Introduction to Functional Programming

Jeremy Berglund

What is Functional Programming?

“a programming paradigm… that treats computation as the evaluation of mathematical functions and avoids changing state and mutable data” [Wikipedia]

What is Functional Programming?

"a programming paradigm… that treats computation as the evaluation of mathematical functions and avoids changing state and mutable data" [Wikipedia]

- Function composition
- Pure functions
- Declarative expressions
- Immutable data
- Monoids
- Recursion
- Functors
- Monads
- First class functions
- Currying
- Referential transparency
- Higher-order-functions

What is Functional Programming?

Alonzo Church (1903 – 1995)

What is Functional Programming?

Lambda Calculus

- A universal model of computation based on computable functions
- Functionally equivalent to the Turing machine model
- Centered around function composition

What is Functional Programming?

Javascript

```javascript
function add(x, y) {
    return x + y;
}
```

What is Functional Programming?

Lambda Calculus simplifies functions

What is Functional Programming?

Lambda Calculus simplifies functions

1. Functions are anonymous

```
(x, y) => x + y
```

What is Functional Programming?

Lambda Calculus simplifies functions

1. Functions are anonymous
2. Functions are unary

```
(x, y) => x + y
```

```
x => y => x + y
```

What is Functional Programming?

Lambda Calculus simplifies functions

1. Functions are anonymous
2. Functions are unary
3. Functions are first class

```
(x, y) => x + y
```

```
x => y => x + y
```

What is Functional Programming?

```
function add(x, y) {
    return x + y;
}

add(2, 6)
```

```
((x, y) => x + y)(2, 6)
```

```
((x => y => x + y)(2)(6)
```

What is Functional Programming?

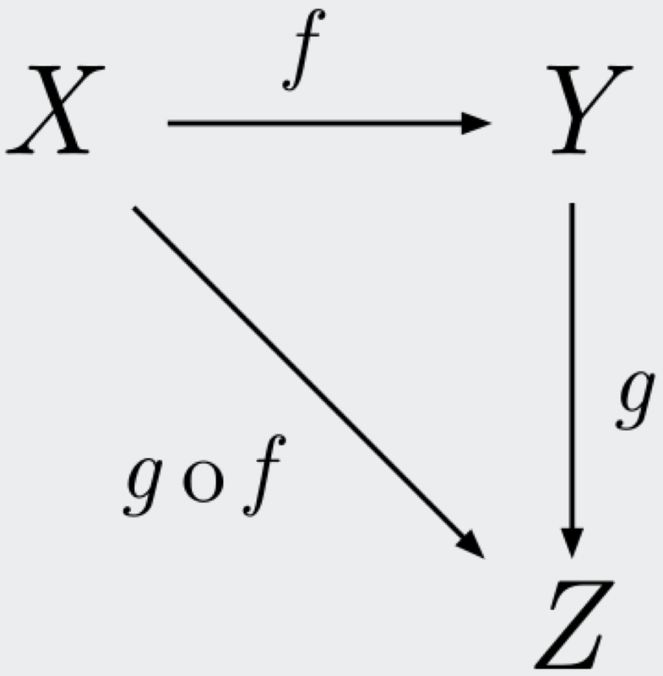
Category Theory

“formalizes mathematical structure and its concepts in terms of a labeled directed graph called a category, whose nodes are objects, and whose labeled directed edges are called arrows (or morphisms)” [Wikipedia]

What is Functional Programming?

Category Theory

- Categories are an algebraic structure for modelling objects and their relationships



What is Functional Programming?

Category Theory

- **Functions**
    - Every arrow (morphism) in a Category is a mapping
    - f(X) => Y

$$X \xrightarrow{f} Y$$

What is Functional Programming?

Function Composition

- Associative – h∘(g∘f) == (h∘g) ∘f
- Every object has an identity morphism I(x) => x
- Every time you chain functions together, you are performing function composition

What is Functional Programming?

Function Composition

```
const f = n => n + 1;
const g = n => n + 2;
const h = n => n * 2;

const compose = n => {
    return h(g(f(n)));
};

compose(10);
// 26
```

What is Functional Programming?

Function Composition

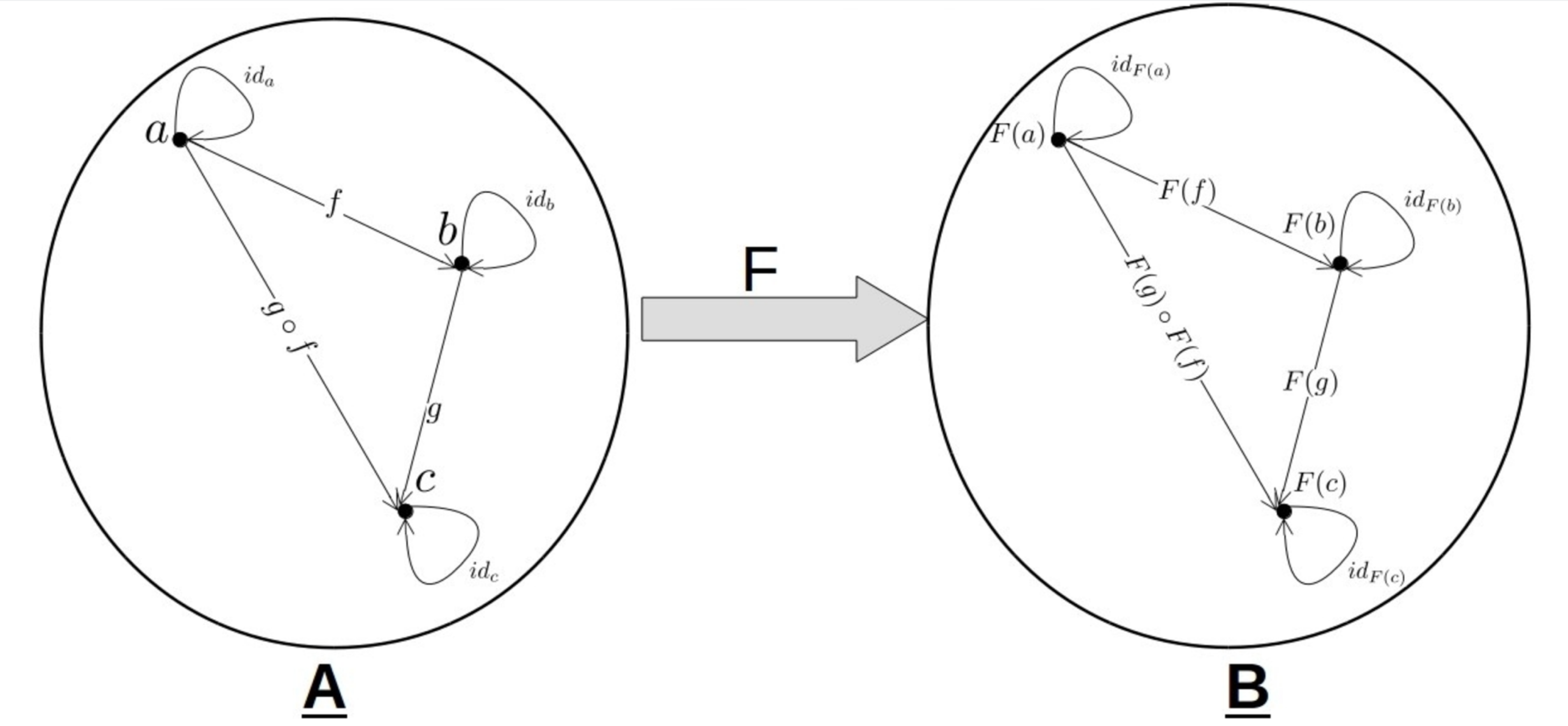
```
import _ from 'lodash/fp';

const f = n => n + 1;
const g = n => n + 2;
const h = n => n * 2;

_.compose(h, g, f)(10);
// 26
```

What is Functional Programming?

Category Theory

- **Functions**
  - Every arrow (morphism) in a Category is a mapping
  - f(a) => b
- **Functors**
  - Functors are morphisms between categories
  - A.K.A. *mappable*
  - F(A) => F(B)
  - Preserves the structure of the mapped category
  - $F(f \circ g) = F(f) \circ F(g)$

What is Functional Programming?



A Functor from A to B.

http://nikgrozev.com/2016/03/14/functional-programming-and-category-theory-part-1-categories-and-functors/

What is Functional Programming?

Category Theory

- **Functions**
  - Every arrow (morphism) in a Category is a mapping
  - f(a) => b
- **Functors**
  - Functors are morphisms between categories
  - A.K.A. *mappable*
  - F(A) => F(B)
  - Preserves the structure of the mapped category
  - $F(f \circ g) = F(f) \circ F(g)$
- **Monads**
  - Map and flatten
  - M(M(a)) => M(b)
  - Promises in JavaScript

What is Functional Programming?

Pure functions

- **Given the same input will always result in the same output**
- **Has no side effects**

What is Functional Programming?

Pure functions

- Given the same input will always result in the same output
- Has no side effects
- Important for function composition

What is Functional Programming?

Pure functions

- Given the same input will always result in the same output
- Has no side effects
- Important for function composition

Impure

```
let x = 10;

function addX(y) {
    x = x + y;
}

addX(5);
// x === 15
```

Pure

```
const add = (x, y) => x + y;
const x = add(10, 5);
// x === 15
```

What is Functional Programming?

Currying

```
const add = (x, y) => x + y;
```

What is Functional Programming?

Currying

```
import _ from 'lodash';

const add = _.curry((x, y) => x + y);
```

↓

```
const add = x => {
    return y => {
        return x + y;
    };
};
```

What is Functional Programming?

Currying

```
import _ from 'lodash';

const add = _.curry((x, y) => x + y);
```

↓

```
const add = x => y => x + y;
```

What is Functional Programming?

Currying

```
import _ from 'lodash';

const add = _.curry((x, y) => x + y);

const increment = add(1);

increment(10);
// 11
```

Practical FP

- Most for-loop operations can be accomplished using map, filter, or reduce

Practical FP

```
function getEven(arr) {
    let evens = [];

    for(let i = 0; i < arr.length; i++) {
        if(arr[i] % 2 === 0) {
            evens.push(arr[i])
        }
    }

    return evens;
}

getEven([1, 2, 4, 9]);
// [2, 4]
```

Practical FP

Using Array.filter

```
const getEven = arr => arr.filter(val => val % 2 === 0);

getEven([1, 2, 4, 9]);
// [2, 4]
```

Practical FP

Using lodash.filter

```
import _ from 'lodash/fp';

const getEven = _.filter(val => val % 2 === 0);

getEven([1, 2, 4, 9]);
// [2, 4]
```

Practical FP

```
function double(arr) {
    let doubled = [];

    for(let i = 0; i < arr.length; i++) {
        doubled[i] = arr[i] * 2;
    }

    return doubled;
}

double([2, 4, 6]);
// [4, 8, 12]
```

Practical FP

```
const double = arr => arr.map(val => val * 2);

double([2, 4, 6]);
// [4, 8, 12]
```

Practical FP

- **Point-free**
  - Never mentioning the data the function is operating on
  - If your code is point-free, it is probably composable and pure

Why use Functional Programming?

- **State management**
- **Declarative expressions**
    - Readable
    - Maintainable
    - Testable
    - Resuable
- **Less room for error**
- **Function composition**

How to get started with FP

- **JavaScript is an excellent starting point**
- **Lodash/Rambda libraries**
    - List manipulation
    - Utility functions
    - Composition
- **You can do as much or as little FP as you want**

How to get started with FP

- Mostly Adequate Guide to FP
- Lodash
- Ramda
- Eric Elliott's Function Programming Series