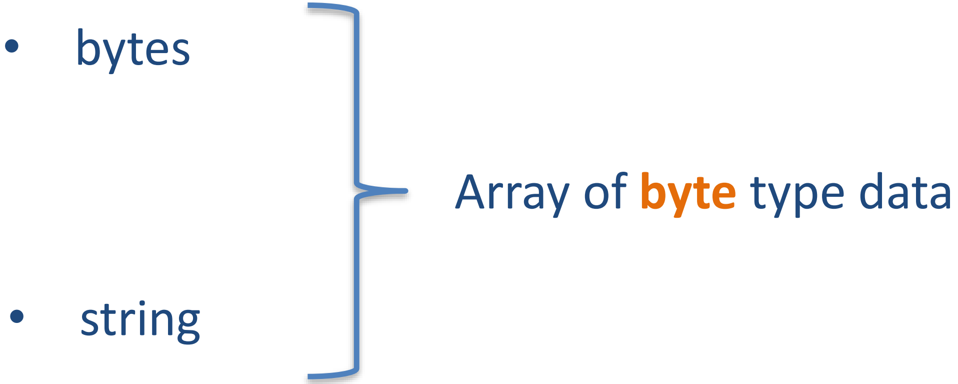
mentoring, seeking Blockchain part time work, project guidance, advice … …
http://www.bcmentors.com



This deck is part of a online course on “Ethereum: Design and Development of Decentralized Apps.

Special Arrays

- Variable of types:



- bytes
- string

Array of **byte** type data

byte Types

- byte data; // Single addressable byte

byte array

// Static	// Dynamic
• byte[15] data;	• byte[] data;
• **bytes**[1 – 32] data;	• **bytes** data;

- **bytes1** data; = byte[1] data;
- **bytes32** data; // 32 byte array

Fixed size bytes array

- bytes24 data;   // Fixed size = 24

  data[4] = 28;   data = [byte(1), 2, 3 …]   // Read-only

  data.length=10;   // Not allowed

- bytes32 bigger;   data = bigger;   // Fails compilation
- bytes16 smaller;   data = smaller;   // OK

byte[ ] data

// Storage arrays

```
data = new byte[](4);

data = [byte(1), 2,3,4];

data[1] = 1; // Read & Write

data.length=10;
```

bytes data

// Storage arrays

```
data = new bytes(4);

data = [byte(1), 2,3,4];  // Error

data[1] = 1; // Read & Write

data.length = 10;
```

string Type

- String is NOT a basic type
- Represents an arbitrary length UTF-8 encoded string
- Dynamically sized
- string = bytes, with some differences

String Literals

- string variable = “abc” or ‘abc’

  - Hex literals prefixed with **hex** E.g., ***hex****”001122”*

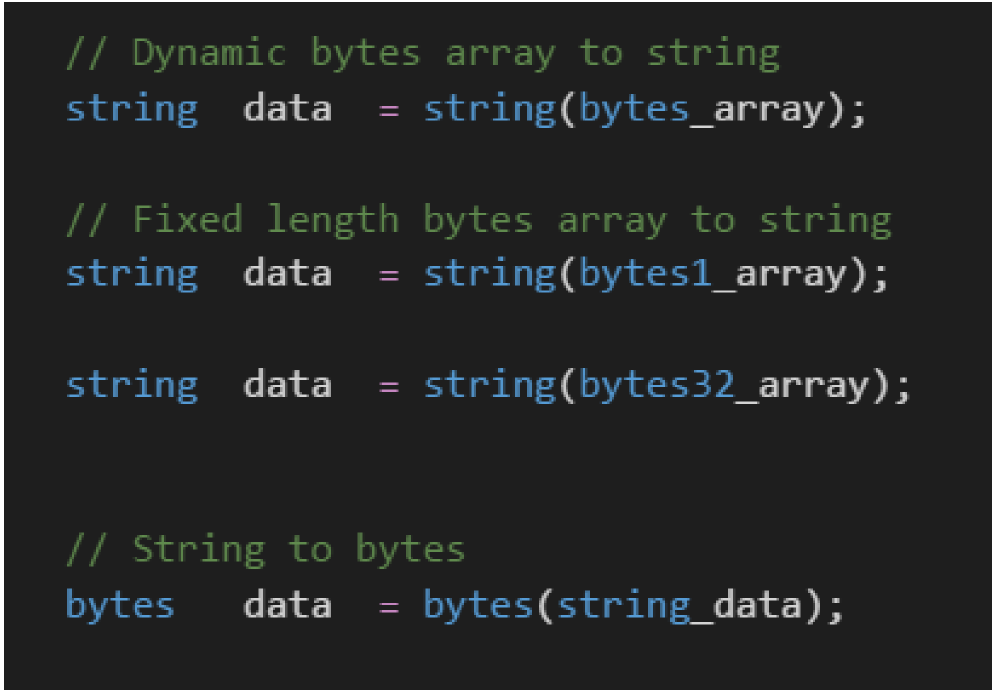
  - Supports the escape characters

    E.g., \n,

    E.g., \xNN for hex

    E.g., \uNNNN for UTF-8

Conversion



```
// Dynamic bytes array to string
string  data  = string(bytes_array);

// Fixed length bytes array to string
string  data  = string(bytes1_array);

string  data  = string(bytes32_array);


// String to bytes
bytes   data  = bytes(string_data);
```

string	*bytes*
• Fixed length NOT supported	• Fixed size supported using bytes(1-32)
• Index access not allowed *string[7]; // Error*	• Index access for Read returns *byte* *bytes[7]; // OK for memory & storage*
• Cannot be expanded i.e., push() NOT available	• *Storage* bytes may be expanded with push() operation

String Functions

- No out of the box support
  - External *StringUtil* libraries
  - Complex string operations may be costly

Solidity Contracts :

- Functions
- Tuples

Discount Coupon Links to UDEMY courses:

https://www.udemy.com/hyperledger/?couponCode=DKHLF1099

https://www.udemy.com/ethereum-dapp/?couponCode=DKETH1099

https://www.udemy.com/rest-api/?couponCode=DKRST1099

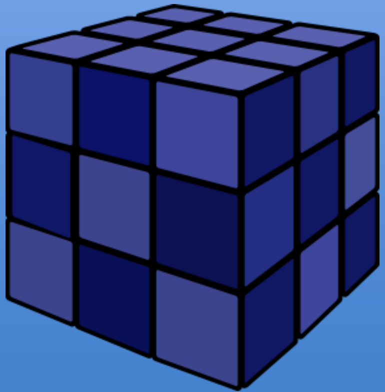
mentoring, seeking Blockchain part time work, project guidance, advice … …
http://www.bcmentors.com

This deck is part of a online course on “Ethereum: Design and Development of Decentralized Apps.

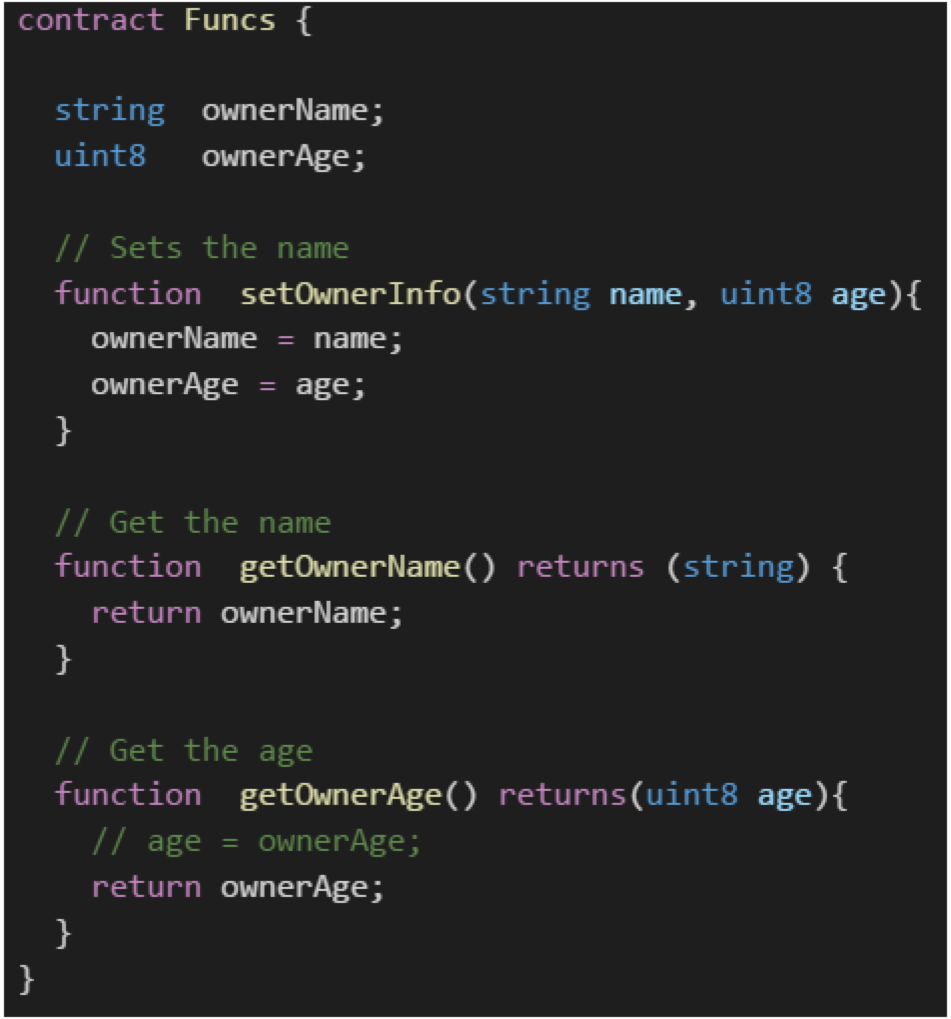
raj@acloudfan.com

@acloudfan

http://ACloudFan.com



Functions



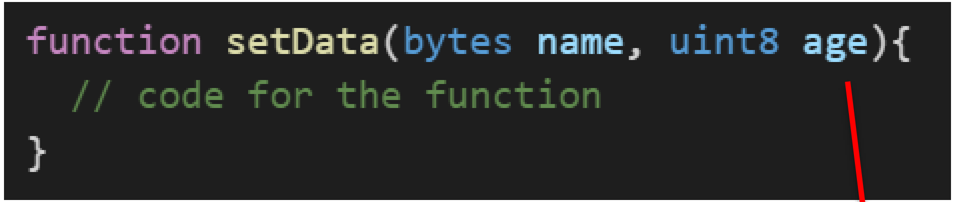
```
contract Funcs {

  string  ownerName;
  uint8   ownerAge;

  // Sets the name
  function  setOwnerInfo(string name, uint8 age){
    ownerName = name;
    ownerAge = age;
  }

  // Get the name
  function  getOwnerName() returns (string) {
    return ownerName;
  }

  // Get the age
  function  getOwnerAge() returns(uint8 age){
    // age = ownerAge;
    return ownerAge;
  }
}
```

Output Parameters

- Use keyword returns(…)
- Multiple return parameters
- You may name the return parameters
  - Named local variable available within the function body
  - Initialized to zeros
  - Values assigned to named variable are automatically returned

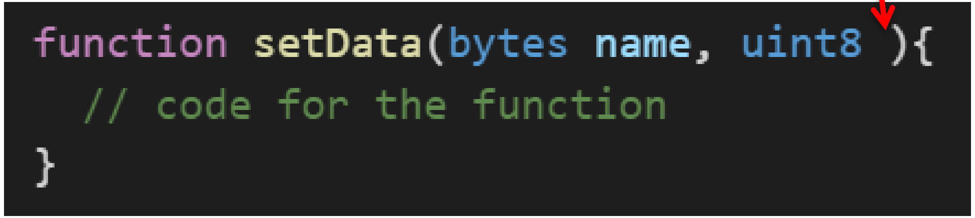
Input Parameters

- Declare the arguments with type/names



```
function setData(bytes name, uint8 age){
  // code for the function
}
```

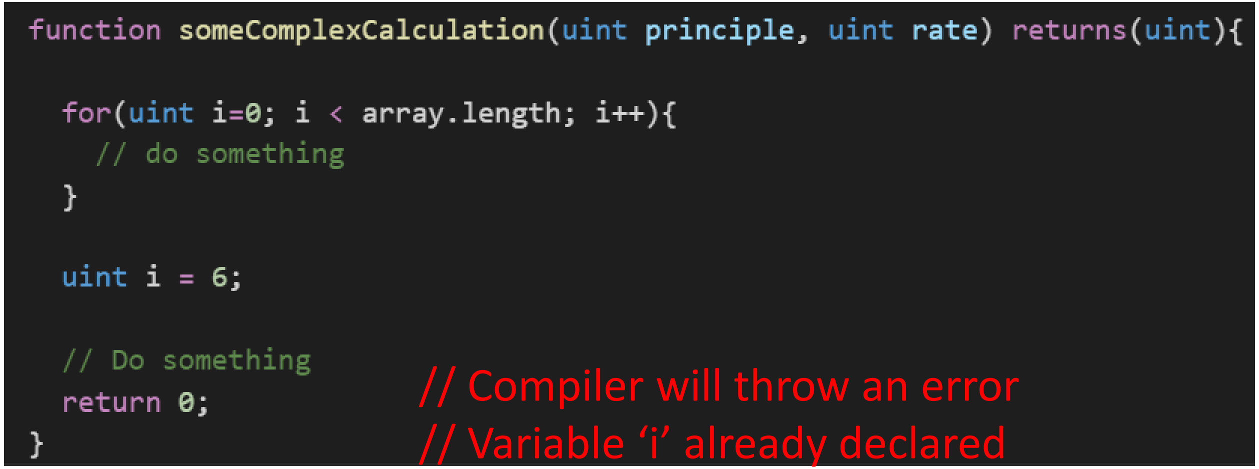
- *But* may omit argument name if unused



```
function setData(bytes name, uint8 ){
  // code for the function
}
```

Local Variables

- Re-declaration of the variable in the function not allowed



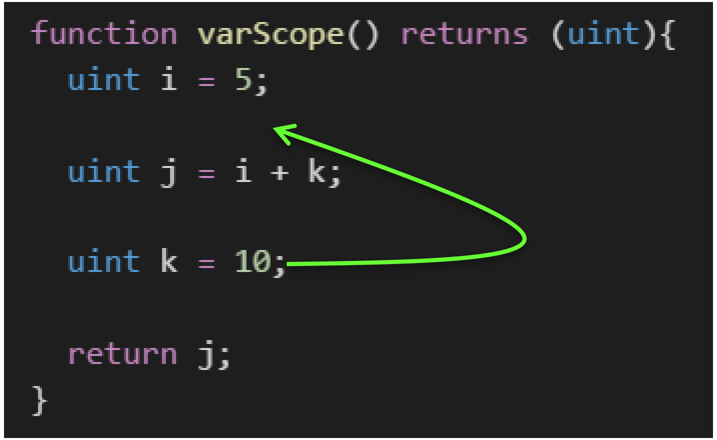
```
function someComplexCalculation(uint principle, uint rate) returns(uint){

  for(uint i=0; i < array.length; i++){
    // do something
  }

  uint i = 6;

  // Do something
  return 0;
}
```

// Compiler will throw an error
// Variable ‘i’ already declared

Variables Initialization

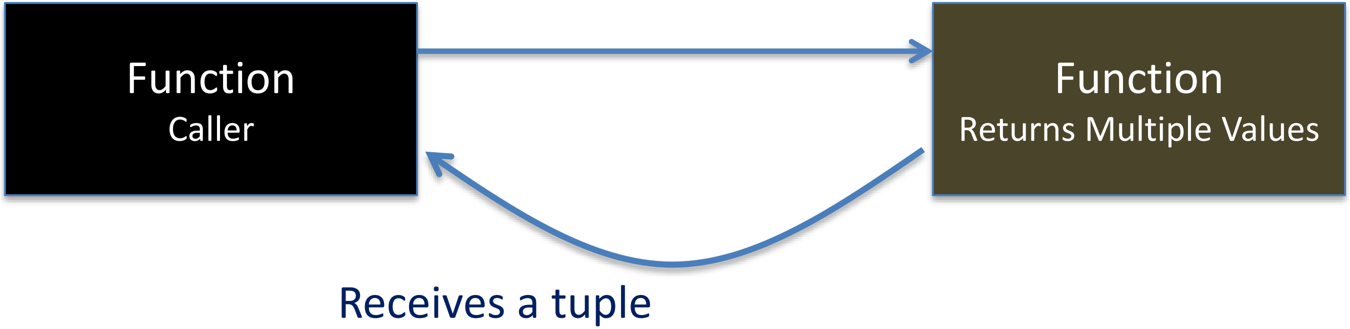
- Bytes initialized to 0s
- Bool to false

- Variables initialized to defaults in the beginning of the function



```
function varScope() returns (uint){
  uint i = 5;

  uint j = i + k;

  uint k = 10;

  return j;
}
```

Tuple types



Function

Caller

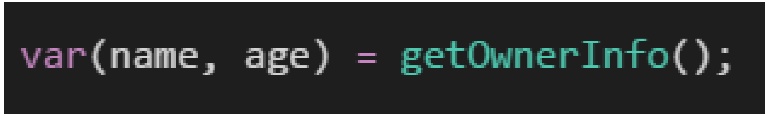
Function

Returns Multiple Values

Receives a tuple

Tuple types

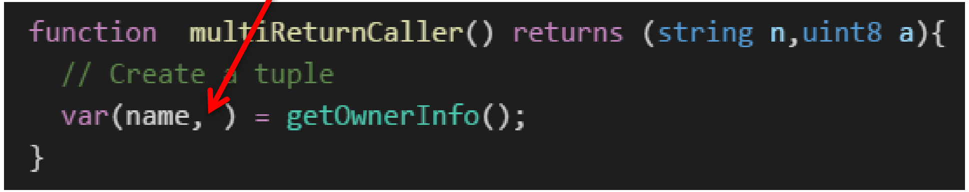
- A tuple is a list of objects



```
var(name, age) = getOwnerInfo();
```

  - Different types in tuple are OK

- You may skip a variable in tuple



```
function  multiReturnCaller() returns (string n,uint8 a){
  // Create a tuple
  var(name, ) = getOwnerInfo();
}
```