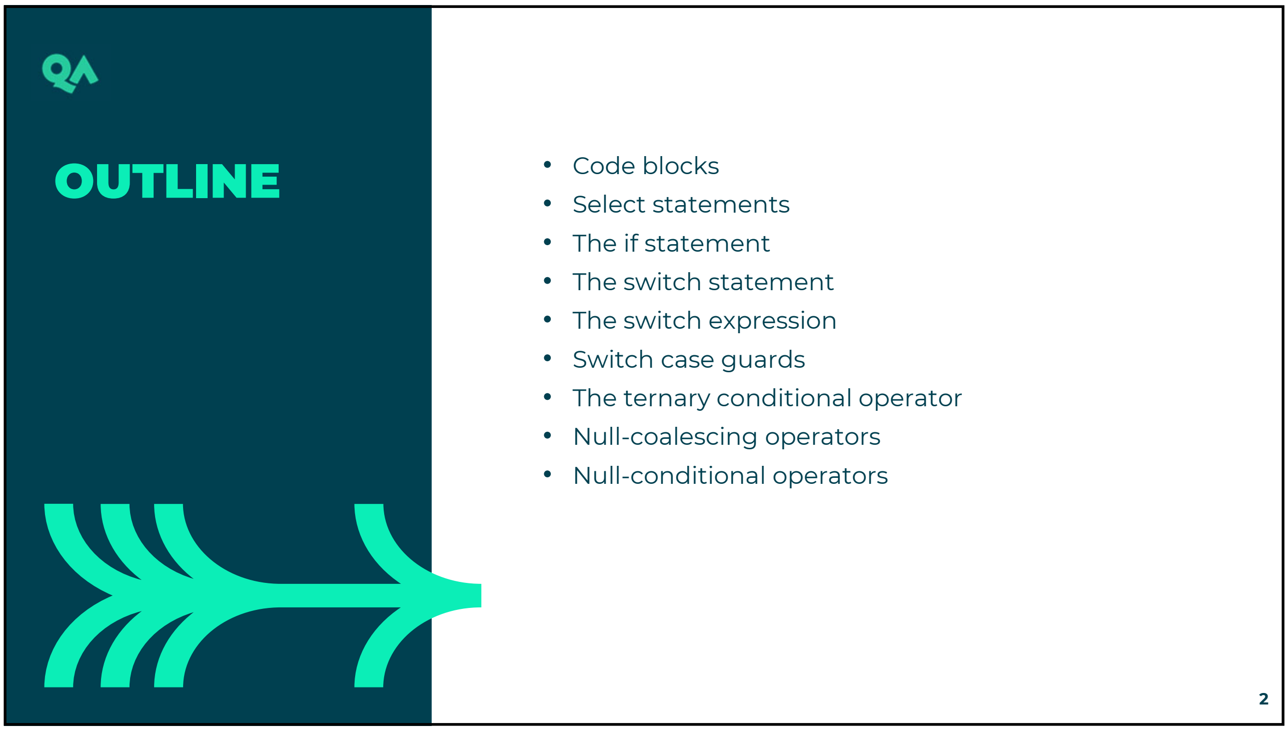
Conditionals

1

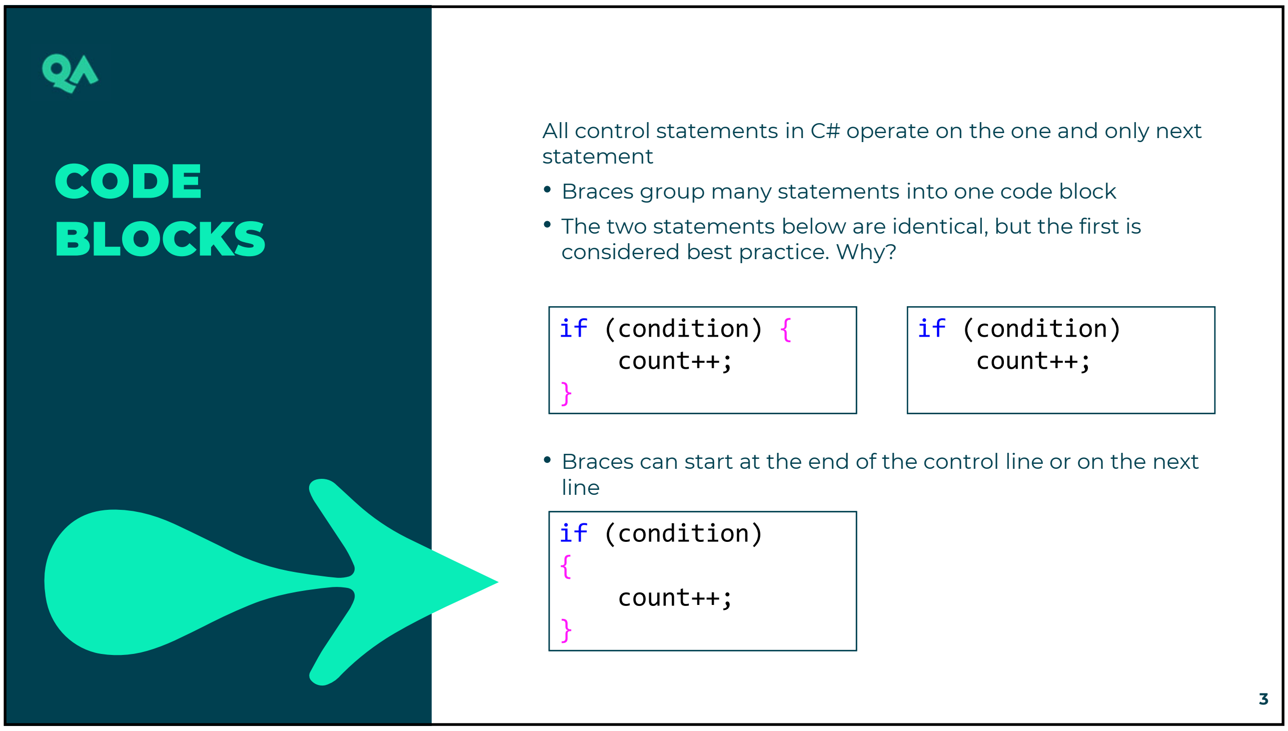


OUTLINE

- Code blocks
- Select statements
- The if statement
- The switch statement
- The switch expression
- Switch case guards
- The ternary conditional operator
- Null-coalescing operators
- Null-conditional operators

2

2



CODE BLOCKS

All control statements in C# operate on the one and only next statement

- Braces group many statements into one code block
- The two statements below are identical, but the first is considered best practice. Why?

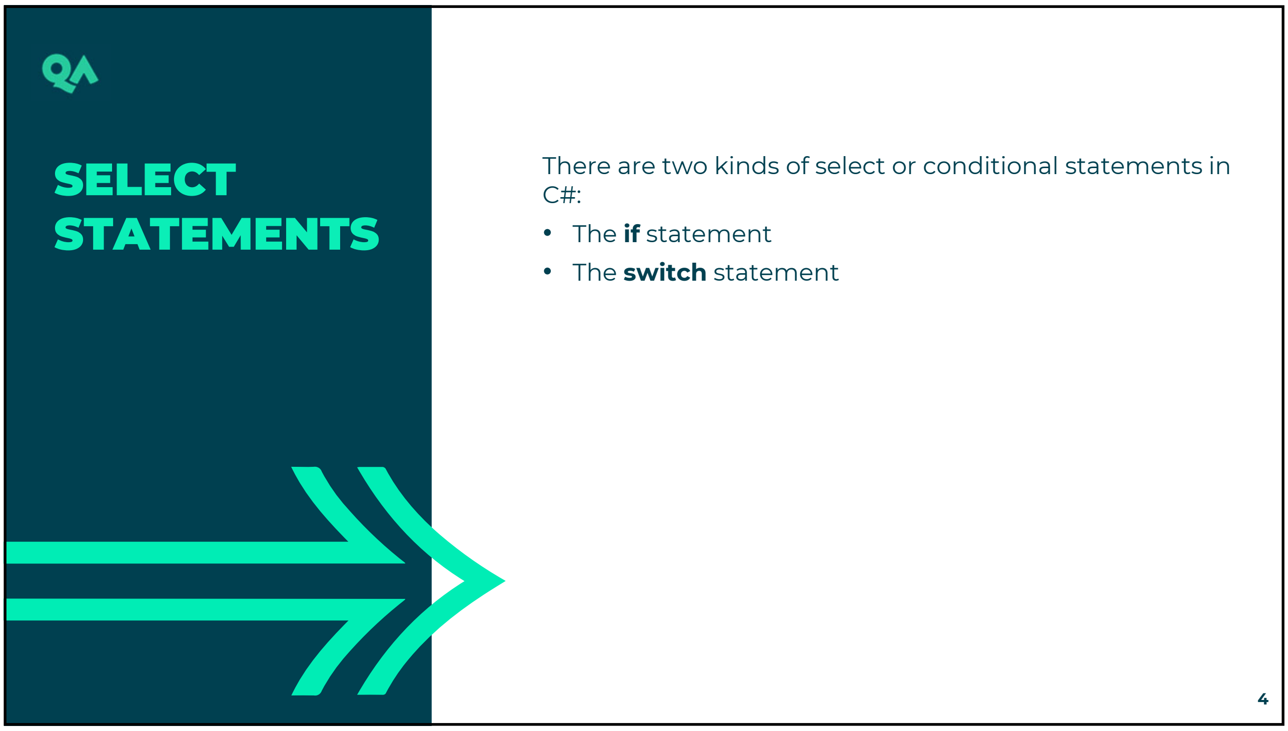
```
if (condition) {
    count++;
}
```

```
if (condition)
    count++;
```

- Braces can start at the end of the control line or on the next line

```
if (condition)
{
    count++;
}
```

3

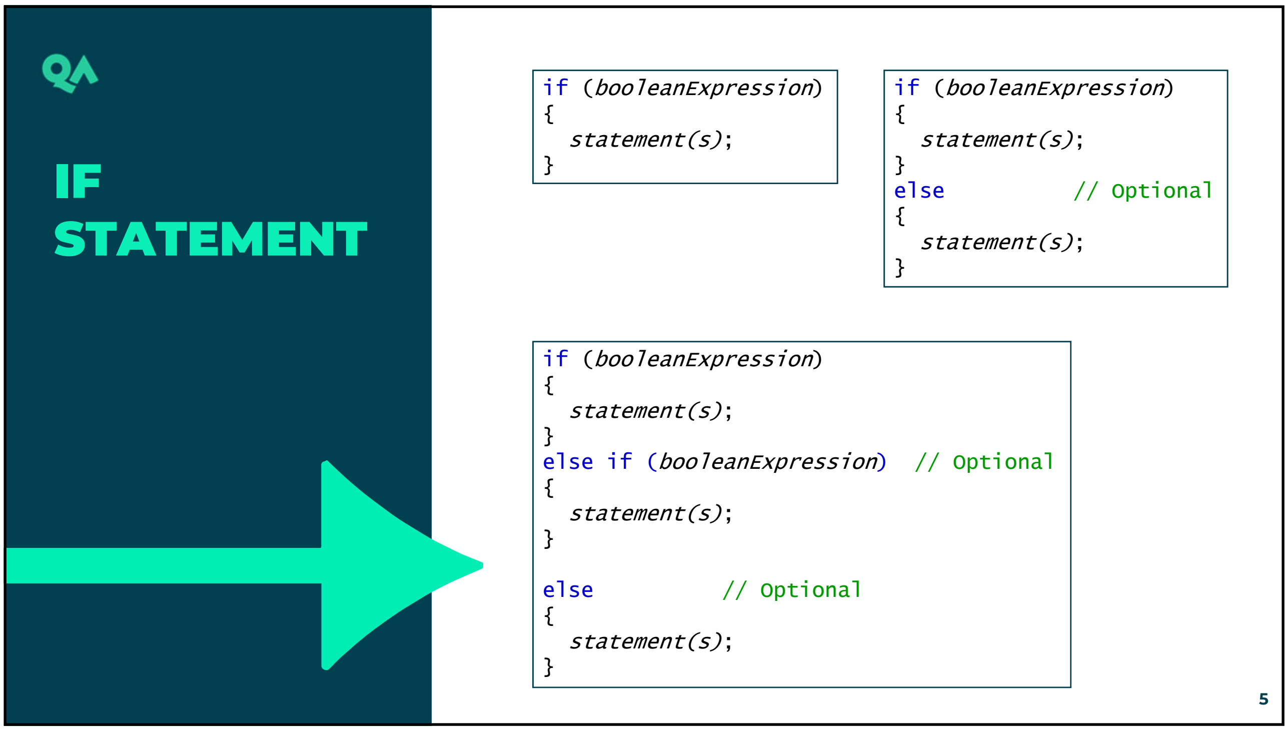


SELECT STATEMENTS

There are two kinds of select or conditional statements in C#:

- The **if** statement
- The **switch** statement

4



IF STATEMENT

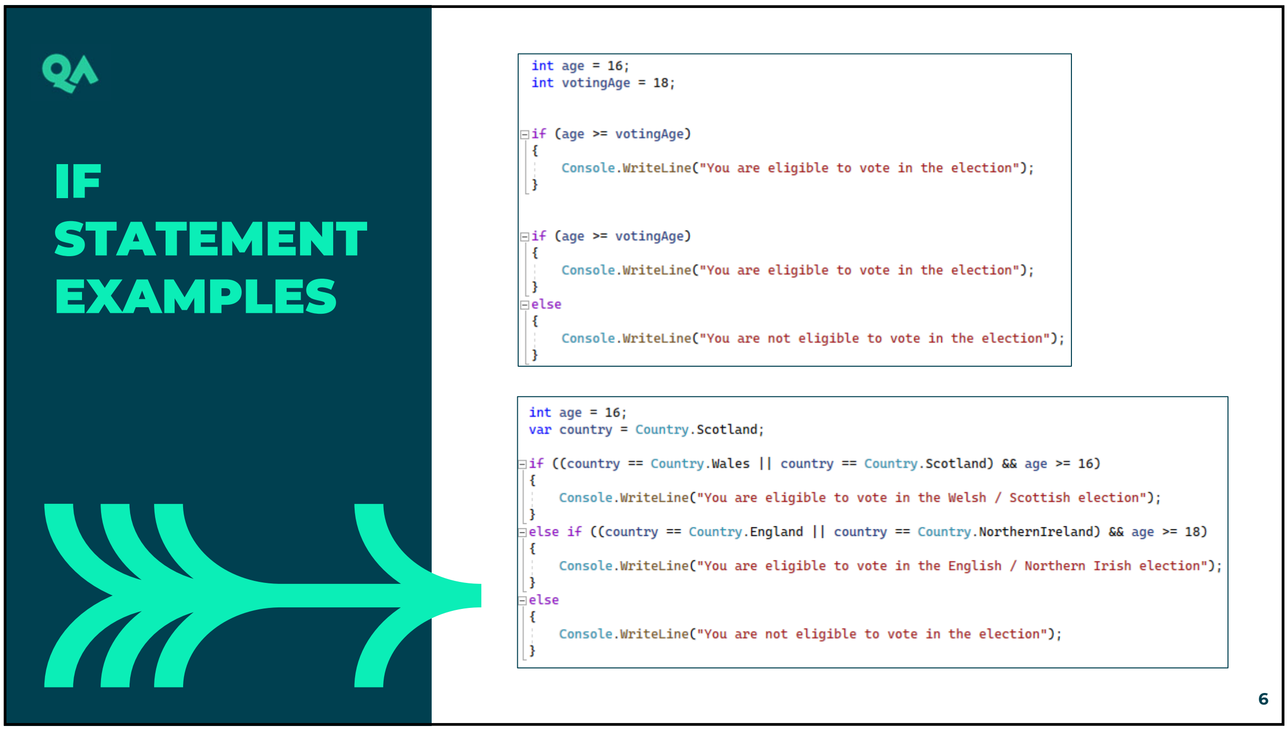
```
if (booleanExpression)
{
  statement(s);
}
```

```
if (booleanExpression)
{
  statement(s);
}
else          // Optional
{
  statement(s);
}
```

```
if (booleanExpression)
{
  statement(s);
}
else if (booleanExpression)   // Optional
{
  statement(s);
}

else          // Optional
{
  statement(s);
}
```

5

5



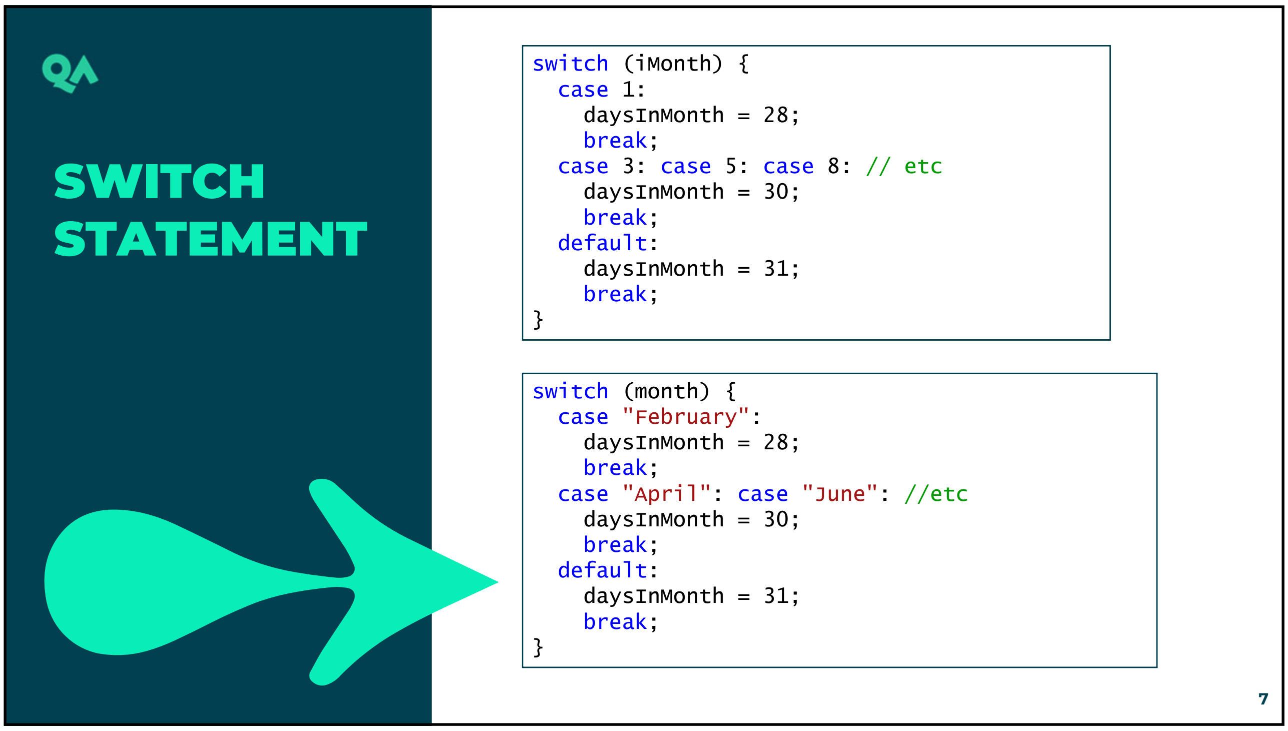
IF STATEMENT EXAMPLES

```
int age = 16;
int votingAge = 18;


if (age >= votingAge)
{
    Console.WriteLine("You are eligible to vote in the election");
}


if (age >= votingAge)
{
    Console.WriteLine("You are eligible to vote in the election");
}
else
{
    Console.WriteLine("You are not eligible to vote in the election");
}
```

```
int age = 16;
var country = Country.Scotland;

if ((country == Country.Wales || country == Country.Scotland) && age >= 16)
{
    Console.WriteLine("You are eligible to vote in the Welsh / Scottish election");
}
else if ((country == Country.England || country == Country.NorthernIreland) && age >= 18)
{
    Console.WriteLine("You are eligible to vote in the English / Northern Irish election");
}
else
{
    Console.WriteLine("You are not eligible to vote in the election");
}
```

6

6



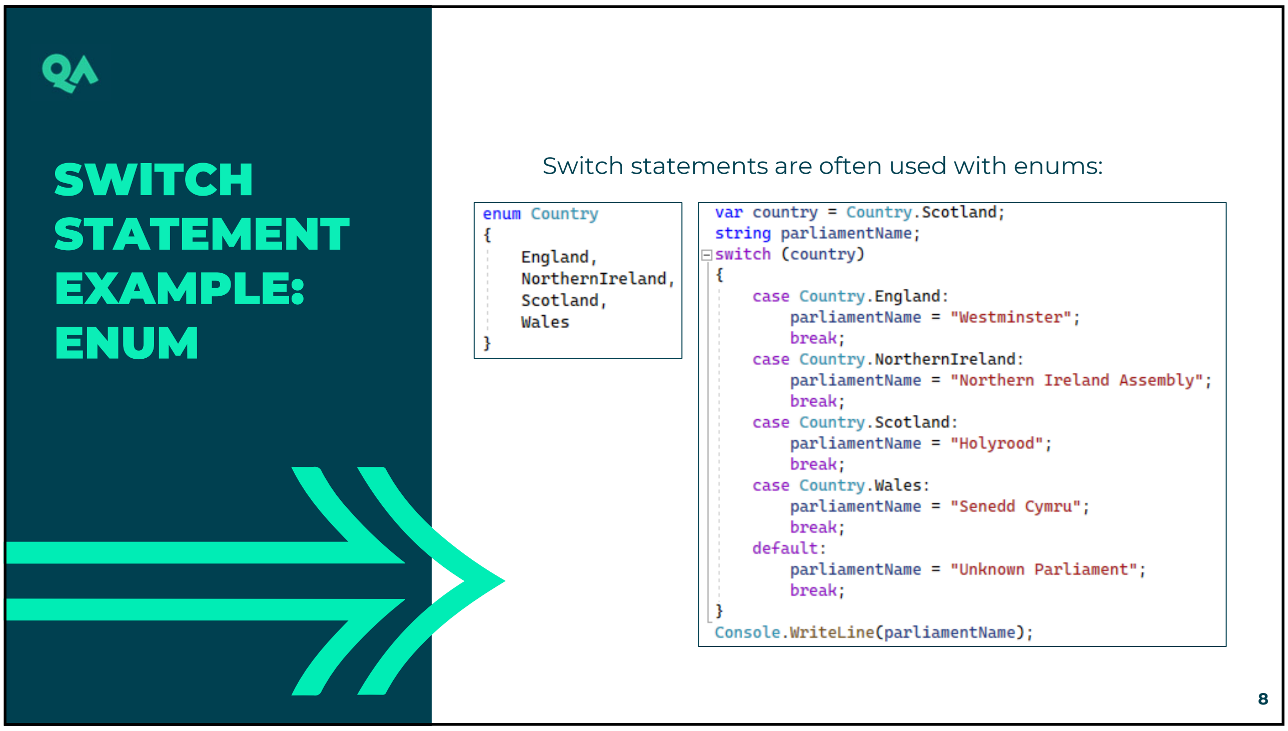
SWITCH STATEMENT

```
switch (iMonth) {
  case 1:
    daysInMonth = 28;
    break;
  case 3: case 5: case 8: // etc
    daysInMonth = 30;
    break;
  default:
    daysInMonth = 31;
    break;
}
```

```
switch (month) {
  case "February":
    daysInMonth = 28;
    break;
  case "April": case "June": //etc
    daysInMonth = 30;
    break;
  default:
    daysInMonth = 31;
    break;
}
```

7

7



SWITCH STATEMENT EXAMPLE: ENUM

Switch statements are often used with enums:

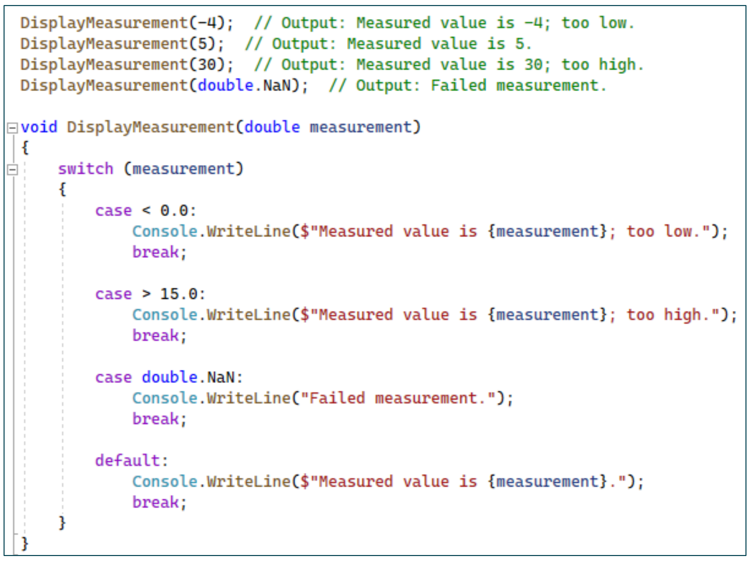
```
enum Country
{
    England,
    NorthernIreland,
    Scotland,
    Wales
}
```

```
var country = Country.Scotland;
string parliamentName;
switch (country)
{
    case Country.England:
        parliamentName = "Westminster";
        break;
    case Country.NorthernIreland:
        parliamentName = "Northern Ireland Assembly";
        break;
    case Country.Scotland:
        parliamentName = "Holyrood";
        break;
    case Country.Wales:
        parliamentName = "Senedd Cymru";
        break;
    default:
        parliamentName = "Unknown Parliament";
        break;
}
Console.WriteLine(parliamentName);
```

8

8

SWITCH STATEMENT EXAMPLE



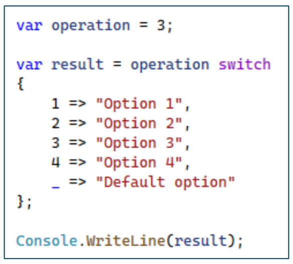
```
DisplayMeasurement(-4);  // Output: Measured value is -4; too low.
DisplayMeasurement(5);  // Output: Measured value is 5.
DisplayMeasurement(30);  // Output: Measured value is 30; too high.
DisplayMeasurement(double.NaN);  // Output: Failed measurement.

void DisplayMeasurement(double measurement)
{
    switch (measurement)
    {
        case < 0.0:
            Console.WriteLine($"Measured value is {measurement}; too low.");
            break;

        case > 15.0:
            Console.WriteLine($"Measured value is {measurement}; too high.");
            break;

        case double.NaN:
            Console.WriteLine("Failed measurement.");
            break;

        default:
            Console.WriteLine($"Measured value is {measurement}.");
            break;
    }
}
```

9

9

SWITCH EXPRESSION

- A **switch expression** is a more lightweight syntax than a *switch statement*
- They do <u>not</u> use **case, break,** or **default** keywords
- They use patterns and expressions separated by an **=>** arrow token
- The underscore **_** is a *discard pattern* which matches any expression, including **null**

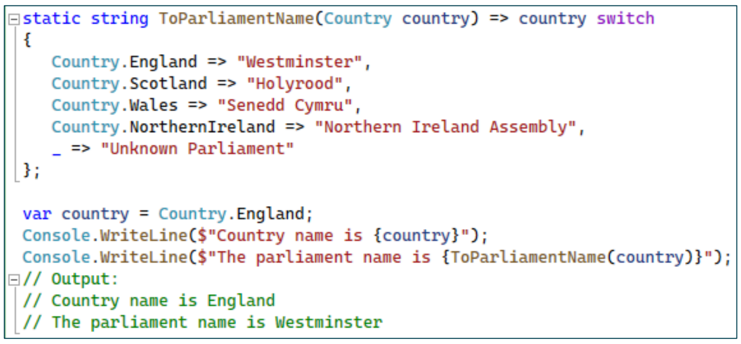


```
var operation = 3;

var result = operation switch
{
    1 => "Option 1",
    2 => "Option 2",
    3 => "Option 3",
    4 => "Option 4",
    _ => "Default option"
};

Console.WriteLine(result);
```

10

10

SWITCH EXPRESSION EXAMPLE



```
static string ToParliamentName(Country country) => country switch
{
    Country.England => "Westminster",
    Country.Scotland => "Holyrood",
    Country.Wales => "Senedd Cymru",
    Country.NorthernIreland => "Northern Ireland Assembly",
    _ => "Unknown Parliament"
};

var country = Country.England;
Console.WriteLine($"Country name is {country}");
Console.WriteLine($"The parliament name is {ToParliamentName(country)}");
// Output:
// Country name is England
// The parliament name is Westminster
```

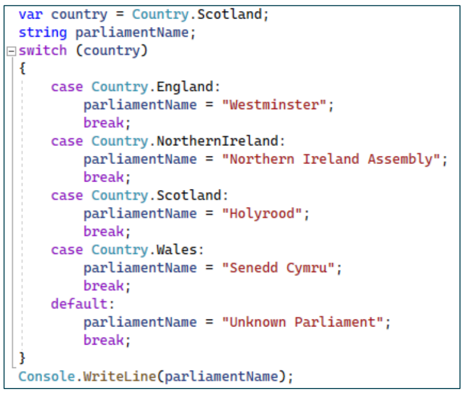
`Country.England => “Westminster”,`

- Is called an *expression arm*

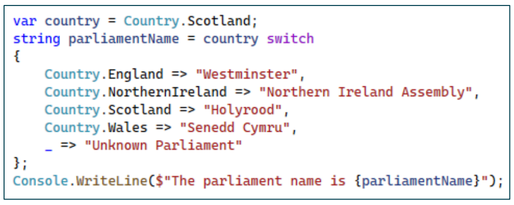
11

11

SWITCH STATEMENT VERSUS SWITCH EXPRESSION



```
var country = Country.Scotland;
string parliamentName;
switch (country)
{
    case Country.England:
        parliamentName = "Westminster";
        break;
    case Country.NorthernIreland:
        parliamentName = "Northern Ireland Assembly";
        break;
    case Country.Scotland:
        parliamentName = "Holyrood";
        break;
    case Country.Wales:
        parliamentName = "Senedd Cymru";
        break;
    default:
        parliamentName = "Unknown Parliament";
        break;
}
Console.WriteLine(parliamentName);
```



```
var country = Country.Scotland;
string parliamentName = country switch
{
    Country.England => "Westminster",
    Country.NorthernIreland => "Northern Ireland Assembly",
    Country.Scotland => "Holyrood",
    Country.Wales => "Senedd Cymru",
    _ => "Unknown Parliament"
};
Console.WriteLine($"The parliament name is {parliamentName}");
```

12

12

SWITCH CASE GUARDS

Expression arms contain:

- A *pattern*
- An optional **case guard**
- The **=>** arrow token
- An *expression*

A **case guard** is an additional condition that must be satisfied together with the matched pattern.

A case guard must be a Boolean expression.

Specify the case guard after the **when** keyword that follows a pattern.

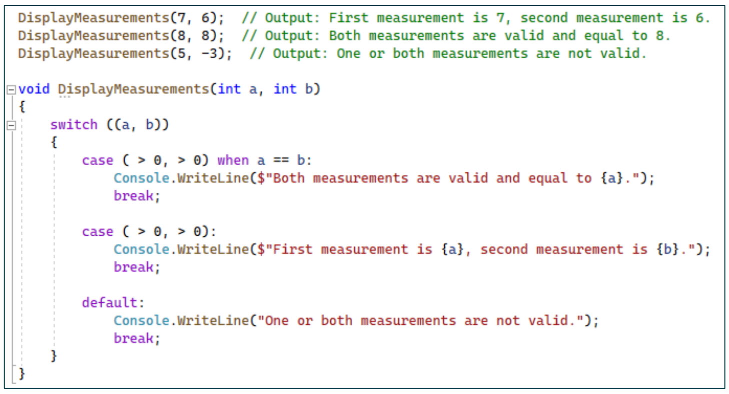
13

13

SWITCH STATEMENT CASE GUARD EXAMPLE

The case guard is:

when a == b

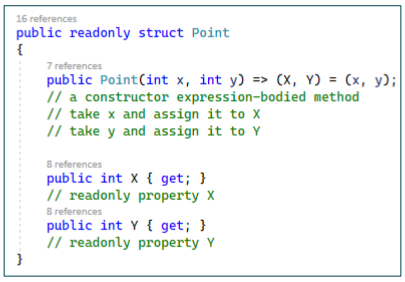


```
DisplayMeasurements(7, 6);  // Output: First measurement is 7, second measurement is 6.
DisplayMeasurements(8, 8);  // Output: Both measurements are valid and equal to 8.
DisplayMeasurements(5, -3);  // Output: One or both measurements are not valid.

void DisplayMeasurements(int a, int b)
{
    switch ((a, b))
    {
        case ( > 0, > 0) when a == b:
            Console.WriteLine($"Both measurements are valid and equal to {a}.");
            break;

        case ( > 0, > 0):
            Console.WriteLine($"First measurement is {a}, second measurement is {b}.");
            break;

        default:
            Console.WriteLine("One or both measurements are not valid.");
            break;
    }
}
```

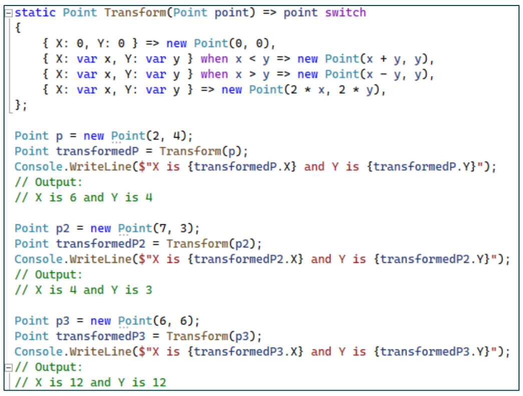
14

14

SWITCH EXPRESSION CASE GUARD EXAMPLE



```
16 references
public readonly struct Point
{
    7 references
    public Point(int x, int y) => (X, Y) = (x, y);
    // a constructor expression-bodied method
    // take x and assign it to X
    // take y and assign it to Y

    8 references
    public int X { get; }
    // readonly property X
    8 references
    public int Y { get; }
    // readonly property Y
}
```



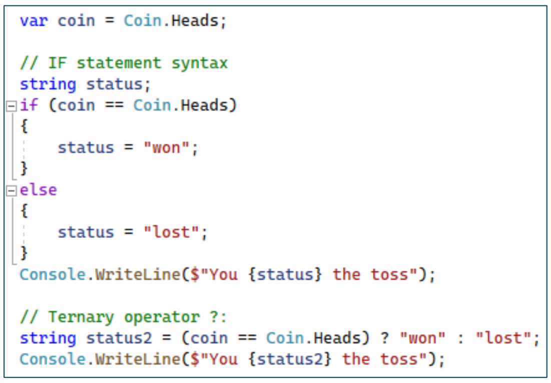
```
static Point Transform(Point point) => point switch
{
    { X: 0, Y: 0 } => new Point(0, 0),
    { X: var x, Y: var y } when x < y => new Point(x + y, y),
    { X: var x, Y: var y } when x > y => new Point(x - y, y),
    { X: var x, Y: var y } => new Point(2 * x, 2 * y),
};

Point p = new Point(2, 4);
Point transformedP = Transform(p);
Console.WriteLine($"X is {transformedP.X} and Y is {transformedP.Y}");
// Output:
// X is 6 and Y is 4

Point p2 = new Point(7, 3);
Point transformedP2 = Transform(p2);
Console.WriteLine($"X is {transformedP2.X} and Y is {transformedP2.Y}");
// Output:
// X is 4 and Y is 3

Point p3 = new Point(6, 6);
Point transformedP3 = Transform(p3);
Console.WriteLine($"X is {transformedP3.X} and Y is {transformedP3.Y}");
// Output:
// X is 12 and Y is 12
```

15

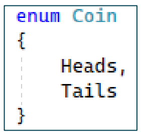
15

TERNARY CONDITIONAL OPERATOR ?:

The *ternary conditional operator* **?:** is a short-hand alternative to an **if** statement with two branches:



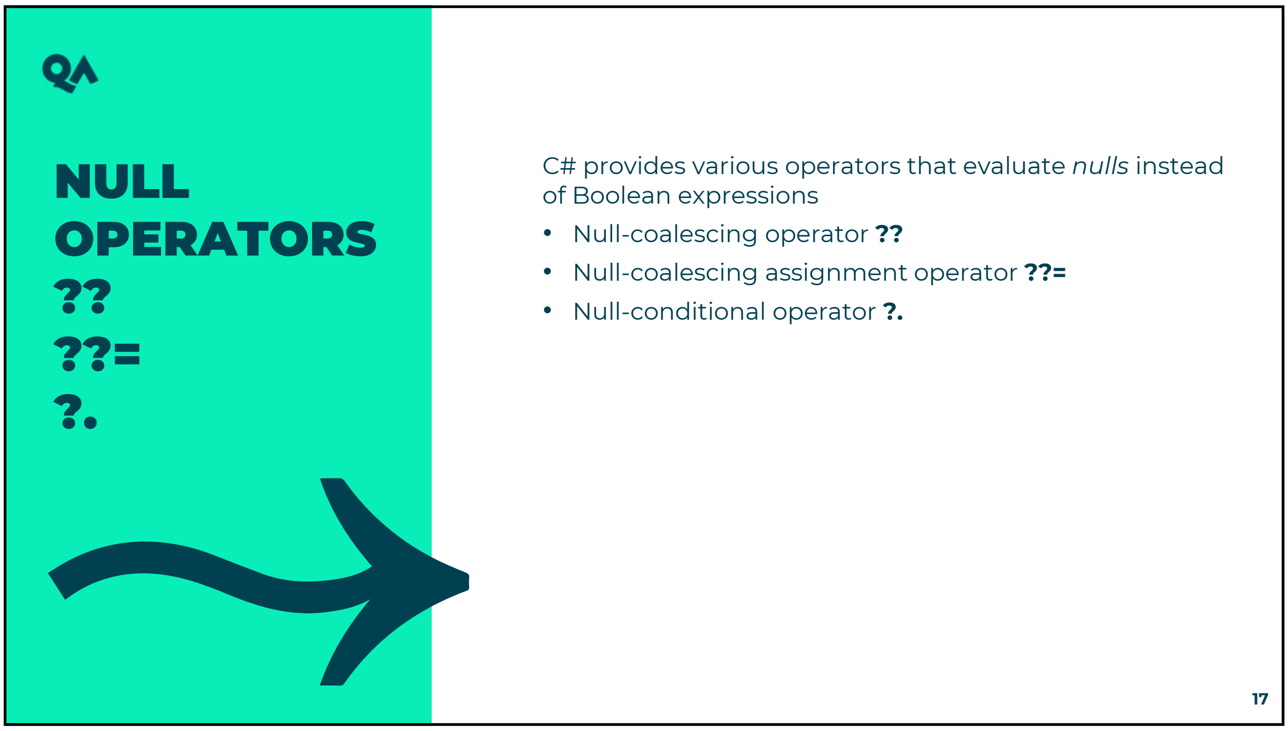
```
var coin = Coin.Heads;

// IF statement syntax
string status;
if (coin == Coin.Heads)
{
    status = "won";
}
else
{
    status = "lost";
}
Console.WriteLine($"You {status} the toss");

// Ternary operator ?:
string status2 = (coin == Coin.Heads) ? "won" : "lost";
Console.WriteLine($"You {status2} the toss");
```



```
enum Coin
{
    Heads,
    Tails
}
```

16

16



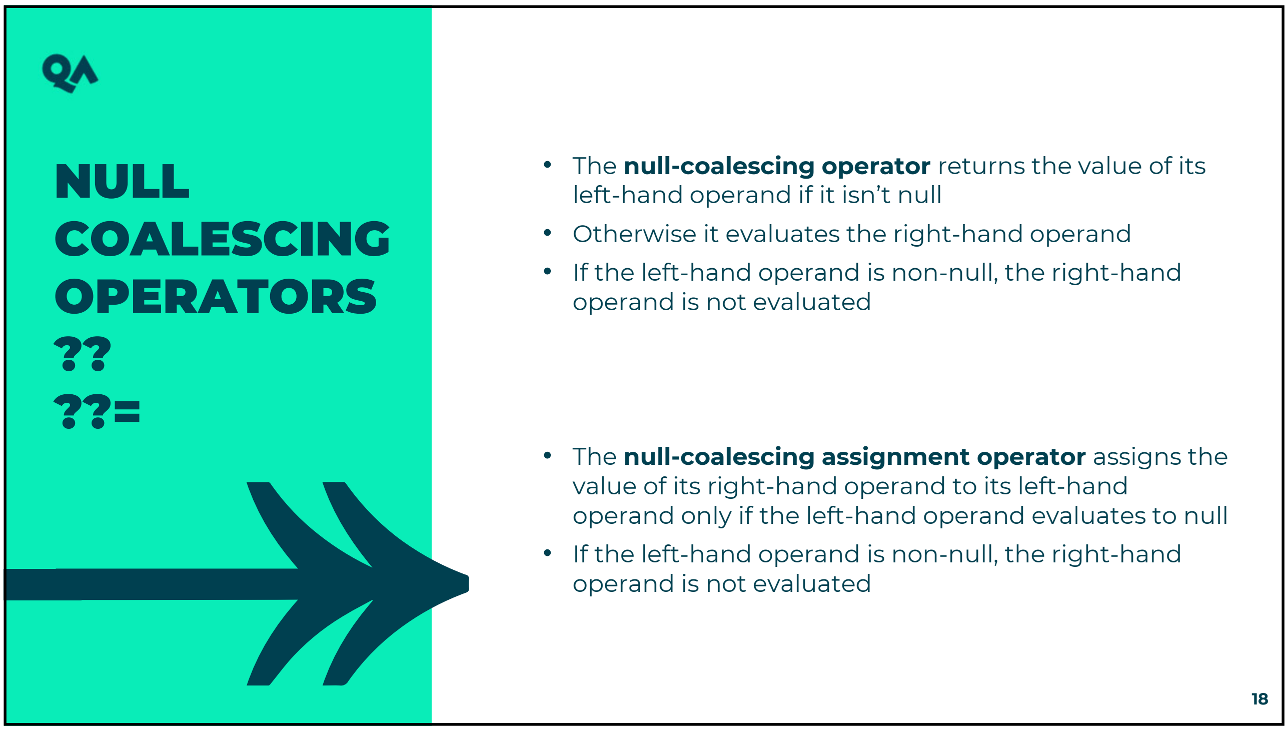
NULL OPERATORS ?? ??= ?.

C# provides various operators that evaluate *nulls* instead of Boolean expressions

- Null-coalescing operator **??**
- Null-coalescing assignment operator **??=**
- Null-conditional operator **?.**

17

17



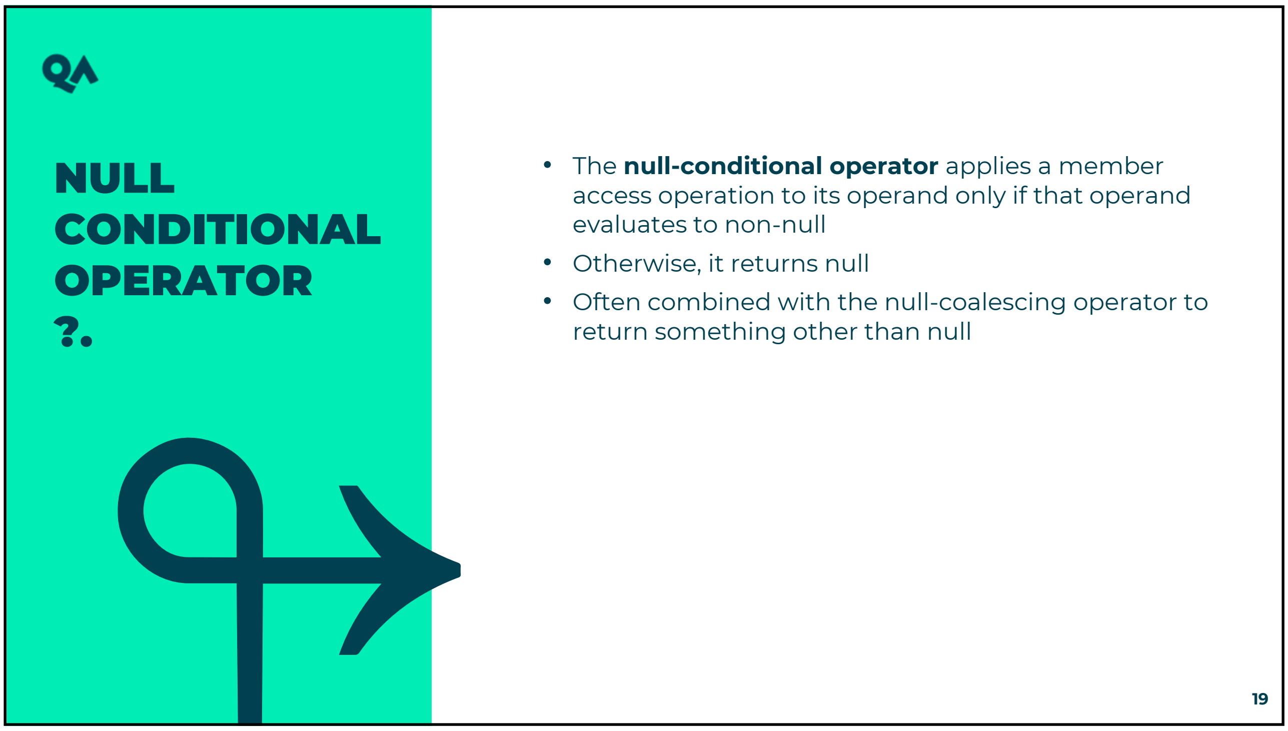
NULL COALESCING OPERATORS ?? ??=

- The **null-coalescing operator** returns the value of its left-hand operand if it isn't null
- Otherwise it evaluates the right-hand operand
- If the left-hand operand is non-null, the right-hand operand is not evaluated

- The **null-coalescing assignment operator** assigns the value of its right-hand operand to its left-hand operand only if the left-hand operand evaluates to null
- If the left-hand operand is non-null, the right-hand operand is not evaluated

18

18

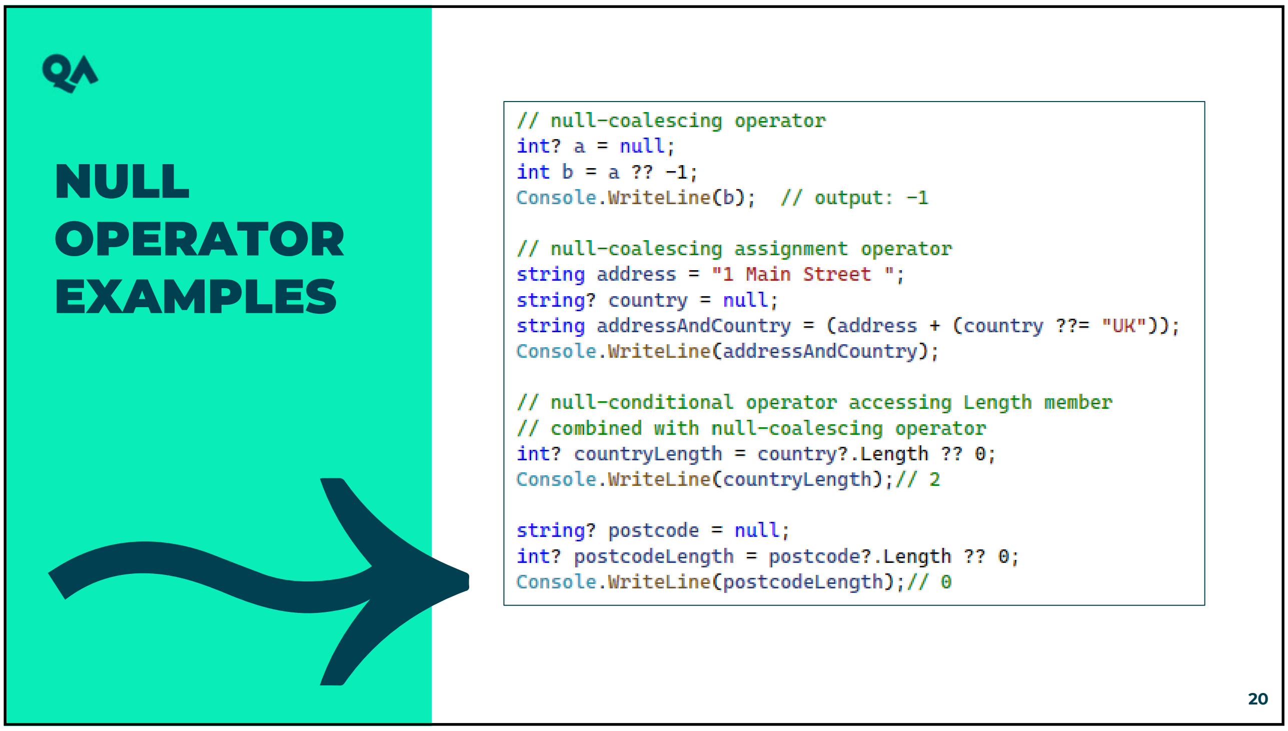


NULL CONDITIONAL OPERATOR ?.

- The **null-conditional operator** applies a member access operation to its operand only if that operand evaluates to non-null
- Otherwise, it returns null
- Often combined with the null-coalescing operator to return something other than null

19

19



NULL OPERATOR EXAMPLES

```
// null-coalescing operator
int? a = null;
int b = a ?? -1;
Console.WriteLine(b);  // output: -1

// null-coalescing assignment operator
string address = "1 Main Street ";
string? country = null;
string addressAndCountry = (address + (country ??= "UK"));
Console.WriteLine(addressAndCountry);

// null-conditional operator accessing Length member
// combined with null-coalescing operator
int? countryLength = country?.Length ?? 0;
Console.WriteLine(countryLength);// 2

string? postcode = null;
int? postcodeLength = postcode?.Length ?? 0;
Console.WriteLine(postcodeLength);// 0
```

20

20

SUMMARY

- Code blocks
- Select statements
- The if statement
- The switch statement
- The switch expression
- Switch case guards
- The ternary conditional operator
- Null-coalescing operators
- Null-conditional operator

21

21

ACTIVITY: Exercise 4

22

22