Programare declarativă[1]

Introducere în programarea funcțională folosind Haskell

Traian Florin Șerbănuță
Ioana Leuștean

Departamentul de Informatică, FMI, UNIBUC
traian.serbanuta@fmi.unibuc.ro
ioana@fmi.unibuc.ro

[1] bazat pe cursul Informatics 1: Functional Programming de la University of Edinburgh

Programare declarativă vs. imperativă

Ce vs. cum

Programare imperativă (Cum)

Explic mașinii, pas cu pas, algoritmic, cum să facă ceva și ca rezultat, se întâmplă ce voiam să se întâmple.

Programare declarativă (Ce)

Îi spun mașinii ce vreau să se întâmple și o las pe ea să știe cum să realizeze acest lucru. :-)

Programare declarativă vs. imperativă

Ce vs. cum

Programare imperativă (Cum)

Explic mașinii, pas cu pas, algoritmic, cum să facă ceva și ca rezultat, se întâmplă ce voiam să se întâmple.

- limbaje procedurale
- limbaje de programare orientate pe obiecte

Programare declarativă (Ce)

Îi spun mașinii ce vreau să se întâmple și o las pe ea să știe cum să realizeze acest lucru. :-)

- limbaje de interogare a bazelor de date
- limbaje de programare logică
- limbaje de programare funcțională

Programare imperativă vs. declarativă

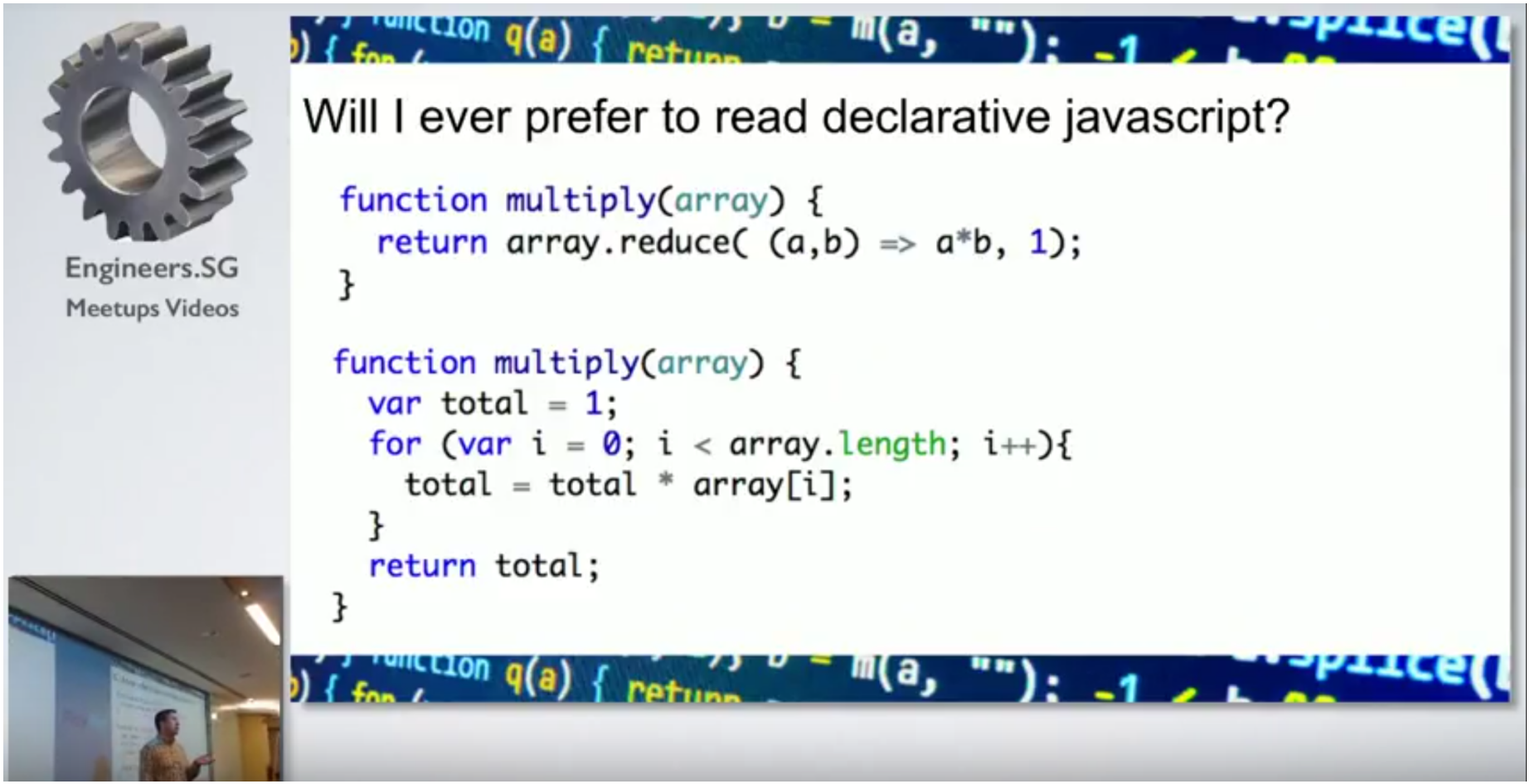
Diferențe

- Modelul de computație: algoritm vs. relație
- Ce exprimă un program: cum vs. ce
- Variabile/parametrii: atribuire distructivă vs. non-distructivă
- Structuri de date: alterabile vs. explicite
- Ordinea de execuție: efecte laterale vs. neimportantă
- Expresii ca valori: nu vs. da
- Controlul execuției: responsabilitatea programatorului vs a mașinii

Agregarea datelor dintr-o colecție (JS)

C. Boesch, Declarative vs Imperative Programming - Talk.JS

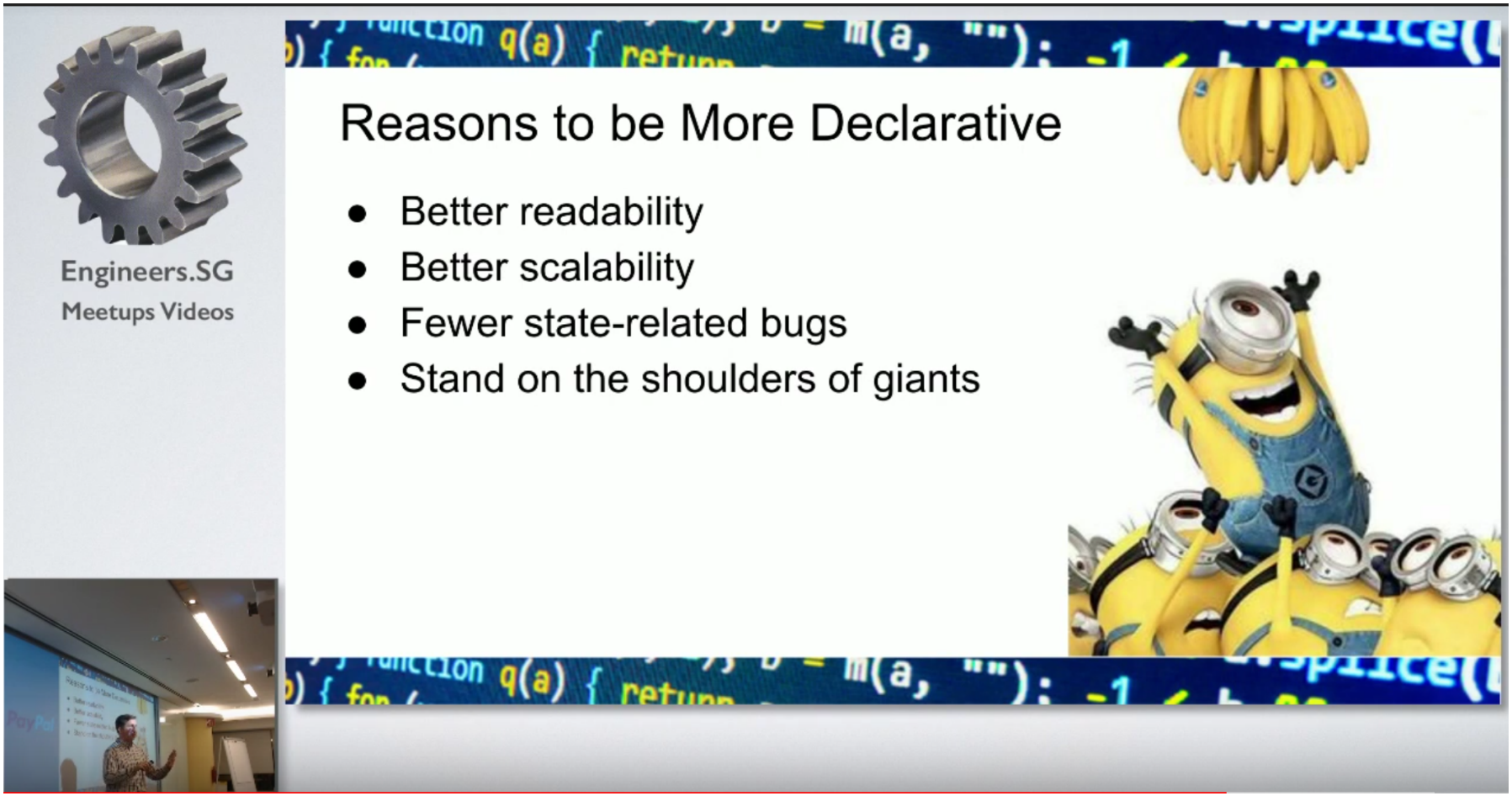
https://www.youtube.com/watch



Agregarea datelor dintr-o colecție (JS)

C. Boesch, Declarative vs Imperative Programming - Talk.JS

https://www.youtube.com/watch



Programare funcțională în Haskell

- Programarea funcțională e din ce în ce mai importantă în industrie
  - Haskell e folosit în proiecte de Facebook, Google, Microsoft, ...
  - mai multe la https://wiki.haskell.org/Haskell_in_industry
- Programare funcțională în limbajul vostru preferat de programare:
  - Java 8, C++x11, C#, Python, PHP, JavaScript
  - Funcții anonime (λ-abstracții)
  - Funcții de procesare a fluxurilor de date: filter, map, reduce

Programare funcțională în Haskell

- Programarea funcțională e din ce în ce mai importantă în industrie
  - Haskell e folosit în proiecte de Facebook, Google, Microsoft, ...
  - mai multe la https://wiki.haskell.org/Haskell_in_industry
- Programare funcțională în limbajul vostru preferat de programare:
  - Java 8, C++x11, C#, Python, PHP, JavaScript
  - Funcții anonime (λ-abstracții)
  - Funcții de procesare a fluxurilor de date: filter, map, reduce

De ce Haskell? (din cartea Real World Haskell)

The illustration on our cover is of a Hercules beetle. These beetles are among the largest in the world. They are also, in proportion to their size, the strongest animals on Earth, able to lift up to 850 times their own weight. Needless to say, we like the association with a creature that has such a high power-to-weight ratio.

Programare funcțională în Haskell

- Programarea funcțională e din ce în ce mai importantă în industrie
  - Haskell e folosit în proiecte de Facebook, Google, Microsoft, ...
  - mai multe la https://wiki.haskell.org/Haskell_in_industry
- Programare funcțională în limbajul vostru preferat de programare:
  - Java 8, C++x11, C#, Python, PHP, JavaScript
  - Funcții anonime (λ-abstracții)
  - Funcții de procesare a fluxurilor de date: filter, map, reduce

De ce Haskell? (din cartea Real World Haskell)

[It] is a deep language and [...] learning it is a hugely rewarding experience.

Nou	Radical diferit de limbajele cu care suntem obișnuiți
Puternic	Cod concis, rapid și sigur
Plăcut	Tehnici elegante pentru rezolvarea de probleme concrete

Nou

Programarea funcțională

O cale profund diferită de a concepe ideea de software

- În loc să modificăm datele existente, calculăm valori noi din valorile existente, folosind funcții
- Funcțiile sunt pure: aceleași rezultate pentru aceleași intrări
- Distincție clară între părțile pure și cele care comunică cu mediul extern
- Haskell e leneș: orice calcul e amânat cât de mult posibil
  - Schimbă modul de concepere al programelor
  - Permite lucrul cu colecții potențial infinite de date precum [1..]

Puternic

- Puritatea asigură consistență
  - O bucată de cod nu poate corupe datele altei bucăți de cod.
  - Mai ușor de testat decât codul care interacționează cu mediul
- Evaluarea lenește poate fi exploatată pentru a reduce timpul de calcul fără a denatura codul

```
firstK k xs = take k (sort xs)
```

- Minimalism: mai puțin cod, în mai puțin timp, și cu mai puține defecte
  - ...rezolvând totuși problema :-)

```
numbers = [1,2,3,4,5]
total = foldl (*) 0 numbers
doubled = map (* 2) numbers
```

- Oferă suport pentru paralelism și concurență

Elegant

- Idei abstracte din matematică devin instrumente puternice practice
  - recursivitate, compunerea de funcții, functori, monade
  - folosirea lor permite scrierea de cod compact și modular
- Rigurozitate: ne forțează să gândim mai mult înainte, dar ne ajută să scriem cod mai corect și mai curat
- Curbă de învățare în trepte
  - Putem scrie programe mici destul de repede
  - Expertiza în Haskell necesită multă gândire și practică
  - Descoperirea unei lumi noi poate fi un drum distractiv și provocator
    `http://wiki.haskell.org/Humor`

Plan curs

- Partea I: Noțiuni de bază de programare funcțională
  - Funcții, recursie, funcții de ordin înalt, tipuri
  - Operații pe liste: filtrare, transformare, agregare
- Partea II: Noțiuni avansate de programare funcțională
  - Polimorfism, clase de tipuri, modularizare
  - Tipuri de date algebrice - evaluarea expresiilor
  - Operațiuni Intrare/Ieșire
- Partea III: Capitole speciale de Haskell și programare funcțională
  - Agregare pe tipuri algebrice, monade, etc.

Resurse

- Paginile cursului:
  http://moodle.fmi.unibuc.ro/course/view.php?id=367
  http://unibuc.ro/~ileustean/PD.html
  - Prezentările cursurilor, forumuri, resurse electronice
  - Știri legate de curs vor fi postate pe ambele pagini
  - Notele la teste vor fi postate tot pe Moodle
- http://bit.do/progdecl
  - Cele mai noi variante ale cursurilor si laboratoarelor.
- Cartea online „Learn You a Haskell for Great Good”
  http://learnyouahaskell.com/
- Pagina Haskell http://haskell.org
  - Hoogle https://www.haskell.org/hoogle
  - Haskell Wiki http://wiki.haskell.org

Evaluare

Notare

- 2 parțiale (par1, par2), examen (ex)
- Nota finală: 1 (oficiu) + par1 + par2 + ex
- Notele vor fi postate (doar) pe pagina Moodle a cursului.

Condiție de promovabilitate

- Nota finală cel puțin 5
  - $5 > 4.99$

Activitate laborator

- La sugestia profesorului coordonator al laboratorului, se poate nota activitatea în plus față de cerințele obșnuite
- Maxim 1 punct (bonus la nota finală)

Parțial 1

- Valorează 4 puncte din nota finală
- Sâmbăta dintre a 6-a și a 7-a săptămână (11 noiembrie)
- Pe calculatoare
- Durată: 1 oră
- Acoperă materia din Partea I — Noțiuni de bază
- Cu acces la materiale descărcate pe calculator
- Fără acces la rețea/internet

Parțial 2

- Valorează 3 puncte din nota finală
- Sâmbăta dinainte de vacanța de iarnă (16 decembrie)
- Pe calculatoare
- Durată: 1 oră
- Materia din Partea II — Noțiuni avansate
- Cu acces la materiale descărcate pe calculator
- Fără acces la rețea/internet

Examen final

- Valorează 2 puncte din nota finală
- În sesiune
- Pe hârtie
- Materia: toată
- Durată: 2 ore
- Cu acces la materiale tipărite
- Fără acces la rețea/internet

Observații despre teste și notare

- Nu este necesar să dați toate testele pentru promovare
  - $par1 = 6,25 \wedge par2 = 5 \implies final = 5$
  - $par1 = 10 \implies final = 5$
  - Dar este necesară nota agregată cel puțin 5
- Pentru a lua nota finală 10 sunt necesare toate cele 3 teste
  - $par1 = 10 \wedge par2 = 10 \wedge lab = 10 \implies final = 9$
  - $par1 = 10 \wedge par2 = 10 \wedge lab = 10 \wedge ex = 2,50 \implies final = 9,50 \approx 10$
  - $test1 = 9 \wedge test2 = 9 \wedge lab = 4 \wedge test3 = 9 \implies final = 9,50 \approx 10$
- Cursurile se bazează unele pe altele
  - $\forall j < i$. examenul i presupune construcții și concepte acoperite de examenul j