JavaCro'16

Are you ready for functional programming JavaScript vs. Scala

Elizabeta Ilievska

Goran Kopevski

Agenda

- Why functional programming?
- Imperative vs Functional programming
- Why Scala? Why Javascript?
- High Order Functions
- Closures
- Currying
- Tail recursion
- Immutability
- Lazy evaluation & Memoization
- Benefits & Conclusion

Why functional programming?

- Concurrency
- Parallelism
- Spend more time on thinking about the implications of results rather than how to generate them
- Limits of silicon technology reached
- Safe ways of programming
- Safe reuse of subprograms
- Generic routines

and the list can go to infinity ….

Imperative vs. Functional programming

Imperative programming describes the steps to solve a problem.

Functional programming focuses on what needs to be done rather the steps that need to be taken.

Characteristic	Imperative approach	Functional approach
Programmer focus	How to perform tasks (algorithms) and how to track changes in state.	What information is desired and what transformations are required.
State changes	Important.	Non-existent.
Order of execution	Important.	Low importance.
Primary flow control	Loops, conditionals, and function (method) calls.	Function calls, including recursion.
Primary manipulation unit	Instances of structures or classes.	Functions as first-class objects and data collections.

Imperative vs. Functional programming

Imperative

```
public static int sum(int[] list) {
    int currentSum = 0;
    for (int i = 0; i < list.length; i++) {
        currentSum = currentSum + list[i];
    }
    return currentSum;
}
```

Functional

```
public static int sumFunctional(int[] list) {
    if (list.length == 0)
        return 0;
    else {
        int[] tail = Arrays.copyOfRange(list, 1, list.length);
        return list[0] + sumFunctional(tail);
    }

}
```

Why Scala?

- Cuts down the boilerplate
- Tightly fuses object-oriented and functional programming concepts
- Runs on the JVM and interoperates with Java :)
- Excellent basis for concurrent, parallel and distributed computing
- Deliver things faster with less code

Why Javascript?

Two reasons according to Douglas Crockford:

- You don't have a choice
- Javascript is really good

Higher Order Functions

- Functions can take other functions as arguments
- Functions are first-order citizens
  - Can be assigned to variables, array entries and properties of other objects
  - Can be arguments/return values from other functions
- Functions can posses properties that can be dynamically created and assigned
- Functions can be created via literals

Closures

- One of the main features of each functional languages
- The scope created when function is declared
- Allows the functions to access and manipulate external variables

Possibilities?
endless….

Currying

- Produce a new function by combining a function and an argument
- Two ways of using currying::
  - By defining functions that return other functions.
  - By defining a function with multiple parameter lists.
- Practical usage:
  - var arguments
  - Adapter pattern

Tail recursion

Feature that each functional language must have!

The bad news for JS: stack overflow

But in reality, stack overflow in JS = you are doing it wrong

```
def nonTailFactorial(number: Long): Long =
  if (number == 0l)
    1l
  else
    nonTailFactorial(number - 1) * number
}
```

Immutability

Immutable object = unchangeable object

Main benefits:

- **Concurrency**

  The Free Lunch Is Over - A Fundamental Turn Toward Concurrency in Software.
- Debugging
- Unit testing

Avoiding mutable state means programming functionally

Lazy evaluation & Memoization

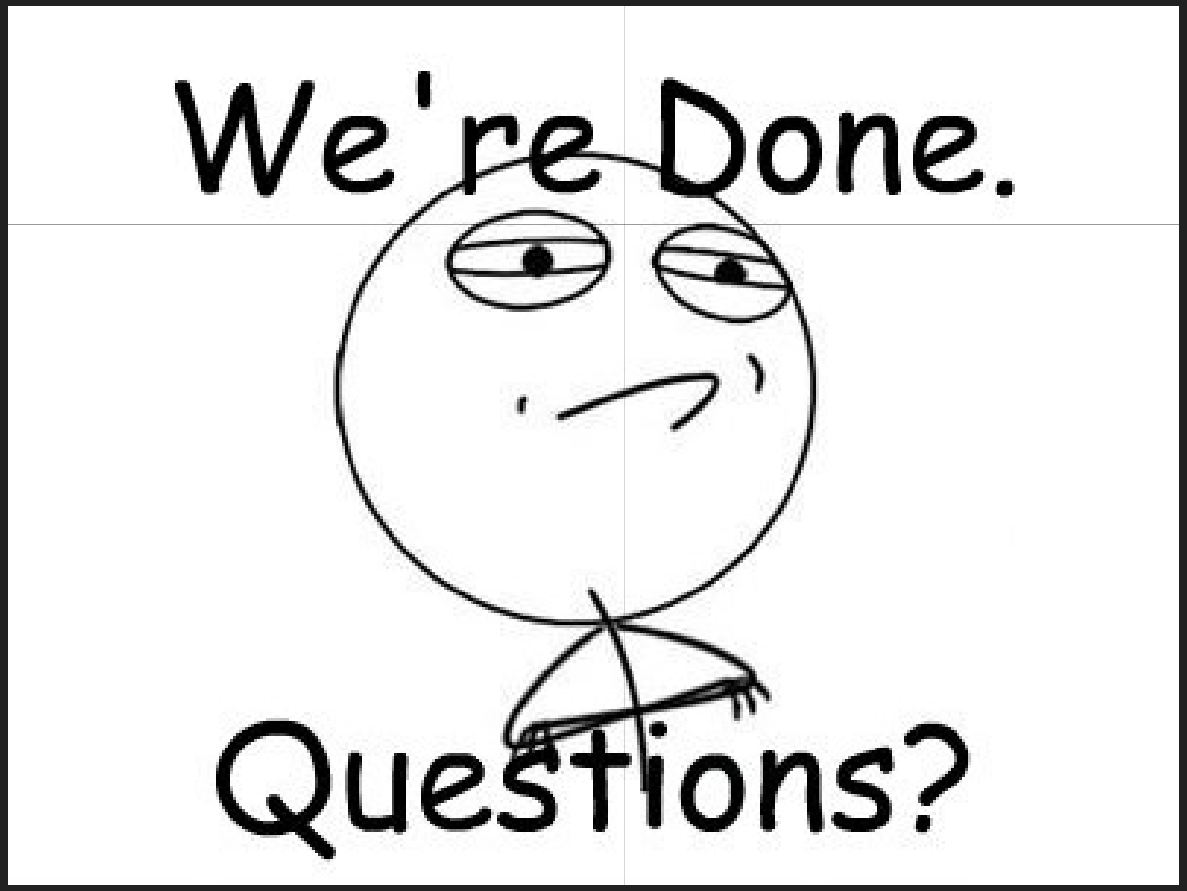
Do things as late as possible and never do them twice. Delay execution as much as possible until it is really needed. TIme is money!

Memoization - a way of optimizing code so that it will return cached results for the same inputs.

Benefits & Conclusion

- No side effects when programming
- Reusability
- Faster execution
- No banana problem
- Scala will have a great future
  - Javascript is already the most popular language

Questions?



Thanks for your attention

Elizabeta Ilievska

Senior Developer @ Endava

skype: en_eilievska

linkedin: elizabeta-ilievska-8b57002b

Goran Kopevski

Senior Developer @ Endava

skype: goran_kopevski

linkedin: goran-kopevski-88444954

github: https://github.com/gkopevski