# MI-RUB Building Stones

Pavel Strnad
pavel.strnad@fel.cvut.cz

Dept. of Computer Science, FEE CTU Prague,
Karlovo nám. 13, 121 35
Praha, Czech Republic

MI-RUB, WS 2011/12

# Contents

# Blocks

## Blocks in Ruby

- blocks are objects (compare to Java),
- blocks are extra parameters to methods,
- blocks implement iterators,
- blocks implement closures!

# Hello World!

```
puts "Hello world!"
5.times { print "ahoj!\n" }
```

# Blocks can have parameters

Blocks can have parameters

```
[1, 2, 3].each {|x| print "#{x}!\n" }
```

# Blocks can have more parameters

Blocks can have more parameters

```
[1, 2, 3].each_with_index {|item, index|
                    print "#{item} at #{index}!\n" }
```

# Blocks can be stored

Blocks can be stored and called later.

```
x = Proc.new {|x| print "number #{x}"}
x.call(5)
```

# Block's methods

Block is object, so we can list its methods.

```
x = Proc.new {|x| print "number #{x}"}
print x.methods
```

# Block as extra argument.

```
def call_block
  puts "Start of method"
  yield
  yield
  puts "End of method"
end
call_block { puts "In the block" }
```

# Block as an extra argument.

```
def who_says_what
  yield("Dave", "hello")
  yield("Andy", "goodbye")
end
who_says_what {|person, phrase| puts "#{person} says #{phrase}"
    }
```

# Block parameter as an extra argument.

```
def who_says_what(&block)
  block.call("Dave", "hello")
  block.call("Andy", "goodbye")
end
who_says_what {|person, phrase| puts "#{person} says #{phrase}"
    }
```

# Lambda functions

## Lambdas in Ruby

- lambdas are almost same as Procs,
- different behaviour for return statement (return in Proc is global, in Lambda local),
- lambdas implement anonymous functions theory (lambda calculi).

# Lambda functions Example

```ruby
def proc_return
  Proc.new { return "Proc.new"}.call
  return "proc_return method finished"
end

def lambda_return
  lambda { return "lambda" }.call
  return "lambda_return method finished"
end

puts proc_return
puts lambda_return

# => Proc.new
# => lambda_return method finished
```

# Method objects

## Method objects in Ruby

- Method object encapsulates a method,
- behaviour similar to lambda funcitons.

```
class Array
  def iterate!(code)
    self.each_with_index do |n, i|
      self[i] = code.call(n)
    end
  end
end
def square(n)
  n ** 2
end
array = [1, 2, 3, 4]
array.iterate!(method(:square))
```

# Blocks, Procs, Lambdas and Method objects

- Procs represent procedures, closures - piece of code that is inserted on place within a context.
- Blocks are unborned Procs. They are automatically converted during call to Proc objects. They are used as method parameters and provides an easy way for iteration and callback programming.
- Lambdas provides function objects. Their behaviour is similar to anonymous functions.
- Method objects are envelopes around methods. They are good for passing methods as parameters.
- If you want to see more examples go to http://www.robertsosinski.com/2008/12/21/understanding-ruby-blocks-procs-and-lambdas/

# Block as iterator.

```
animals = %w( ant bee cat dog elk ) # create an array
animals.each {|animal| puts animal } # iterate over the
    contents
```

# For loops can have many forms.

```
animals = %w( ant bee cat dog elk ) # create an array
animals.each {|animal| puts animal } # iterates over the
    contents

for animal in animals
  puts animal
end

#compare to java form of for loop
for i in 1..10
  puts i
end
```

# Iterators in Ruby

## We have many iterators in Ruby

- **each** iterator,
- **times** iterator,
- **upto** iterator,
- **step** and many more...

## But...

- all of them are regular methods with a block parameter!!!

# Iterators in Ruby

```
10.times {|i| print "#{i} "}

1.upto(10) {|i| print "#{i} "}

1.step(10, 2) { |i| print "#{i} "}
```

Examples

# Iterators in Ruby - Exercise 1

## Exercise 1

**Traversing an array**

- Write a method which yields the odd-indexed elements for an array if the user supplied a block to the method, and which returns an array of the results otherwise.
- Hint: the method block_given? returns true if a method was invoked with a block.

# Iterators in Ruby - Exercise 1

```ruby
def oddElements3(l)
  ...
end

puts oddElements3([1,2,3,4,5,6])
oddElements3([1,2,3,4,5,6]) do |x|
   puts x
end
```

# Basic I/O

- **I/O** methods are implemented in **Kernel** module.
- *Kernel* module provides these I/O methods *gets, open, print, printf, putc, puts, readline, readlines*.
- Base class for I/O objects is **IO** class.

```
print "Enter your name: "
name = gets
```

Input from stdin is simple

# Working with files

```
f=File.open("testfile","w")
f << "test line 1\n" << "test line 2\n"
f.close

f=File.open("testfile","r")
puts f.readlines
f.close

f=File.open("testfile","r")
puts f.gets
f.close
```

# Working with files 2

Blocks rules!

```
File.open("testfile","w") {|f|
  f << "test line 1\n" << "test line 2\n"
}

File.open("testfile","r") {|f|
  puts f.readlines
}

f=File.open("testfile","r") {|f|
  puts f.gets
}
```

# ARGV

- All command-line arguments are stored in ARGV variable.
- Name of the script is not included.

```
case ARGV[0]
when "start"
  STDOUT.puts "called start"
when "stop"
  STDOUT.puts "called stop"
else
  STDOUT.puts <<-EOF
Please provide command name
Usage:
  server start
  server stop
  server restart
EOF
end
```

# Files in Ruby - Exercise 2

## Searching a file

- Write a method sameword(file) which searches through a file for any potential word duplications such as "the the".
- How would you extend this to search for duplications that occured across two lines ("...the
the...")?

```
def sameword (file)
    ...
end

sameword("sametest")
```

# Files in Ruby - Exercise 3

## Counting words

- Write a method count(file) which outputs a table of words and their frequency.

```
def count (file)
    ...
end

sameword("sametest")

File sametest:
Cat car cat.

Output:
cat 2
car 1
```