
Put Some
FP
In Your
OOP

2017 Cincinnati Day of Agile & Cincy.Develop(); Sponsors

Diamond

Gold

Silver

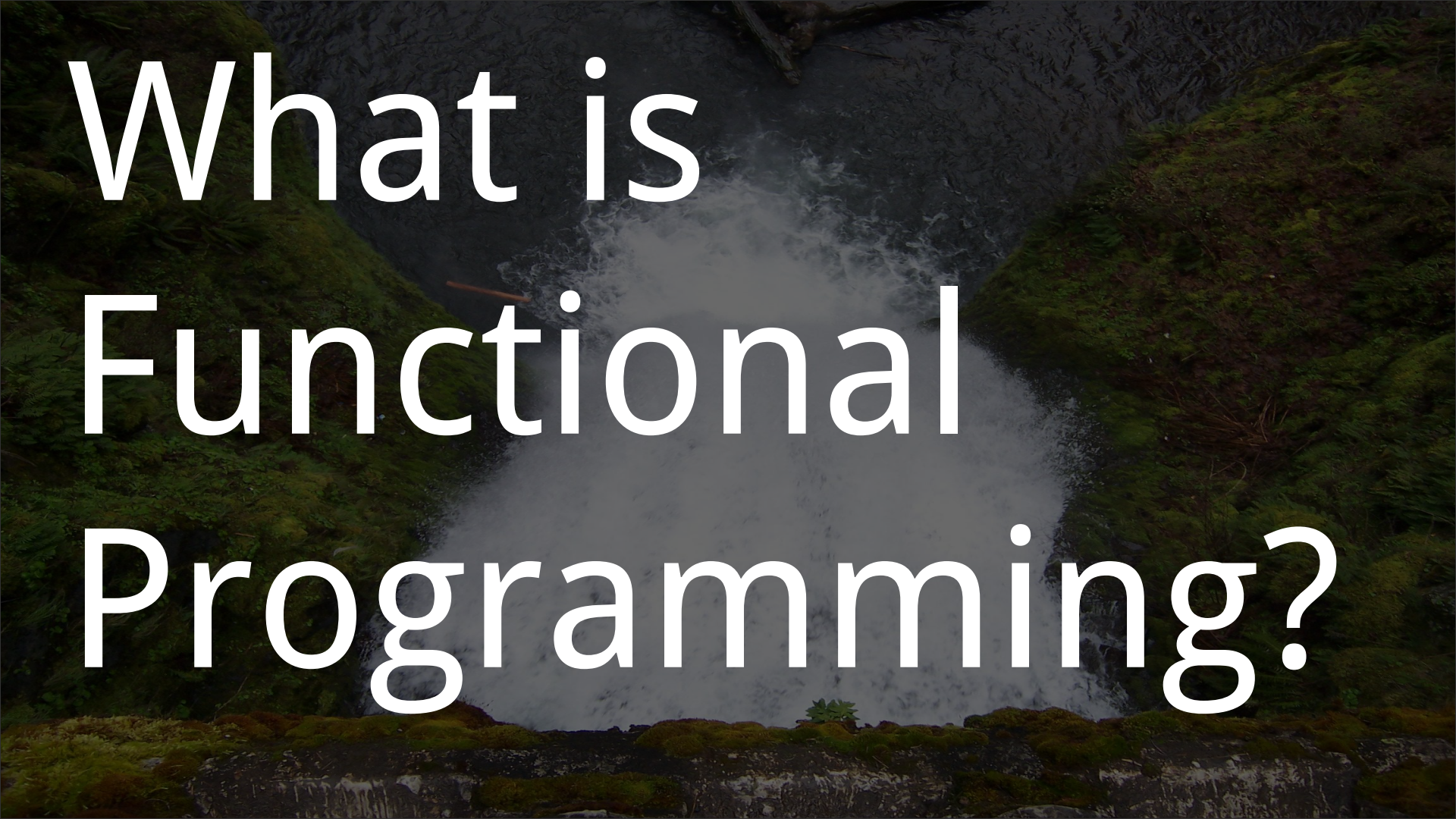
Michael Richardson

@anaccidentaldev

@accidentaldeveloper


Put Some
FP
In Your
OOP


Functional
Programming


What is
Functional
Programming?


FP is about
functions


FP is about
restrictions

Why Functional?

It’s Easier

It’s Easier

- Write
- Read
- Debug
- Test

Why mix FP and OOP?

You (probably) already know OO

Your company & coworkers

Functional concepts integrate easily

Modern languages are doing it!

2 Concepts:

- Immutability
- Pure Functions

Immutability

What is immutability?

Immutable objects can’t be changed (mutated)

State passed into the constructor

```
class Foo()
{
    Foo(myState)
    {
    }
}
```

No setters/mutating methods

```
class Foo()
{
    public mutateFoo(string buzz)
    {
        this.bar = buzz;
    }
}
```

Don’t expose mutable state

```
class Foo()
{
    public MutableObject MutateMe { get; }
}
```

Why use immutable objects?

Why use immutable objects?

Easier to reason about!

Never have to ask:
“Did I set this property?”
“When did I set it?”
“Is this the original value?”

Eliminate entire classes of bugs

```
public Recipe CreateRecipe() {
    var recipe = new Recipe();
    recipe.RecipeId = "12345";
    recipe.Title = "Poison Apple";
    return recipe;
}
```

```
public Recipe CreateRecipe() {
    var recipe = new Recipe();
    recipe.RecipeId = "12345";
    recipe.Title = "Poison Apple";
    return recipe;
}

public class Recipe {
    string RecipeId { get; set; }
    string Title { get; set; }
    string Description { get; set; }
}
```

```
public Recipe CreateRecipe() {
    var recipe = new Recipe("12345", "Poison Apple");
}

public class Recipe {
    public Recipe(string recipeId, string title, string description) {
        RecipeId = recipeId;
        Title = title;
        Description = description;
    }
    string RecipeId { get; }
    string Title { get; }
    string Description { get; }
}
```

Using Immutabilty

Default to Immutable

```
class Foo {
    public string Bar { get; set; }
}
```

Default to Immutable

```
class Foo {
    public string Bar { get; set; }
}
```

Choose to not Mutate

Why is mutation dangerous?

```
class Recipe {
    string RecipeId { get; set; }
    string Title { get; set; }
}

class RecipesService {
    public static List<Recipe> Recipes = GetRecipes();
    private static List<Recipe> GetRecipes () {
        // Deserialize json from file
        return recipes;
    }
}
```

```
class Recipe {
    string RecipeId { get; set; }
    string Title { get; set; }
    bool IsFavorite { get; set; }
}

List<Recipe> GetRecipesForUser(List<string> favoriteRecipeIds) {
    var recipes = RecipesService.Recipes;
    foreach (var recipe in recipes) {
        recipe.IsFavorite = favoriteRecipeIds.Contains(recipe.RecipeId);
    }

    return recipes;
}
```

```
class Recipe {
    string RecipeId { get; set; }
    string Title { get; set; }
    bool IsFavorite { get; set; }
}

List<Recipe> GetRecipesForAnonymousUser() {
    return RecipesService.Recipes;
}
```

```
class Recipe {
    public Recipe(string recipeId, string Title) {
        RecipeId = recipeId;
        Title = title;
    }
    string RecipeId { get; }
    string Title { get; }
}
```

```
public class RecipeModel {
    public RecipeModel(Recipe recipe, bool isFavorite) {
        RecipeId = recipe.RecipeId;
        Title = recipe.Title;
        IsFavorite = isFavorite;
    }
    string RecipeId { get; }
    string Title { get; }
    bool IsFavorite { get; }
}
```

```
IEnumerable<RecipeModel> GetRecipesForLoggedInUser(
        List<string> favoriteRecipeIds
) {
    var recipes = RecipesService.Recipes;
    foreach (var recipe in recipes) {
        var isFavorite = favoriteRecipeIds.Contains(recipe.RecipeId);
        yield return new RecipeModel(recipe, isFavorite);
    }
}
```

Immutability:

- Immutable objects are limiting
- Lead to better coding practices
- Avoid bugs
- Immutable is the best default
- Treat mutable objects as immutable (when it makes sense)

Pure Functions

What are pure functions?

$f(x) = x + 1$

x	f(x)
1	2
2	3
5	6

Identical input

=>

Identical output

```
int AddOne(int x){
    return x + 1;
}
```

No side effects

Why use pure functions?

```
public bool IsExpired(Coupon coupon) {
    return coupon.ExpirationDate > DateTime.Today;
}
```

```
public bool IsExpired(Coupon coupon) {
    return coupon.ExpirationDate > DateTime.Today;
}

public bool IsExpiredAtDate(Coupon coupon, DateTime
date) {
    return expirable.ExpirationDate > date.Date;
}
```

It’s Easier

- Write
- Read
- Debug
- Test

Pure functions

Immutability

Using Pure Functions

Primary Abstraction

OOP => class

FP => function

Map/
Filter/
Reduce

What is Map/Filter/Reduce?

Work with collections without mutation

Different names in different languages

.NET => LINQ

Java => Stream API

JS => added to Array in ES5

Why use map, filter, reduce?

Declarative Syntax

Terse, easy to read

```
Dim selectedTickets = db.Tickets.Where(GetWhereExpression())
Dim foo = From t In selectedTickets
          Group By l = t.Location
          Into lTickets = Group
          Let count = lTickets.Count()
          Order By count Descending
          Select New With {.LocationName = l.LocationName, .Count = count}
          Take 10
```

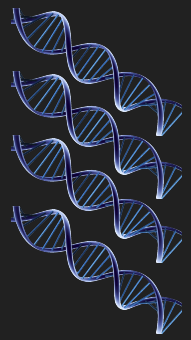
Gateway drug

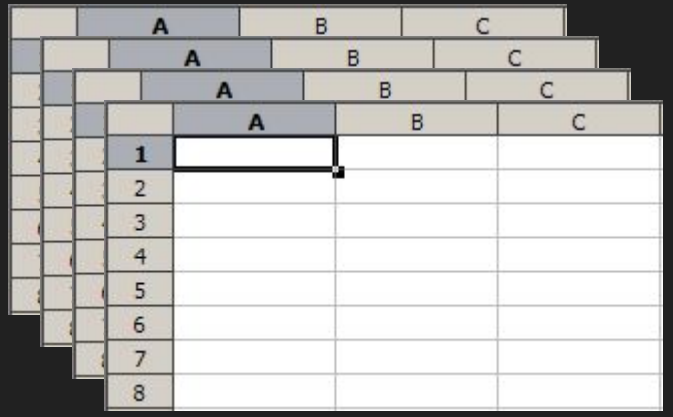
Partial Application

```
const getProperty =
    (propertyName, object) => object[propertyName];

const getTheDude =
    (object) => getProperty("The Dude", object);

const theDude =
    getTheDude({ "The Dude": "Jeffery Lebowski" })
```

Partial Application

Real Scenario




A
B
C
1
2
3
4
5
6
7
8

Configuration

ColumnA: GetGeneticMarker(type, id, format, data),
ColumnB: GetGeneticMarker(type, id, format, data),
ColumnC: GetGeneticMarker(type, id, format, data),

Configuration

↓

Partial Application

↓

```
ColumnA: columnAFunc(data),
ColumnB: columnBFunc(data),
ColumnC: columnCFunc(data),
```

Lenses

Closing Thoughts

Shift your mental model

(in small increments)

Skeptical?
Try it out, see what happens

Try out FP languages

Thank You!

Michael Richardson

@anaccidentaldev

@accidentaldeveloper