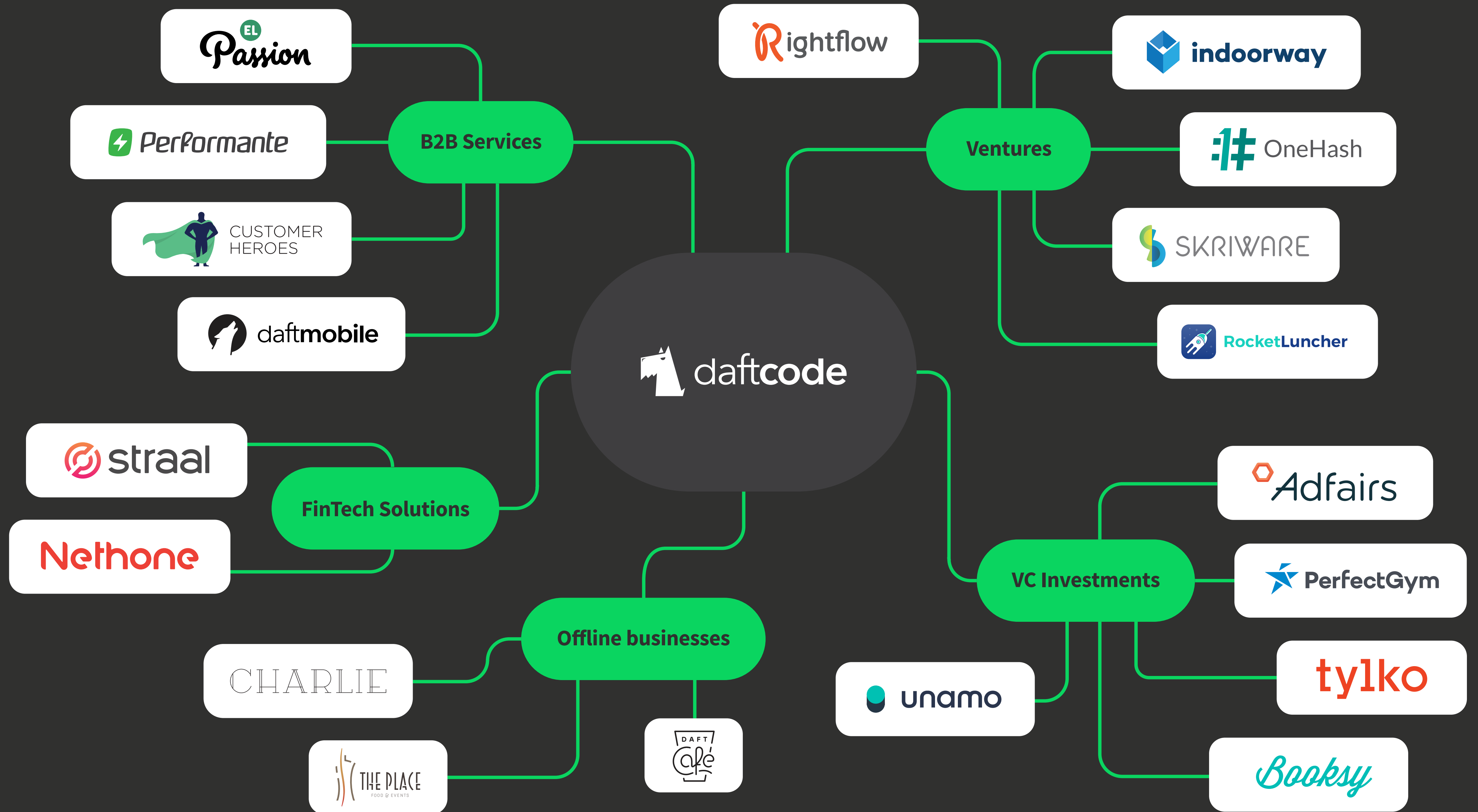


Ruby on Rails

w praktyce





Ruby

Cechy języka

Obiekty

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

Strukturalny

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

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

  return outcome
end

factorial 5 ==> 120
```

Refleksyjny

```
my_method() #=> NoMethodError
```

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

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

Typowanie

dynamiczne

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

```
things[0].class #=> Fixnum
```

```
things[1].class #=> Float
```

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

```
things[2] = []
```

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


Typowanie

ducktype

```
def get_excited_with thing
  "#{thing.class}s are awesome!"
end
```

```
get_excited_with 1.5           #=> Floats are awesome!
get_excited_with 'text'        #=> Strings are awesome!
get_excited_with [:this, 'array'] #=> Arrays are awesome!
```

Typowanie

silne

```
'string' + 1           #=> TypeError  
'string' * 'another'  #=> TypeError  
10 - 'string'         #=> TypeError
```

Podstawowe konstrukcje

Definicja metody

```
def my_method(param1, param2='default value')  
    param1 + param2  
end
```

```
my_method('this is a ')           #=> 'this is a default value'  
my_method('this is ', 'custom')  #=> 'this is custom'
```


Splat

```
def sentence *args  
  args.join(' ').capitalize  
end
```

```
sentence 'hello', 'world' #=> 'Hello world'
```

```
array = ['ala', 'ma', 'kota']
```

```
sentence *array           #=> 'Ala ma kota'
```

If-else

```
if condition  
  condition_true  
else  
  condition_false  
end
```

```
unless condition  
  condition_false  
else  
  condition_true  
end
```

```
do_something if condition
```

Case

```
def grade percent
  case percent
  when 100
    5
  when 70...100
    4
  when 50...70
    3
  else
    2
  end
end
```

```
grade 100      #=> 5
grade 66       #=> 3
grade 'error'  #=> 2
```

While

```
while condition_true  
  do_something  
end
```

```
until condition_false  
  do_something  
end
```

```
loop do # while true  
  do_something  
  break if condition_false  
end
```

```
do_something until condition_false
```


Rescue

```
begin
  do_something
rescue SomeError => e
  error_action
ensure
  always_do_this
end

do_something rescue error_action
```

Warunkowe przypisania

```
variable = get_value rescue get_other_value
```

```
variable = get_value if condition_true
```

```
variable = if condition_true  
  get_value  
else  
  get_other_value  
end
```

```
variable = condition_true ? get_value : get_other_value
```

Nil

```
nil.object_id #=> 8  
nil.object_id #=> 8
```

```
5.nil?          #=> false
```

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

Symbol

```
'string'.object_id  #=> ...680
'string'.object_id  #=> ...540
'string' + 's'      #=> 'strings'

:symbol.object_id    #=> 795548
:symbol.object_id    #=> 795548
:symbol + :s         #=> NoMethodError
```


Hash

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

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

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 { |f| Math.log(f) >= 0 } #=> 1.0
```

Blok

```
[1, 2, 3].each do |number|  
  number.to_s  
end
```

```
[1, 2, 3].each { |number| number.to_s }
```

Wywołanie bloku

```
def message_for number
  if number.is_a? Fixnum
    yield(number) if block_given?
  else
    'NaN'
  end
end
```

```
message_for(5) { |n| "#{n} is fine!" }      #=> '5 is fine!'
message_for('asd') { |n| "#{n} is not :(" } #=> 'NaN'
```


For / foreach

```
10.times { |t| puts "hello #{t}" }
```

```
(0..10).each { |t| puts "hello #{t}" }
```

Each / map

```
array = [1] * 10
```

```
array.each { |e1| puts "hello #{e1}" }
```

```
array.each.with_index { |e1, i| puts "hello #{i}" }
```

```
array.map { |e1| e1 + 1 }
```

```
array.map.with_index { |e1, i| e1 + i }
```

Shorthand bloku

```
some_method do |first_argument, *all_the_rest|  
  first_argument.do_something(*all_the_rest)  
end
```

```
some_method(&:do_something)
```

Przykład inject

```
[1, 2, 3].inject { |element, sum| sum + element }
```

```
[1, 2, 3].inject { |element, sum| element + sum }
```

```
[1, 2, 3].inject { |element, sum| element.+(sum) }
```

```
[1, 2, 3].inject(&:+)
```

```
[1, 2, 3].inject(:+) #=> 6
```

```
[1, 2, 3].reduce(:+) #=> 6
```

The Ruby way

Refactor

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

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

  return outcome
end
```

```
factorial 5 #=> 120
```

Refactor

```
def factorial n
  outcome = 1

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

  return outcome
end

factorial 5 #=> 120
```


Refactor

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

  return outcome
end
```

```
factorial 5 #=> 120
```

Refactor

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

```
factorial 5 #=> 120
```

Refactor

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

```
factorial 5 ==> 120
```

Refactor

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

```
factorial 5 #=> 120
```

Refactor

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

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

  return outcome
end
```

```
factorial 5 #=> 120
```

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

```
factorial 5 #=> 120
```

Zajęcia praktyczne

Dlaczego RoR?

- kompleksowy
- popularny - dużo rozwiązań
- duże możliwości, niski entry level

Zajęcia praktyczne



Ruby on Rails

w praktyce

1. 30.03.2017

Ruby on Rails - czas start!

Rozpocznij swoją przygodę z RoR

2. 06.04.2017

Z czym jeść MVC?

Trzy 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

The image features a dark blue background with decorative elements in the corners. In the top-right and bottom-left corners, there are clusters of small, light blue dots arranged in a pattern that resembles a stylized 'X' or a series of overlapping triangles. The text 'Pytania?' is centered in the middle of the image in a light blue, serif font.

Pytania?