RoR 4 beginners

1. Wstęp do Ruby

daftcode

(logo) Ruby

Interpretowany

```
> puts 'hello!'
hello!
=> nil
> 2 + 3
=> 5
>
```

Obiektowy

```
2.class   #=> Integer
2.methods #=> ...
2.+(3)    #=> 5
2 + 3     #=> 5
```

Obiektowy

```
3.even?            #=> false
5.78.round         #=> 6
'hello'.capitalize #=> 'Hello'
[1, 2, 3].min      #=> 1
Time.now.friday?   #=> false
```

Dynamiczny

```
a = 3     #=> 3
a * 2     #=> 6
a = 'qwe' #=> "qwe"
a * 2     #=> "qweqwe"
```

Dynamiczny

```ruby
things = [0, 0.5, 'string']

things[0].class #=> Integer
things[1].class #=> Float
things[2].class #=> String

things[2] = []
things[2].class #=> Array
```

Refleksyjny

```
my_method() #=> NoMethodError

define_method(:my_method) do
  'Works now!'
end

my_method() #=> "Works now!"
```

Składnia

Blok

```ruby
puts 'poza blokiem'

begin
  puts 'w bloku'
end
```

Logika

```
if 2 + 3 == 5
  # do something
end

finished = true

unless finished
  # do something
end
```

Logika

```
condition = false

a = if condition
  3
else
  5
end

a = condition ? 3 : 5
```

Logika

```
if true && true
  # and operator
end

if true || false
  # or operator
end

if !false
  # not operator
end
```

Błędy

```
begin
  2 + 'string?'
rescue TypeError
  puts 'aw, snap'
ensure
  # to się wykona zawsze
end

throw StandardError
```

Pętla

```ruby
i = 0
while i < 3
  i += 1
end

i = 0
loop do
  i += 1
  break if i < 3
end
```

Inline

```ruby
broken = false
puts 'works!' unless broken

i = 0
i += 1 while i < 10

number = 2 + 'string' rescue 7
```

Metoda

```
def factorial(n)
  outcome = 1
  base = 1

  while base <= n
    outcome *= base
    base += 1
  end

  return outcome
end

factorial(5) #=> 120
factorial 3  #=> 6
```

Konstrukcje

true, false, nil

```
5 && 'string' && true #=> true
!123 || nil || false  #=> false

array = [0, 1, 2]
array[3] #=> nil
```

string i symbol

```ruby
"Ala ma #{2 + 3} kotów"         #=> "Ala ma 5 kotów"

'a'.object_id == 'a'.object_id  #=> false
'a' + 'bc'                      #=> "abc"

:a.object_id == :a.object_id    #=> true
:a + :bc                        #=> NoMethodError
```

Hash

```
hash = {
  'key' => 'value',
  3 => 8.5,
  symbol: :value
}

hash[3]          #=> 8.5
hash[:symbol]    #=> :value
hash['not_here'] #=> nil
```

Splat

```
def sentence word='hey', *words
  "#{word.capitalize} #{words.join(' ')}."
end

sentence                #=> "Hey ."
sentence 'hi'           #=> "Hi ."
sentence 'hi', 'hello'  #=> "Hi hello."

array = ['you', 'too']
sentence 'hi', *array   #=> "Hi you too."
```

Double splat

```
def sentence word: 'hey', **words
  "#{word.capitalize} #{words.values.join(' ')}."
end

sentence               #=> "Hey ."
sentence word: 'hi'    #=> "Hi ."
sentence desc: 'hi'    #=> "Hey hi."

hash = {a: 'you', b: 'too'}
sentence **hash        #=> "Hey you too."
```

Range

```
(1..3).to_a          #=> [1, 2, 3]
(1...3).to_a         #=> [1, 2]

('a'..'e').to_a.join #=> 'abcde'

(0.2..1.6).bsearch do |f|
  Math.log(f) >= 0
end #=> 1.0
```

Enumeracja

Wywołanie bloku

```
def operation n
  if block_given?
    yield n
  else
    n + 3
  end
end

operation(3) #=> 6

operation(3) do |passed|
  passed + 5
end #=> 8

operation(3) { |p| p + 5 }
```

Iteracja

```
10.times { |t| puts t }

(0..10).each { |t| puts t }

loop { puts 'ping'; sleep 1 }
```

Mapowanie

```
array = [1] * 4
#=> [1, 1, 1, 1]

array.map { |el| el + 1 }
#=> [2, 2, 2, 2]

array.map.with_index { |el, i| el + i }
#=> [1, 2, 3, 4]

array.map! { |el| -el }
#=> [-1, -1, -1, -1]

array #=> [-1, -1, -1, -1]
```

Mapowanie

```
hash = {
  name: 'Jan',
  surname: 'Kowalski'
}
#=> {:name=>"Jan", :surname=>"Kowalski"}

hash.map{ |k, v| "#{k}: #{v}"  }.join(', ')
#=> "name: Jan, surname: Kowalski"
```

Shorthand

```
chain do |base, *args|
  base.other_method(*args)
end

chain(&:other_method)
```

```
[1, 2, 3].map do |base|
  base.to_i
end

[1, 2, 3].map(&:to_i)
```

Shorthand

```
[1, 2, 3].map { |e| -e }
# => [-1, -2, -3]

[1, 2, 3].map { |e| e.-@() }
# => [-1, -2, -3]

[1, 2, 3].map(&:-@)
# => [-1, -2, -3]
```

```
[1, 2, 3].map.with_index { |e, i| e + i }
# => [1, 3, 5]

[1, 2, 3].map.with_index { |e, i| e.+(i) }
# => [1, 3, 5]

[1, 2, 3].map.with_index(&:+)
# => [1, 3, 5]
```

The Ruby way

Refactor

```
def factorial(n)
  outcome = 1
  base = 1

  while base <= n
    outcome *= base
    base += 1
  end

  return outcome
end
```

Refactor

```ruby
def factorial(n)
  outcome = 1
  base = 1

  while base <= n
    outcome *= base
    base += 1
  end

  return outcome
end
```

```ruby
def factorial(n)
  outcome = 1

  (1..n).each do |base|
    outcome *= base
  end

  return outcome
end
```

Refactor

```ruby
def factorial(n)
  outcome = 1

  (1..n).each do |base|
    outcome *= base
  end

  return outcome
end
```

```ruby
def factorial(n)
  outcome = (1..n).inject(1) do |base, acc|
    acc * base
  end

  return outcome
end
```

Refactor

```ruby
def factorial(n)
  outcome = (1..n).inject(1) do |base, acc|
    acc * base
  end

  return outcome
end
```

```ruby
def factorial n
  (1..n).inject(1) do |base, acc|
    acc * base
  end
end
```

Refactor

```ruby
def factorial n
  (1..n).inject(1) do |base, acc|
    acc * base
  end
end
```

```ruby
def factorial n
  (1..n).inject(1) { |base, acc| base * acc }
end
```

Refactor

```ruby
def factorial n
  (1..n).inject(1) { |base, acc| base * acc }
end
```

```ruby
def factorial n
  (1..n).inject(1, :*)
end
```

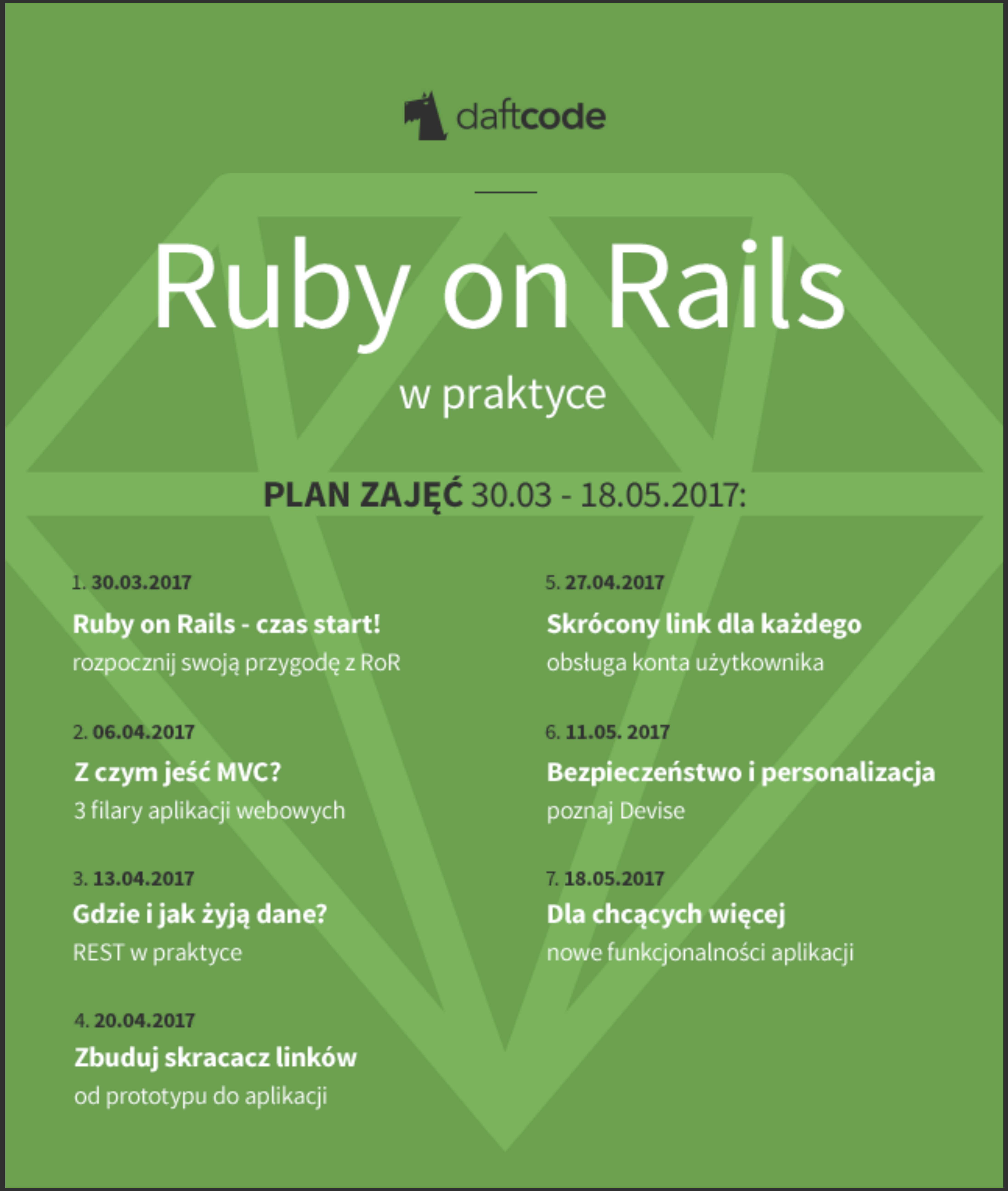
Refactor

```ruby
def factorial(n)
  outcome = 1
  base = 1

  while base <= n
    outcome *= base
    base += 1
  end

  return outcome
end
```

```ruby
def factorial n
  (1..n).inject(1, :*)
end
```



daftcode

Ruby on Rails

w praktyce

PLAN ZAJĘĆ 30.03 - 18.05.2017:

1. **30.03.2017**

Ruby on Rails - czas start!

rozpocznij swoją przygodę z RoR

2. **06.04.2017**

Z czym jeść MVC?

3 filary aplikacji webowych

3. **13.04.2017**

Gdzie i jak żyją dane?

REST w praktyce

4. **20.04.2017**

Zbuduj skracacz linków

od prototypu do aplikacji

5. **27.04.2017**

Skrócony link dla każdego

obsługa konta użytkownika

6. **11.05. 2017**

Bezpieczeństwo i personalizacja

poznaj Devise

7. **18.05.2017**

Dla chcących więcej

nowe funkcjonalności aplikacji

Have a spooky