solirius | Engineering

Introduction to Functional Programming

What makes programming functional?

Imperative code

```
function countGreaterThan10(numbers) {
  let total = 0;

  for (const num of numbers) {
    if (num > 10) {
      total++;
    }
  }

  return total;
}
```

Functional code

```
function countGreaterThan10(numbers) {
  return numbers.filter(n => n > 10).length;
}
```

Higher Order Functions

- A function is data
- Therefore it can be used like data
- So a function can be passed to a function
- Let's talk about map...

Imperative code

```
function sqrtAll(values) {
  for (const i in values) {
    values[i] = Math.sqrt(value);
  }

  return values;
}
```

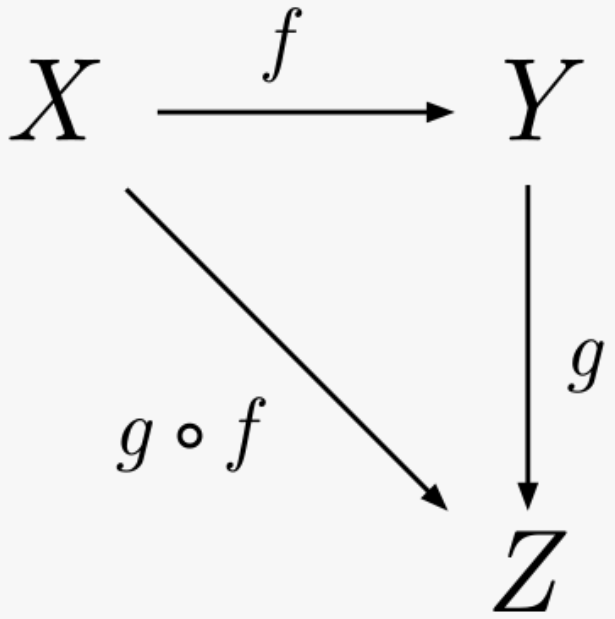
Functional code

```
function sqrtAll(values) {
  return values.map(x => Math.sqrt(x));
}
```

What makes a function pure?

Category Theory and Composition

- Given a function f that maps X -> Y
- And a function g that maps Y -> Z
- We know that applying f and g will turn X -> Z



Composition

```
const result = countGreaterThan10(sqrtAll(filterEven([ 1, 5, 10, 15, 20, 25])));

const numEvenAndSqrtGreaterThan10  = compose(
  filterEven,
  sqrtAll,
  countGreaterThan10
);

const result = numEvenAndSqrtGreaterThan10([ 1, 5, 10, 15, 20, 25]);

function compose(...fns) {
  return function (arg) {
    return fns.reduceRight((lastResult, fn) => fn(lastResult), arg);
  }
}
```

ESNext Magic

```
const result = [1, 5, 10, 15, 20, 25]
  |> filterEven
  |> sqrtAll
  |> countGreaterThan10;
```

Currying and Partial Application

```
function addToAll(amount, numbers) {
  return numbers.map(n => n + amount);
}

const add100ToAll = curry(addToAll, 100);
const result = [1, 5, 10, 15, 20, 25]
  |> add100ToAll
  |> filterEven
  |> sqrtAll
  |> countGreaterThan10;

function curry(fn, arguments) {
  return function (remainingArguments) {
    return fn(...arguments, ...remainingArguments)
  }
}
```

Go forth and be functional

- Higher Order Functions
- Pure Functions
- Category Theory
- Composition
- Curring
- ... and monads

ljn.io/posts/introduction-to-functional-programming