

# **Въведение в Scheme**

**8 октомври, 2019**

**Малко за мен**



**Софийски университет „Св. Климент Охридски“  
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА**



I ❤️ teaching

I ❤️ functional programming



ТЕХНОЛОГИЧНО УЧИЛИЩЕ  
•ЕЛЕКТРОННИ СИСТЕМИ•





We turn innovative ideas  
into great software products

# Представете се

- Какви интереси имате?
- Какъв опит имате с програмирането?

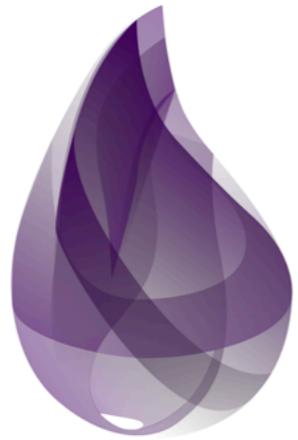
# Организационни неща

- Курс в moodle
- [github.com/triffon/fp-2019-20](https://github.com/triffon/fp-2019-20)
- Discord: <https://discord.gg/peH4M53>
- [dimitar.uzunov.dev@gmail.com](mailto:dimitar.uzunov.dev@gmail.com)

**Какво е функционално  
програмиране?**

# Стил на програмиране

**Защо да учит  
функционално  
програмиране?**



elixir



---

# Structure and Interpretation of Computer Programs

Second Edition



Harold Abelson and  
Gerald Jay Sussman  
with Julie Sussman

# Scheme

- Функционален език за програмиране
- Има прост синтаксис и правила – бързо се научава
- Това ни дава възможност да се концентрираме над структурата на програмите, които пишем, и процесите, породени от тяхното изпълнение
- Ще обръщаме внимание на структурата на програмите

# Scheme имплементации

- Chez Scheme
- Racket
- MIT Scheme
- Ще използваме R5RS стандарта

# Среди за разработка

- DrRacket
- Emacs
- Vim + tmux + Chez Scheme (например)
- repl.it

# Всеки мощен език за програмиране има:

- **примитивни изрази** – най-простите елементи в езика
- **средства за комбинация** – за създаване на съставни елементи от по-прости
- **средства за абстракция** – за именуване на съставни елементи, които да можем да използваме като примитивните елементи

# Изрази

- Примитивни изрази
- Комбинации
- Специални форми
- Всеки израз има стойност

# Примитивни изрази

- Булеви константи – #t, #f
- Числови константи – 42, -1, 3.14, 1/3
- Знакови константи – #\a, #\newline
- Низови константи – "Scheme is cool"
- Символи – +, square, odd?

# Комбинации

; Combinations

```
(+ 1 2) ; 3
(- 1000 334) ; 666
(* 2 3) ; 6
(/ 10 5) ; 2
```

; Arbitrary number of operands

```
(+ 1 2 3 4 5) ; 15
(* 25 4 12) ; 1200
```

; Nested combinations

```
(+ (* 3 (+ (* 2 4) (+ 3 5))) (+ (- 10 7) 6)) ; 57
```

; Pretty-printing

```
(+ (* 3
      (+ (* 2 4)
          (+ 3 5)))
      (+ (- 10 7)
          6))
```

# **Специални форми**

# define

```
; "define" special form
(define pi 3.14159) ; pi
(define radius 100) ; radius
(* pi (* radius radius)) ; 31415.89999999998

(define circumference (* 2 pi radius)) ; circumference
circumference ; 628.318
```

# Дефиниране на процедура

```
; Defining procedures
(define (square x) (* x x))
(square 5) ; 25
(square (+ 2 5)) ; 49
(square (square 3)) ; 81

(define (sum-of-squares x y)
  (+ (square x) (square y)))
(sum-of-squares 3 4) ; 25
```

```
(define (square x) (* x x))  
| | | | |  
To square something, multiply it by itself.
```

# cond

; "cond" special form

```
(define (abs x)
  (cond ((< x 0) (- x))
        ((= x 0) 0)
        ((> x 0) x)))
```

```
(define (abs x)
  (cond ((< x 0) (- x))
        (else x)))
```

# **if**

; "if" special form

```
(define (abs x)
  (if (< x 0)
    (- x)
    x) )
```