C# Ternary Expression

This lesson discusses ternary operators in detail including compound ternary expressions using examples

WE'LL COVER THE FOLLOWING

Ternay Operator

This is a short way of representing *conditional statement* in **C#**.

Ternary *operator* has one **boolean** expression, and *returns* **one** of two values depending on the value of a **Boolean** expression.

Syntax

Here's the syntax

```
condition ? expression_if_true : expression_if_false;
```

Ternary Operator Syntax

Example

Let's take a look at an example which uses *ternary* operators.

```
using System;
```

```
class TernaryExample
{
    static void Main()
    {
        string name = "Frank"; //change this name to see the false condition execute
        Console.WriteLine(name == "Frank" ? "The name is Frank" : "The name is not Frank");
    }
}
```

Code Explanation

In the code above

- The string `name` is set to **"Frank"**
- Hence in line **8**, *expression* **1** executes as it evaluates to **true**
- At the end the output dispayed is: **The name is Frank**

You can try changing the name to another, for example, "**Mary**"

- Now *expression* **2** will execute since the *condition* will evaluate to **false** this time as `name` does not equal **"frank"**
- At the end, the output displayed will be: **The name is not Frank**

Compound Ternary Expressions

The ternary operator is **right-associative** which allows for **compound** *ternary* expressions to be used.

- This is done by adding additional *ternary* equations in either the **true** or **false** position of a **parent** *ternary* equation.

> **Note:** Care should be taken to ensure readability, but this can be useful shorthand in some circumstances.

Example

Let's take a look at an example for **compound** *ternary* expressions.

```
using System;
```

```
class CompoundTernaryExample
{
    static void Main()

    {
      //case 1 will execute as x is greater than y
      int x = 5;
      int y = 4;
      Console.WriteLine((x > y) ? "x is greater than y" : (x < y) ? "x is less than y" : (x =

      //case 2 will execute as x is less than y
      x = 4;
      y = 5;
      Console.WriteLine((x > y) ? "x is greater than y" : (x < y) ? "x is less than y" : (x =

      //case 3 will execute as x is equal to y
      x = 5;
      y = 5;
      Console.WriteLine((x > y) ? "x is greater than y" : (x < y) ? "x is less than y" : "x i

    }
}
```

Code Explanation

In the code above **Compound** *ternary* operation evaluates **two** conditions

- `x>y`
- `x<y`

and if both of them don’t evaluate to **true** that means

- `x==y`

so the last case is executed.

First, we set `x>y` so

- **x is greater than y** is displayed in line **10**

Second, we set x<y so

- **x is less than y** is displayed in line **15**

Lastly, we set x==y so

- **x is equal to y** is displayed in line **20**

This marks the end of our discussion on *conditional statements* in C#. In the

This marks the end of our discussion on *conditional statements* in C#. In the upcoming chapter we will discuss more interesting concepts such as **loops** in

C#!