C# 10 Playbook

Control Flow and Loops

Simon Robinson
Software Developer

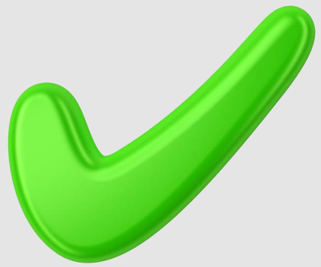
@TechieSimon www.SimonRobinson.com

The Playbook Course

This course teaches:

Problem Solving in C#

Each module solves problems focused on some part of the language



It does not teach

The C# language

I assume you already know C#!

Version Check

This course was created by using:

- C# 10
- .NET 6

Playbook Course Structure

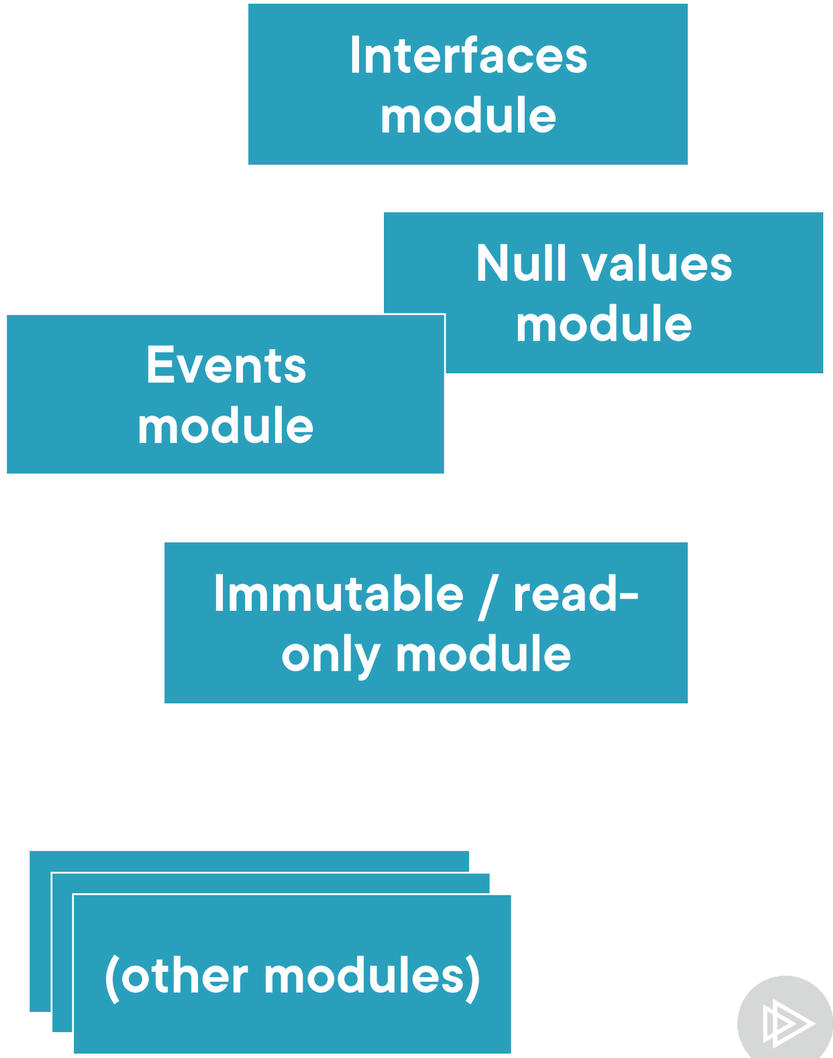
All modules are independent

There is no progression

Watch any modules in any order!

You can watch the course ‘from start to finish’ but you don’t have to

More modules might get added over time



Module Overview

Control flow using loops

- Simple topic for this 1st module:
- How do you decide which loop to use?
- Loop design

Loops in C#

```
for (/* setup */; /* condition */ ; /* continuation */ )
{ }
```

```
foreach (/* variable */ in /* sequence */)
{ }
```

```
while (/* condition */)
{ }
```

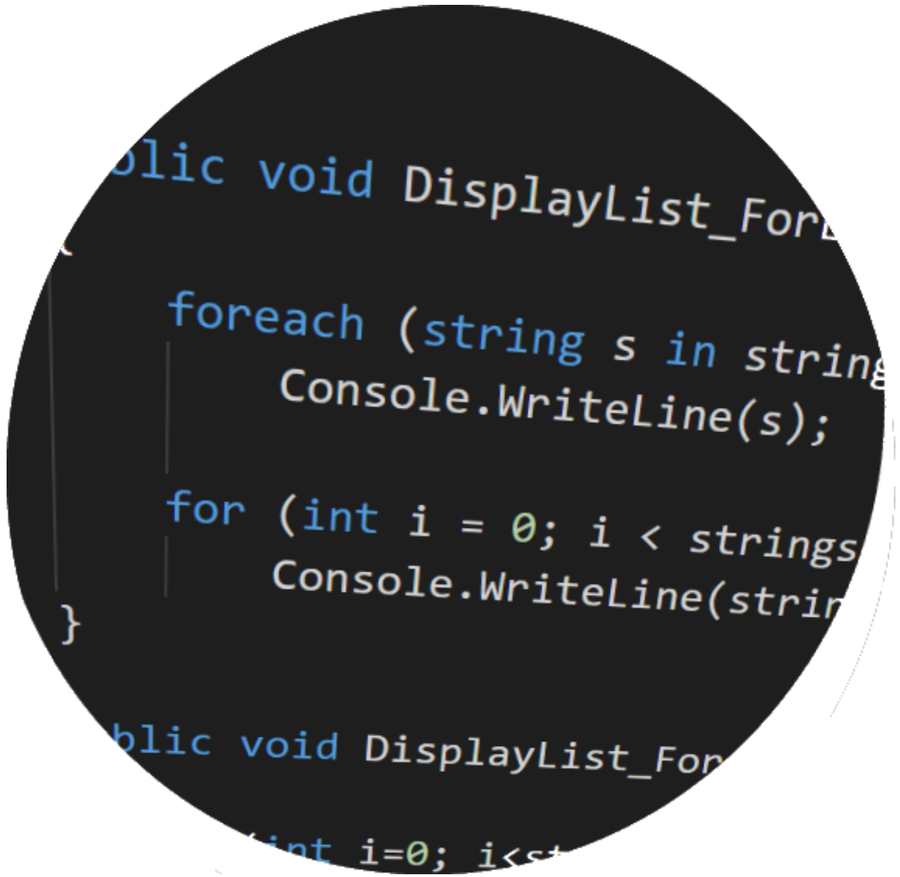
```
do
{ } while (/* condition */)
```

Which Loop Should You Use?

Demo

Iterating through sequences

- Comparing `for` and `foreach`
- Example: Display sequence of strings



To process a collection or sequence:

- Almost always prefer `foreach`
- `for` is only possible for an indexed collection
  - Prefer `for` if you need to use the index
  - Must use `for` to modify items

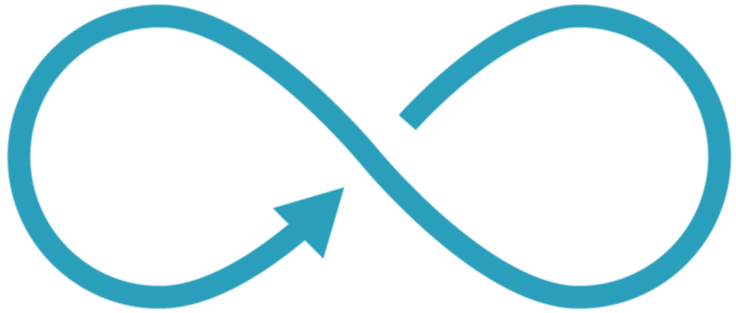
Loops without Sequences

Demo

Display directory tree above a file

- Compare solutions involving `while`, `for` and do loops
- Good practice for designing these kinds of loops

Loops: Good Practice



For loops with non-trivial iteration logic:

- Consider letting the loop just return results (for example with `yield return`)
- Let the caller deal with processing the loop results

These loops are completely equivalent

```
for(Setup(); ContinueCondition(); LoopChange())
{
    // loop body
};
```

```
Setup();
while (ContinueCondition())
{
    // loop body
    LoopChange();
}
```

Use whichever is clearest (But favor the for loop if there is an obvious indexer)

Summary

This is a playbook course

- Teaches solving problems

Loop design

- `foreach` for sequences
- `for` if you need to use the index of a list
- `do` if the loop must execute at least once
- `for` is equivalent to `while`
  - Use whichever gives simplest code in each situation

For complex loops:

- `yield return` to separate iteration logic from results processing