Introduction to Solidity: Coding Ethereum Smart Contracts

Session Three Scope

- **Scope**

Learn about ERC20, understand additional programming nuances, and build more complex apps to strengthen fundamentals,

- **What you will know**

What ERC20 is, how to build more nuanced smart contracts, know what modifiers are

- **Next Steps: Session Four**

Using session three knowledge we will delve deeper into ERC20 tokens and begin writing our own ERC20 token

Agenda

Session Three

blockchain nyc

Session Three Prerequisites

- Google Chrome
- Remix - https://remix.ethereum.org/

blockchain nyc

Loops and Modifiers

blockchain nyc

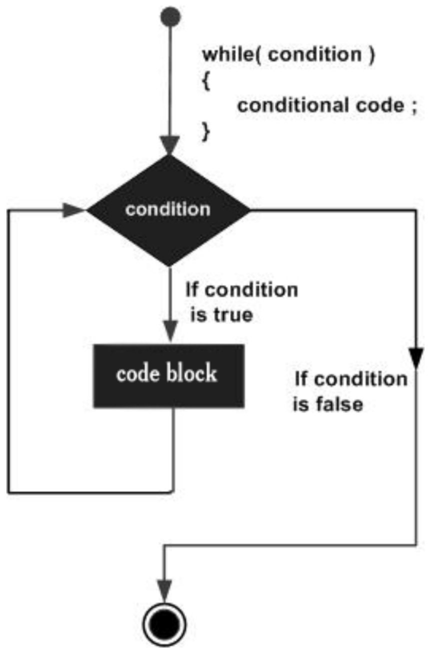
Function Modifiers

- *Owner*
- *Payable – this is a special modifier*

While Loops

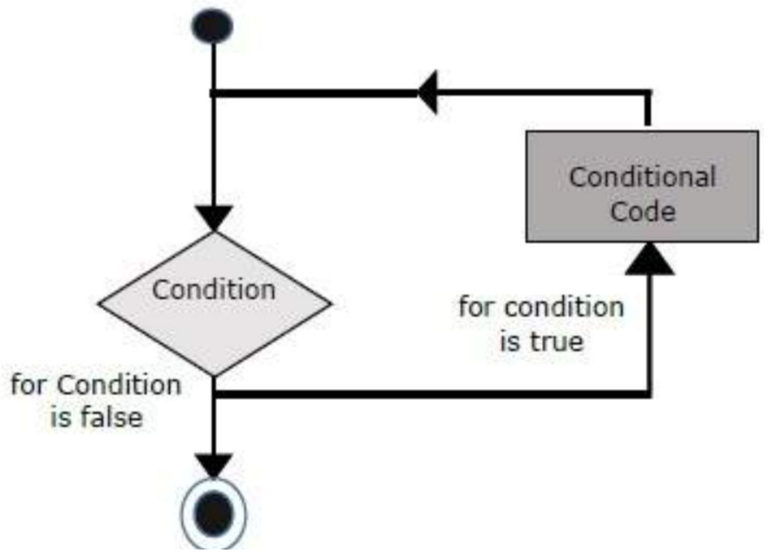
Sample Syntax:

```
while (expression) {
   Statement(s) to be executed
        if expression is true
}
```



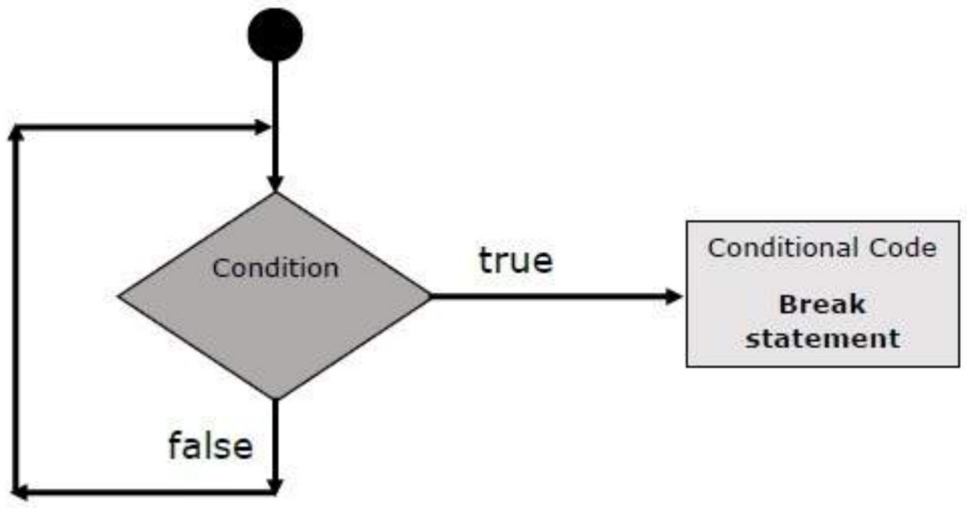
For Loops

Sample Syntax:
```
for (initialization; test condition;
iteration statement) {
   Statement(s) to be executed if test
       condition is true
}
```



blockchain nyc

Loop Control



First Application with Remix

Instructions:

Please create a new contract with the following components:

- Contract named “FirstApplication”
- Add a comment stating “This is my first Solidity Application”
- Create two variables that are both public. Both should be set to unsigned numbers.
- Create a function called “theadder”. Make it public and have it return a uint
- Create a while loop nested in the function. Make your counter the sum of your two variables. Make the counter reduce by one and set your while loop criteria so that the loop stops when counter equals 0. There should be a control flow to prevent the loop from running continuously; we want to control the reduction of the counter!
- Return a value of 1 in your function when the while loop terminates.

(Activity Length ~20 minutes)

Updating your First Application with Remix

Hello World Exampled: Setter/Getter Reference

```
pragma solidity ^0.4.0;

// A simple smart contract
contract MessageContract {
    string message = "Hello World";

    function getMessage() public constant returns(string) {
        return message;
    }

    function setMessage(string newMessage) public {
        message = newMessage;
    }
}
```

Instructions:

Please update your application with the following:

- Please add a setter method to update the counter variable. This should update the number_a and number_b variables as well.
- Please add a getter method to return the value of your new counter
- Please change the scope on theadder function to "`payable`"

HINT: You need to modify the current way that the counter variable is stored.

(Activity Length ~15 minutes)

Appendix:
Additional Learning Resources

- *Tutorialspoint - https://www.tutorialspoint.com/solidity/index.htm*
- *ERC20 Tutorial - https://betterprogramming.pub/python-blockchai token-deployment-tutorial-create-an-erc20-77a5fd2e1a58*

blockchain nyc

Solution Set

First Application with Remix

```
pragma solidity 0.6.4;
contract FirstApplication {
    /*This is my first Solidity
Application*/
    uint public number_a = 10;
    uint public number_b = 20;
    function theadder() public view returns
(uint){
        uint counter = number_a + number_b;
        while (counter > 0) {
            counter-=1;
            break;
        }
        return 1;
    }
}
```

Updating your First Application with Remix

```
pragma solidity 0.6.4;
contract FirstApplication {
    /*This is my first Solidity Application*/
    uint public number_a = 10;
    uint public number_b = 20;
    uint counter = number_a + number_b;

    function getCounter() public view returns(uint){
        return counter;
    }

    function setCounter(uint Newnumber_a, uint Newnumber_b) public {
        number_a = Newnumber_a;
        number_b = Newnumber_b;
        counter = number_a + number_b;
    }

    function theadder() public payable returns (uint){
        while (counter > 0) {
            counter-=1;
            break;
        }
        return 1;
    }
}
```