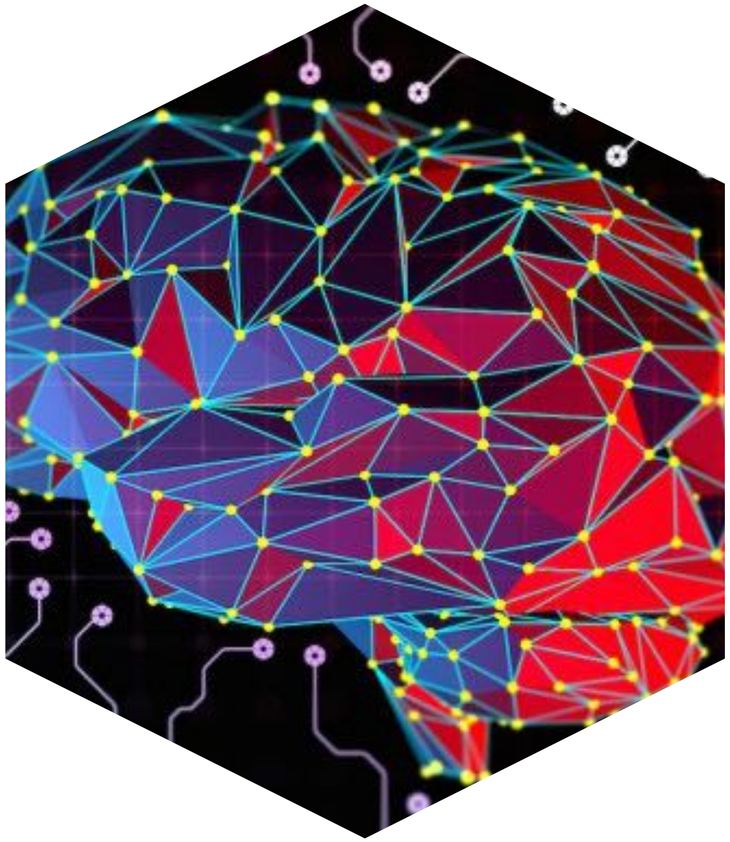
PROGRAMMING
PARADIGMS &

FUNCTIONAL PROGRAMMING



Jan P.

GitHub.com/fafk
https://fafk.github.io

JavaScript, Rust, Java, LISP & crypto
patomil@gmail.com

OVERVIEW

- Programming paradigms
- Functional programming motivation
- The magic of “this”
- Higher order functions, apply, call, arguments, …rest
- forEach, filter, map, reduce
- Recursion

There are different ways to think about computation — different paradigms.

All programs are translated into machine code that is closest to the imperative paradigm.

The goal is always the same: to translate some input into output.
But how we think about the necessary steps and how we represent them in code can change!
In different words, there are many ways to represent an algorithm.

Data Structures & Algorithms

algorithm: a step-by-step procedure for solving a problem

Computation:
input data
+
instructions (algorithm)
=>
output data

A computer program is a set of instructions (an algorithm) operating on data structures.

We can make our program run efficiently
by choosing the right data structures and algorithms.

Wide-spread paradigms

Imperative

Language: C

Model: Turing machine.

```
Let i = 0;
while (i < 10) {
  console.log(i);
  i++;
}
```

Declarative

Language: SQL

Describe what you need and let the interpreter figure out how to get it.

Used for working with databases.

```
SELECT SUM() FROM range(1..10);
```

Object Oriented

Language: Java

Model the world as a set of objects communicating with each other.

```
class Counter {
  i = 0;

  public inc() {
    this.i = this.i + 1;
  }
}
```

Examples

Imperative

```
const data = [1,2,3,4,5,6,7,8,9,10];
let i = 0;
let sum = 0;

while (i < data.length) {
  sum = sum + data[i];
  i++;
}

console.log(sum)
```

Declarative

```
SELECT sum() FROM range(1..10);
```

Object Oriented

```
class Accumulator {


  constructor(range) {
    this.range = range;
  }


  getSum() {
    return this.range.reduce((acc, i) => acc += i, 0);
  }
}


const acc = new Accumulator([1,2,3,4,5,6,7,8,9,10]);
console.log(acc.getSum());
```

What are idioms

Why functional programming?

- 👉 An addition to your programming toolbox
- 👉 Improves code readability
  - 👉 Shorter, more expressive idioms

What are idioms

Why functional programming?

- 👉 Improves code maintainability
  - 👉 Change in a **pure function** changes nothing but that function
  - 👉 A pure function immediately becomes an independent “module”

What are idioms

Why functional programming?

- 👉 Conceptually nice!
  - 👉 There are languages where everything is an expression
    - 👉 But there is implications: you need to handle all the cases of ifs, no exceptions etc.

Function Context — “this”

The problem: https://codepen.io/gaearon/pen/xEmzGg?editors=0010

👉 A function get its “context” (the value of “this”) from its caller, unless explicitly bound
👉 Arrow functions get its context from its lexical scope
👉 Context can be changed by using apply() and call()
👉 Sometimes “context” is called “scope”

https://repl.it/@fafk/functionContext
https://repl.it/@fafk/applyCall

“arguments”, “...” operator

Access arguments without names.
Helpful for function with with a variable number of arguments.

https://repl.it/@fafk/variableArgs

Arrays... functionally!

- 👉 Array.forEach() — iterate over
- 👉 Array.map() — create a new array based on the old one
- 👉 Array.reduce() — create one value from the array
- 👉 Recursion 🤯

https://repl.it/@fafk/CityBikesFunctional

Thanks!

Any questions?

You can find me at patomil@gmail.com