Function Overloading in C#

# ✅ What is Function Overloading in C#?

Function Overloading (also called Method Overloading) in C# allows you to define multiple methods with the same name in the same class, but with different parameter lists (type, number, or order of parameters). It's a way to provide flexibility and improve readability by allowing a method to perform similar operations depending on the input.

# 🔁 Overload Rules in C#

• Methods must differ by: number of parameters, or type of parameters, or order of parameters.

• ❌ Return type is NOT considered when overloading.

The compiler decides which method to call based on the method signature, which includes the method name and parameter types, not the return type.  
  
Example:  
int Add(int a, int b) { return a + b; }  
double Add(int a, int b) { return a + b; } // ❌ Compile-time error  
  
The call Add(2, 3) is ambiguous without considering parameter types.

# 🌍 Real-World Scenario: Function Overloading in Action

🎯 Scenario: Online Shopping – Calculating Discounts

You run an e-commerce platform and want to offer different kinds of discounts:  
- A fixed discount amount  
- A percentage-based discount  
- A combination of both

# 🧱 Implementation using Method Overloading

public class DiscountCalculator  
{  
 // 1. Fixed discount  
 public double ApplyDiscount(double amount, double fixedDiscount)  
 {  
 return amount - fixedDiscount;  
 }  
  
 // 2. Percentage discount  
 public double ApplyDiscount(double amount, int percent)  
 {  
 return amount - (amount \* percent / 100.0);  
 }  
  
 // 3. Combined fixed and percentage  
 public double ApplyDiscount(double amount, double fixedDiscount, int percent)  
 {  
 double discounted = amount - fixedDiscount;  
 return discounted - (discounted \* percent / 100.0);  
 }  
}

# 🧪 Usage

var calc = new DiscountCalculator();  
  
Console.WriteLine(calc.ApplyDiscount(1000, 100)); // Fixed discount  
Console.WriteLine(calc.ApplyDiscount(1000, 10)); // Percentage discount  
Console.WriteLine(calc.ApplyDiscount(1000, 100.0, 10)); // Combined

# ✅ Summary Table

|  |  |
| --- | --- |
| Concept | Explanation |
| Function Overloading | Same method name, different parameters |
| Return type only | ❌ Not allowed for overload resolution |
| Useful for | Method flexibility, cleaner APIs |
| Decided at | Compile-time based on method signature |