Complete C# Masterclass

Your progress

Share

Expression Bodied Members in C#

Expression Bodied Members in C#

Introduction

In this section of the course, we have covered different ways to define methods and properties in C#. However, we have not yet explored a concise way to define simple members using a special syntax called Expression Bodied Members.

Rather than using the standard block syntax for methods and properties, Expression Bodied Members provide a more compact and readable way to write certain types of class members. This approach improves code clarity and makes the intent of the method or property more obvious.

This article will introduce Expression Bodied Members, explain why they are useful, provide a real-world analogy, and walk through their implementation with examples.

1. What Are Expression Bodied Members?

An Expression Bodied Member is a shorthand syntax in C# that allows defining simple methods, properties, or even constructors using the lambda arrow (=>) instead of curly braces { }.

This feature is useful when a method or property contains only a single expression, making the code cleaner and reducing unnecessary syntax.

Analogy

Imagine writing a short note versus a long letter. If you want to quickly remind a friend about a meeting, you could write:

- ***"Meeting at 5 PM."***

Instead of writing:

- ***"Hello, just wanted to let you know that our meeting is scheduled for 5 PM. Looking forward to seeing you then."***

Both say the same thing, but the first one is more concise and direct. Expression Bodied Members in C# work similarly – they let you express simple logic in fewer lines of code without unnecessary boilerplate.

2. Declaring and Using Expression Bodied Members

Expression Bodied Members can be used in methods, properties, constructors, destructors, and even indexers.

Let's go through each case step by step.

2.1 Expression Bodied Methods

Normally, a method in C# is written like this:

```
1. class MathOperations
2. {
3.     public int Square(int number)
4.     {
5.         return number * number;
6.     }
7. }
```

Since this method only contains a single expression, we can simplify it using an Expression Bodied Member:

```
1. class MathOperations
2. {
3.     public int Square(int number) => number * number;
4. }
```

✔ **What changed?**

- **The curly braces { } were replaced with the => arrow.**
- **No need for the return keyword; the expression is automatically returned.**

This makes the code cleaner and more readable.

2.2 Expression Bodied Properties

If you have a read-only property, you can also use the same shorthand syntax.

Without Expression Bodied Members:

```
1. class Person
2. {
3.     private string name;
4.
5.     public Person(string name)
6.     {
7.         this.name = name;
8.     }
9.
10.    public string Name
11.    {
12.        get { return name; }
13.    }
14. }
```

With Expression Bodied Members:

```
1. class Person
2. {
3.     private string name;
4.
```

```
5.  public Person(string name)
6.  {
7.      this.name = name;
8.  }
9.
10. public string Name => name;
11. }
```

✔ **What changed?**

- **The get block { return name; } is replaced with => name.**
- **The property Name remains read-only but is now more concise.**

2.3 Expression Bodied Constructors

Constructors can also use Expression Bodied syntax if they only contain a single statement.

Without Expression Bodied Syntax:

```
1. class Logger
2. {
3.  private string message;
4.
5.  public Logger(string msg)
6.  {
7.      message = msg;
8.  }
9. }
```

With Expression Bodied Syntax:

```
1. class Logger
2. {
```

```
3.      private string message;
4.
5.      public Logger(string msg) => message = msg;
6.  }
```

✔ **What changed?**

- **The constructor { message = msg; } is replaced with => message = msg;.**
- **The functionality remains the same but is more concise.**

2.4 Expression Bodied Destructors

Expression Bodied Members also work for destructors, which are used for cleanup when an object is destroyed.

Without Expression Bodied Syntax:

```
1.  class FileHandler
2.  {
3.      ~FileHandler()
4.      {
5.          Console.WriteLine("Destructor called!");
6.      }
7.  }
```

With Expression Bodied Syntax:

```
1.  class FileHandler
2.  {
3.      ~FileHandler() => Console.WriteLine("Destructor called!");
4.  }
```

✔ **What changed?**

- **The destructor { Console.WriteLine(...); } is replaced with => Console.WriteLine(...);.**

2.5 Expression Bodied Indexers

Indexers allow objects to be indexed like arrays. Expression Bodied Members make indexers shorter when they contain a single return statement.

Without Expression Bodied Syntax:

```
1. class Collection
2. {
3.     private int[] numbers = { 1, 2, 3, 4, 5 };
4.
5.     public int this[int index]
6.     {
7.         get { return numbers[index]; }
8.     }
9. }
```

With Expression Bodied Syntax:

```
1. class Collection
2. {
3.     private int[] numbers = { 1, 2, 3, 4, 5 };
4.
5.     public int this[int index] => numbers[index];
6. }
```

✔ **What changed?**

- **The get block { return numbers[index]; } is replaced with => numbers[index].**

3. When to Use Expression Bodied Members?

✔ **Use them when methods or properties contain only a single expression.**

✔ **They are best for readability and conciseness.**

✔ **They are great for simple get-only properties, lightweight methods, constructors, destructors, and indexers.**

4. Best Practices and Common Mistakes

Best Practices

✔ **Use Expression Bodied syntax for simple operations.**

✔ **Keep code concise but still readable.**

✔ **Use it in conjunction with standard methods when necessary.**

Common Mistakes

✘ **Using it for complex logic – If a method has multiple lines of code, stick to regular methods instead of Expression Bodied syntax.**

```
1. // ✘ Not recommended for multiple expressions
2. public void Display()
3. {
4.     Console.WriteLine("Hello");
5.     Console.WriteLine("World");
6. }
```

```
1. // ✘ Incorrect usage of Expression Bodied syntax
2. public void Display() => Console.WriteLine("Hello"); Console.WriteLine("World"); // ERROR
```

✔ **Fix: Use standard method syntax for multi-line logic.**

5. Conclusion

Expression Bodied Members allow you to write concise, readable, and efficient code by reducing unnecessary syntax. While they aren't required for all methods or properties, they are an excellent tool for simplifying short, single-expression logic.

If you have any questions, feel free to ask in the Q&A section.
Happy coding! 🚀

Course content

Course content

Overview

Q&AQuestions and answers

Notes

Announcements

Reviews

Learning tools

Section 1: UPDATED: Introduction, Overview of Visual Studio, DataTypes And Variables

51 / 56 | 3hr 6min51 of 56 lectures completed3hr 6min

Section 2: UPDATED: Making Decisions

20 / 28 | 1hr 33min20 of 28 lectures completed1hr 33min

Section 3: UPDATED: Loops

22 / 24 | 1hr 37min22 of 24 lectures completed1hr 37min

Section 4: UPDATED: Functions and Methods

17 / 20 | 1hr 34min17 of 20 lectures completed1hr 34min

Section 5: UPDATED: Object Oriented Programming (OOP)

18 / 43 | 3hr 10min18 of 43 lectures completed3hr 10min

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

109. Objects Intro

2min

- **Lecture completed. Progress cannot be changed for this item.**

Play

110. Introduction To Classes And Objects

3min

- **Lecture completed. Progress cannot be changed for this item.**

Play

111. Creating our First own Class

8min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Play

112. Member Variable and Custom Constructor

7min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Play

113. Properties - Autogenerated - to protect our member variable

6min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Play

114. Defining how a property is set

8min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Play

115. Modifying the Get of our Property Part 1

7min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Play

116. Modifying the Get of our Property part 2

5min

- **Lecture completed. Progress cannot be changed for this item.**

Play

117. Having Multiple Constructors

7min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Play

118. Default Constructor and Use Cases

6min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Start

Quiz 12: Understanding Constructors

- **Lecture completed. Progress cannot be changed for this item.**

Play

119. Methods in Classes

7min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Play

120. Methods in Classes in more detail

8min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Start

121. Expression Bodied Members in C#

3min

- **Lecture completed. Progress cannot be changed for this item.**

Start

122. What are Inner Classes in C#?

3min

- **Lecture completed. Progress cannot be changed for this item.**

Start

123. Partial Classes and Methods

3min

- **Lecture completed. Progress cannot be changed for this item.**

Play

124. Optional Parameters

4min

Resources

- **Lecture completed. Progress cannot be changed for this item.**

Play

125. Named Parameters

3min

- **Lecture completed. Progress cannot be changed for this item.**

Start

126. Operator Overloading in C#

3min

- **Lecture incomplete. Progress cannot be changed for this item.**

Start

127. Passing Arguments by Value and by Reference

4min

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

128. Computed Properties and No Constructor

3min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

129. Static Methods

7min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Start

Coding Exercise 10: Using Static Methods

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

130. Static Fields

3min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Start

131. Static Keyword Considerations

3min

- **Lecture incomplete. Progress cannot be changed for this item.**

Start

132. The is Operator and the as Operator in C#

3min

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

133. Public and Private Keywords

5min

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

134. ID Key and readonly

7min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

135. Read Only Properties

3min

- **Lecture incomplete. Progress cannot be changed for this item.**

Start

Coding Exercise 11: Working with Read-Only Properties

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

136. Write Only Properties

5min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

137. Const and ReadOnly

5min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Start

Quiz 13: Working with Read-Only Properties

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

138. Quiz Project Introduction

4min

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

139. QuizApp - Question Class

5min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

140. Keyword This

3min

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

141. Displaying Questions

6min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

142. Displaying Answers, Console.Write and Console.ForegroundColor

7min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

143. Getting the UserInput and checking if it is right

6min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

144. Displaying Multiple Questions and if we are right or wrong

8min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Play

145. Displaying the Results

8min

Resources

- **Lecture incomplete. Progress cannot be changed for this item.**

Start

146. CHEATSHEET - Object Oriented Programming in C#

0min

- **Lecture incomplete. Progress cannot be changed for this item.**

Start

Coding Exercise 12: ADVANCED EXERCISE: Creating a Class with Properties and Methods

Section 6: UPDATED: Collections in C#

0 / 27 | 2hr 1min0 of 27 lectures completed2hr 1min

Section 7: UPDATED: Error Handling

0 / 14 | 45min0 of 14 lectures completed45min

Section 8: UPDATED: Inheritance

0 / 22 | 1hr 21min0 of 22 lectures completed1hr 21min

Section 9: UPDATED: Interfaces and Polymorphism

0 / 24 | 1hr 22min0 of 24 lectures completed1hr 22min

Section 10: UPDATED: Structs in C#

0 / 9 | 58min0 of 9 lectures completed58min

Section 11: UPDATED: Events and delegates

0 / 14 | 1hr 21min0 of 14 lectures completed1hr 21min

Section 12: UPDATED: Regular Expressions

0 / 11 | 43min0 of 11 lectures completed43min

Section 13: WPF - Windows Presentation Foundation

0 / 42 | 2hr 31min0 of 42 lectures completed2hr 31min

Section 14: WPF Project - Currency Converter - Part 1

0 / 8 | 1hr 14min0 of 8 lectures completed1hr 14min

Section 15: Using Databases With C#

0 / 12 | 2hr 2min0 of 12 lectures completed2hr 2min

Section 16: WPF Project - Currency Converter - Part 2

0 / 9 | 1hr 31min0 of 9 lectures completed1hr 31min

Section 17: Linq

0 / 13 | 2hr 18min0 of 13 lectures completed2hr 18min

Section 18: WPF Project - Currency Converter with GUI Database and API - Part 3

0 / 3 | 31min0 of 3 lectures completed31min

Section 19: The exercises for your coding interviews

0 / 4 | 5min0 of 4 lectures completed5min

Section 20: C# Clean Code

0 / 24 | 1hr 37min0 of 24 lectures completed1hr 37min

Section 21: C# Generics

0 / 18 | 1hr 38min0 of 18 lectures completed1hr 38min

Section 22: Threads

0 / 8 | 1hr 10min0 of 8 lectures completed1hr 10min

Section 23: Unit Testing - Test Driven Development TDD

0 / 36 | 3hr 24min0 of 36 lectures completed3hr 24min

Section 24: UNITY - Basics

0 / 16 | 1hr 35min0 of 16 lectures completed1hr 35min

Section 25: UNITY - Building the Game Pong with Unity

0 / 20 | 2hr 34min0 of 20 lectures completed2hr 34min

Section 26: UNITY - Building a Zig Zag Clone With Unity

0 / 18 | 2hr 11min0 of 18 lectures completed2hr 11min

Section 27: UNITY - Building a Fruit Ninja Clone With Unity

0 / 14 | 2hr 8min0 of 14 lectures completed2hr 8min

Section 28: Thank you for completing the course!

0 / 1 | 4min0 of 1 lecture completed4min

Section 29: Bonus

0 / 1 | 1min0 of 1 lecture completed1min

Teach the world online

Create an online video course, reach students across the globe, and earn money

Teach on Udemy

Top companies choose Udemy Business to build in-demand career skills.

Explore top skills and certifications

Certifications by Issuer

- Amazon Web Services (AWS) Certifications
- Six Sigma Certifications
- Microsoft Certifications
- Cisco Certifications
- Tableau Certifications
- See all Certifications

Web Development

- Web Development
- JavaScript
- React JS
- Angular
- Java

IT Certifications

- Amazon AWS
- AWS Certified Cloud Practitioner
- AZ-900: Microsoft Azure Fundamentals
- AWS Certified Solutions Architect - Associate
- Kubernetes

Leadership

- Leadership
- Management Skills

- Project Management
- Personal Productivity
- Emotional Intelligence

Certifications by Skill

- Cybersecurity Certification
- Project Management Certification
- Cloud Certification
- Data Analytics Certification
- HR Management Certification
- See all Certifications

Data Science

- Data Science
- Python
- Machine Learning
- ChatGPT
- Deep Learning

Communication

- Communication Skills
- Presentation Skills
- Public Speaking
- Writing
- PowerPoint

Business Analytics & Intelligence

- Microsoft Excel
- SQL
- Microsoft Power BI

- Data Analysis
- Business Analysis

About

- About us
- Careers
- Contact us
- Blog
- Investors

Discovery Udemy

- Get the app
- Teach on Udemy
- Plans and Pricing
- Affiliate
- Help and Support

Udemy for Business

- Udemy Business

Legal & Accessibility

- Accessibility statement
- Privacy policy
- Sitemap
- Terms