Treating Functions as Objects, Big Time

Zoran Horvat
CEO AT CODING HELMET

@zoranh75   http://csharpmentor.com

Introducing Higher-order Functions

Higher-order functions

Receives one or more functions:

```
int GiveStars(
    Func<string, float> HotelRating);
```

Returns a function:

```
Func<float> MyHotelRating(
    int desiredStars);
```

Receives and returns functions:

```
Func<int> AnyHotelStars(
    Func<string, float> authority);
```

First-order functions

Receives common values:

```
void GiveStars(string hotelName);
```
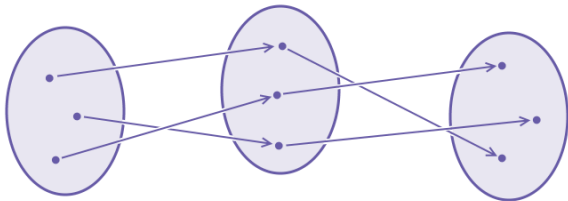
Returns common values:

```
float RateHotel(string name);
```

Receives and/or returns objects:

```
IStars GiveStars(IRatings ratings);
```

Function composition:
$g \circ f$

$$(g \circ f)(x) = g(f(x))$$

F#

```
x |> f |> g
```

First to execute   Last to execute

C#

```
objB.g(objA.f(x))
```

Last to execute   First to execute

C# fluent interface

```
x.f(this).g(this)
```

Partial Function Application in Use
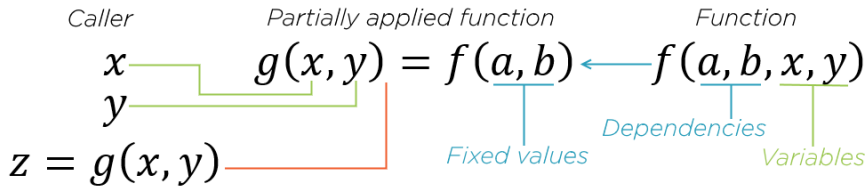
Simplifying the caller

Caller *Partially applied function* *Function*

$$x \quad g(x, y) = f(a, b) \leftarrow f(a, b, x, y)$$
$$y$$

$$z = g(x, y)$$

Fixed values *Dependencies* *Variables*

Application Lifecycle Management (ALM)

Caller *Existing function*

$$f(x, y)$$
$$x$$
$$y$$

Partial Function Application in Use

Simplifying the caller

Caller   *Partially applied function*   *Function*

$$x$$
$$y$$

$$g(x, y) = f(a, b) \longleftarrow f(a, b, x, y)$$

$$z = g(x, y)$$

Fixed values   *Dependencies*   *Variables*

Application Lifecycle Management (ALM)

Caller   *Existing function*   *New function*

$$f(x, y)$$

$$x$$
$$y$$

$$f(x, y) = g(p) \longleftarrow g(x, y, p)$$

Partial Function Application in Use

Simplifying the caller

Caller   Partially applied function   Function

$$x$$
$$y$$
$$g(x, y) = f(a, b) \leftarrow f(a, b, x, y)$$

$$z = g(x, y)$$

Fixed values

Dependencies

Variables

Application Lifecycle Management (ALM)

Caller   Existing function   New function

$$x$$
$$y$$

$$f(x, y)$$
$$f(x, y) = g(p) \leftarrow g(x, y, p)$$

$$p$$
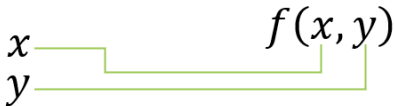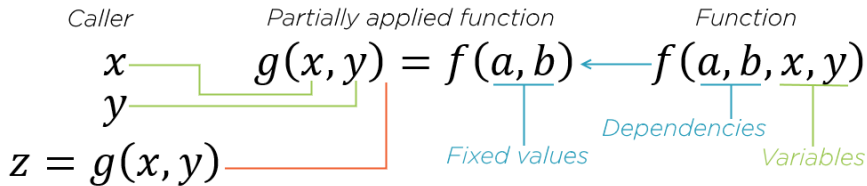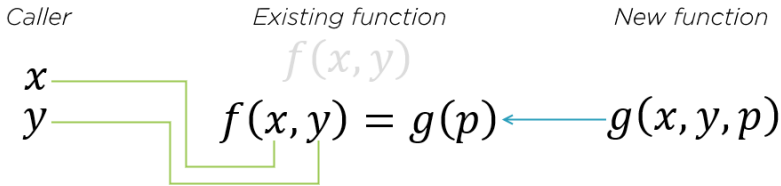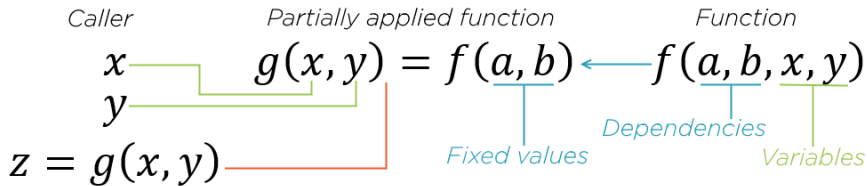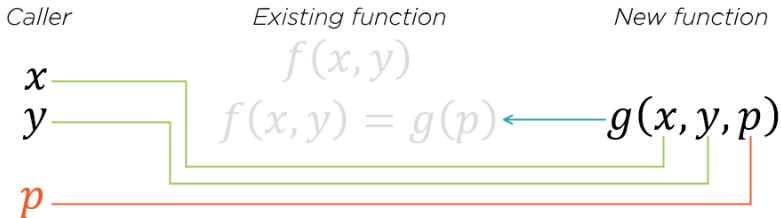
Summary

Higher-order functions
- A function receives a delegate
- Delegate can have dependencies
  - Types
  - Other functions
- Dependencies not visible to the consumer

Higher-order template functions
- Receive a delegate to fill the blanks
  - E.g. tax calculation delegate
- Template function knows *when* to call the delegate

Summary

Function composition

- Supported by functional languages
- Chained calls in object-oriented code
- Use extension methods that are chainable

Summary

Partial function application
- Fix values of one or more arguments
- Results in a new function
- Expects the remaining (not fixed) arguments

No native support in C#
- Construct a Func delegate
- Overload a function
  - Receive shorter argument list
  - Internally call the larger function

Next module:
Attaining Truly Functional Functions