## Lambda Expressions

משתמשים בביטויי Lambda בכדי ליצור פונקציה אנונימית.

יכולת שהגיעה ב C# 3 לשפה, ומקצרת את התחביר עוד יותר מאשר ב Anonymous Methods (שהגיעו עם C# 2).

ביטויי למבדה מקובלים היום כתחביר מקוצר לפונקציות במגוון של שפות בתעשייה.

ליצירת ביטוי למבדה, נשתמש באופרטור [=>](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/lambda-operator)  בכדי להפריד בין רשימת הפרמטרים של הפונקציה לגוף הפונקציה. נקרא לזה מעתה אופרטור הלמבדה.

הפונקציות שנכתוב בתחביר Lambda מתחלקות לשניים:

1. ביטויי למבדה, המכילות רק ביטוי שמחזיר ערך, והמבנה שלהן הוא:

(input - parameters) => expression

2. משפטי למבדה, או פקודות למבדה, בהן גוף הפונקציה, הוא בלוק של קוד:

(input-parameters) => { <sequence-of-statements> }

בכדי ליצור ביטוי למבדה, נרשום בצד שמאל של אופרטור הלמבדה, את רשימת הפרמטרים (אם ישנם) ובצד ימין ביטוי או בלוק של קוד.

ניתן להמיר כל ביטוי למבדה ל delegate. סוג ה delegate נקבע על פי הפרמטרים שפונקציית הלמבדה הגדירה והערך המוחזר שלה.

בואו נתבונן בדוגמה הבאה:

delegate void Greet(string name);

Greet greet3 = (name)=> { Console.WriteLine("Shalom {0}", name); };

Greet greet4 = (name)=> { Console.WriteLine("Welcome {0}", name); };

הגדרנו delegate בשם Greet עם חתימה מסוימת. אנחנו מגדירים פונקציה אנונימית מקומית ומצביעים עליה עם משתנים מסוג ה delegate שהגדרנו. התחביר של הפונקציה קצרצר.

ודוגמה לביטוי למבדה:

delegate int MathDelegate (int x);

MathDelegate Power = number => number \* number;

Console.WriteLine(Power(5));//25

כפי שאנו רואים, כאשר אנחנו רק רוצים להחזיר ערך, ניתן להשתמש בביטוי, בלי להגדיר סוגריים מסולסלות {} , כלומר, בלי להגדיר בלוק של קוד, ויש לנו ביטוי למבדה.

זה שווה ערך לשורה הבאה:

MathDelegate Power = number => { return number\* number};

ניתן להבחין שבהגדרת הפרמטר number בצד שמאל של אופרטור הלמבדה, השמטתי את הסוגריים. ניתן להשמיט את הסוגריים של הפרמטרים כאשר יש לנו ביטוי למבדה שמכיל רק פרמטר אחד בודד. אם אין פרמטרים בפונקציה, נכתוב סוגריים ריקות ().

לדוגמה:

MyFirstDelegate first4 = () =>

{

Console.WriteLine("Another function called with same delegate");

};