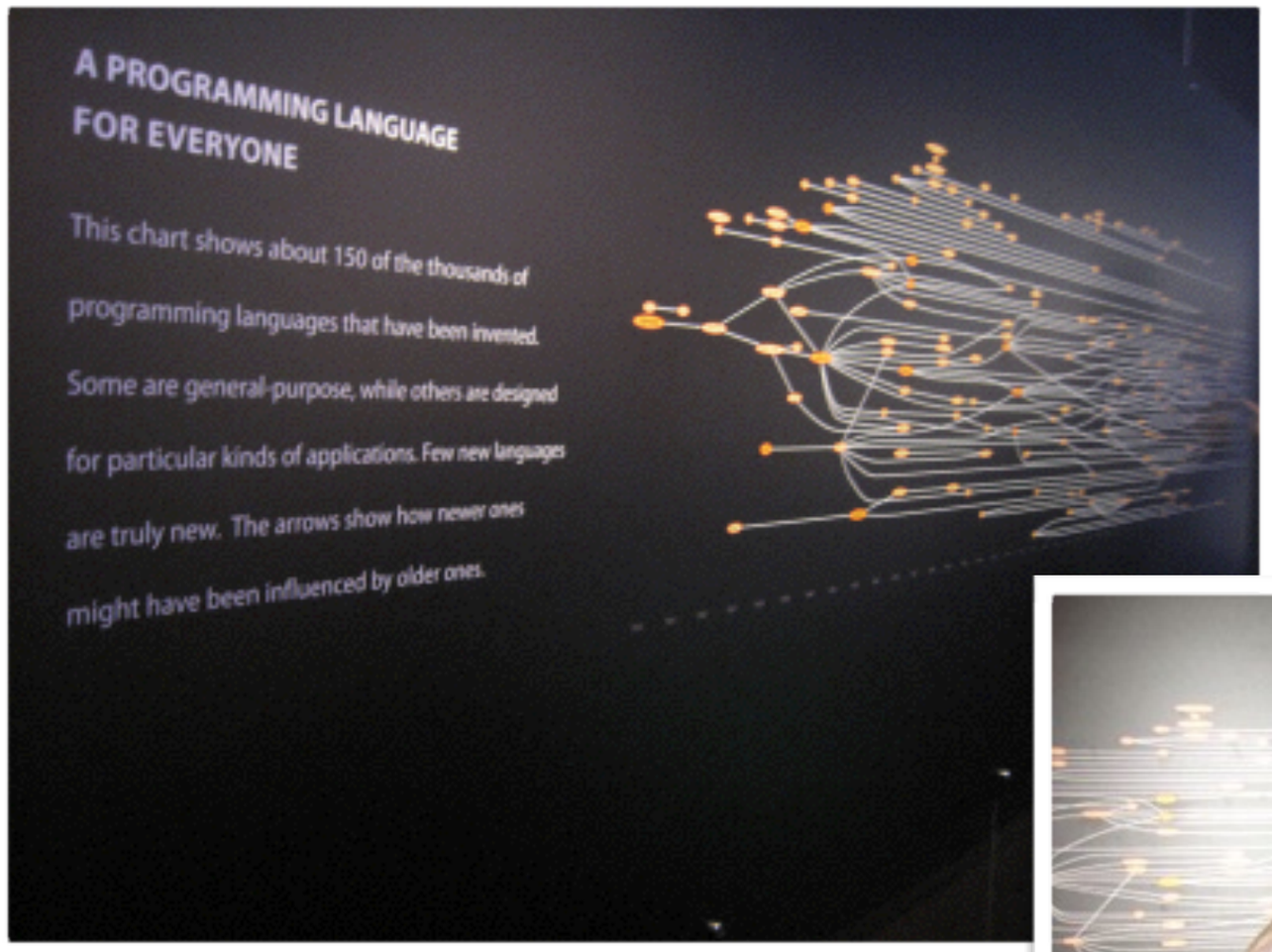
PRODUCENT OPROGRAMOWANIA
komp

Wieża Babel języków programowania

Czyli jak zostać poliglotą
i dlaczego warto.

Adam Kruszewski
<adam.kruszewski@tkomp.pl>

Pieter Bruegel Starszy - “Mała” Wieża Babel (1563), fragment



Ściana języków programowania
w Muzeum Historii Komputerów
w Kalifornii

COMPUTER HISTORY MUSEUM

http://www.computerhistory.org/atchm/tag/programming-language/

Każdy znajdzie
język programowania
dla siebie.

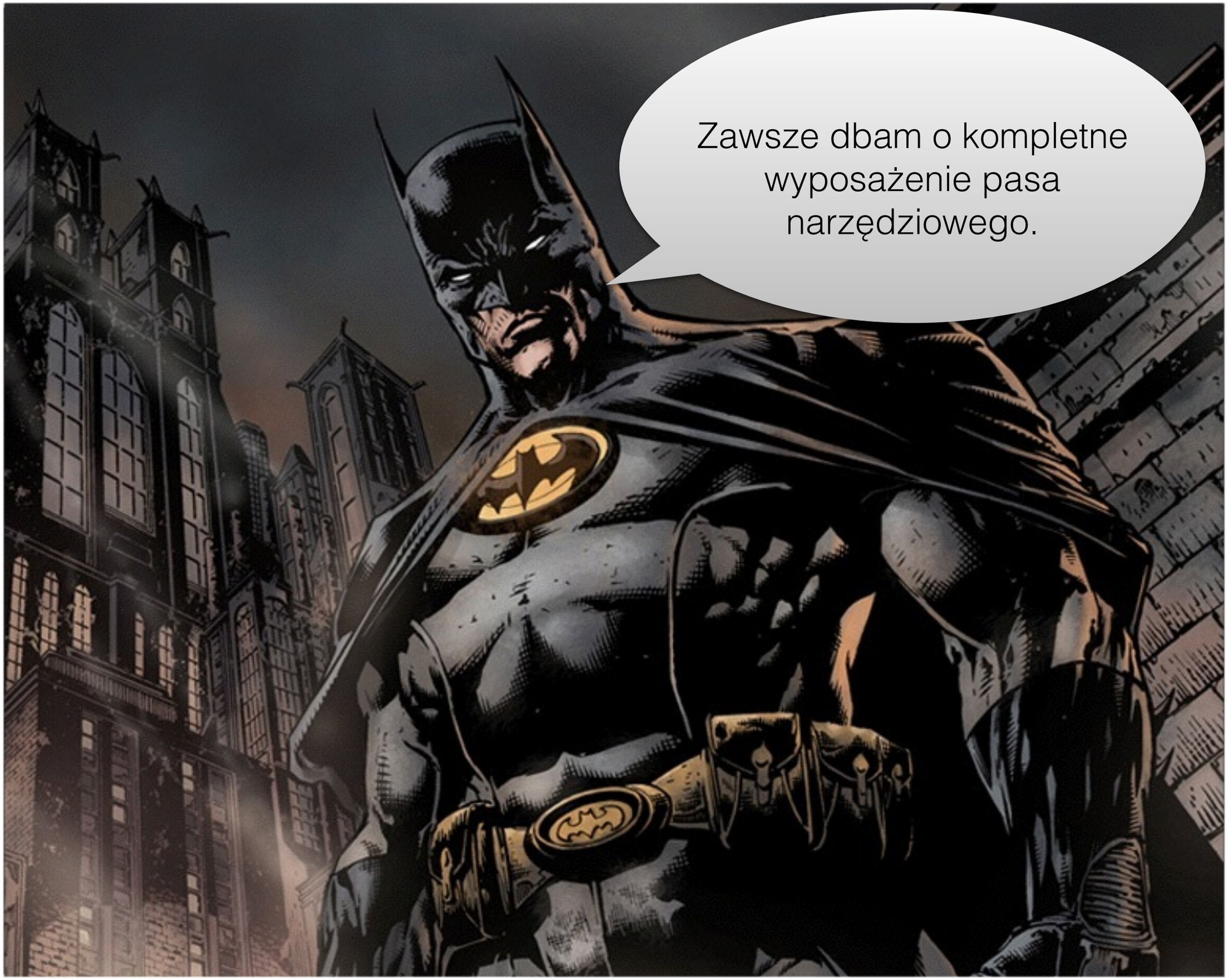
- używany głównie w środowisku typu Enterprise (aplikacje typu webowe)
- wsparcie dla środowiska rozproszonego, ESB, sprawdzone serwery aplikacyjne
- ogromna ilość dostępnych bibliotek
- “wielki powrót” do aplikacji nie serwerowych dzięki Androidowi.

- używany do aplikacji desktopowych dla systemu Windows
- oraz do aplikacji webowych (ASP.NET + MVC.NET)
- całkiem spora ilość dostępnych bibliotek (większość popularnych bibliotek z Javy została sportowana)
- nie obecny na większości urządzeń mobilnych (możliwe, że to się zmieni dzięki tabletom z Windows 8 na architekturze ARM)

Jeżeli jedyne co posiadasz to młotek, każdy problem zaczyna przypominać gwóźdź.

- tradycyjnie przypisywane Abrahamowi Maslowowi

z filmu Thor, Paramout Pictures



fragment z http://static.comicvine.com/uploads/original/2/25835/2013064-fabokbatman.jpg

Ruby

- stworzony w latach 90 przez Yukihiro “Matz-a” Matsumoto

- dynamicznie typowalny, **obiektowy**, ze wsparciem dla “**refleksji**”, **multi-paradygmatowy** z automatycznym zarządzeniem pamięcią, **interpretowany**.

- inspirowany językami Perl, Smalltalk, Eiffel, Ada oraz Lisp.

- zyskał popularność dzięki frameworkowi do tworzenia aplikacji webowych **Ruby on Rails.**

- Używany przez yellowpages.com, twich.tv, GitHub, SlideShare.

Ruby

```
18 + 'trzy owieczki'
TypeError: String can't be coerced into Fixnum
  from (irb):5:in `+'
  from (irb):5
  from /usr/bin/irb:12:in `<main>’

["100", 100.0, 100].each { |x| puts x.to_i }
100
100
100
```

Jeśli chodzi jak kaczka i kwacze jak kaczka…

```
18 + 'trzy owieczki'
TypeError: String can't
  from (irb):5:in `+'
  from (irb):5
  from /usr/bin/irb:12:

["100", 100.0, 100].eac
100
100
100
```

http://cdn.trendhunterstatic.com/thumbs/duckvengers-by-loki.jpeg

Ruby

```
class NilClass
  def blank?
    true
  end
end

class String
  def blank?
    self.size == 0
  end
end

["", "BitCamp", nil].each do |elem|
   puts elem unless elem.blank?
end

=> BitCamp
```

Ruby

```
class Filozof
  def method_missing ( m_name, *args, &block )
    "nie wiem jak wykonac polecenie #{m_name} ale niech nad tym
pomysle chwilke ..."
  end
end

f.wyrzuc_smieci
=> "nie wiem jak wykonac polecenie wyrzuc_smieci ale niech nad tym
pomysle chwilke ..."
```

Ruby

```
class Filozof
  def method_missing ( m_name, *args, &block )
    "nie wiem jak wykonac polecenie #{m_name} ale niech nad tym
pomysle chwilke ..."
  end
end

f.wyrzuc_smieci
=> "nie wiem jak                                    ym
pomysle chwilke
```

```
class Product < ActiveRecord::Base
end

p = Product.new
p.product_no = 1234
p.desc = "You need to have it!”

p.save

produkt = Product.find_by(product_no: 1234)
```

Ruby

"I used to be a programmer like you, then I took and arrow in the knee."

```
    "nie wiem jak wykonac polecenie #{
pomysle chwilke ..."
  end
end

f.wyrzuc_smieci
=> "nie wiem jak
pomysle chwilke
```

```
class Product < Activ
end

p = Product.new
p.product_no = 1234
p.desc = "You need to

p.save

produkt = Product.find_by(product_no: 1234)
```

http://img0.joyreactor.com/pics/post/games-Skyrim-arrow-in-the-knee-memes-826466.jpeg

Ruby

- Ruby, strona języka (http://ruby-lang.org/)
- Aptana RedRails, IDE (http://www.aptana.com/products/radrails)
- Ruby on Rails, framework do tworzenia aplikacji webowych (http://rubyonrails.org/)
- Why's (poignant) guide to Ruby, zabawne wprowadzenie do programowania w Rubym (http://mislav.uniqpath.com/poignant-guide/)
- TryRuby, interaktywny interpreter Rubyego via www (http://tryruby.org/)
- TheChangelog, agregator blogów związanych w dużej mierze z tworzeniem aplikacji internetowych w oparciu o Rubyego i Rails (http://thechangelog.com/)

SOFTWARE DEVELOPER komp

Zwięzła składnia, obiektowy i ściśle typowalny język programowania głównie używany przy szybkim (rapid) tworzeniu lub prototypowaniu aplikacji internetowych przy użyciu frameworka RubyOnRails. Dostępne implementacje dla JVM i CLR.NET.

Nie jest mocno zoptymalizowany pod względem prędkości czy zarządzania pamięcią. Otwartość klas może ułatwiać powodowanie nieporządanych efektów przy pracy wielu osób nad jedną aplikacją/systemem.

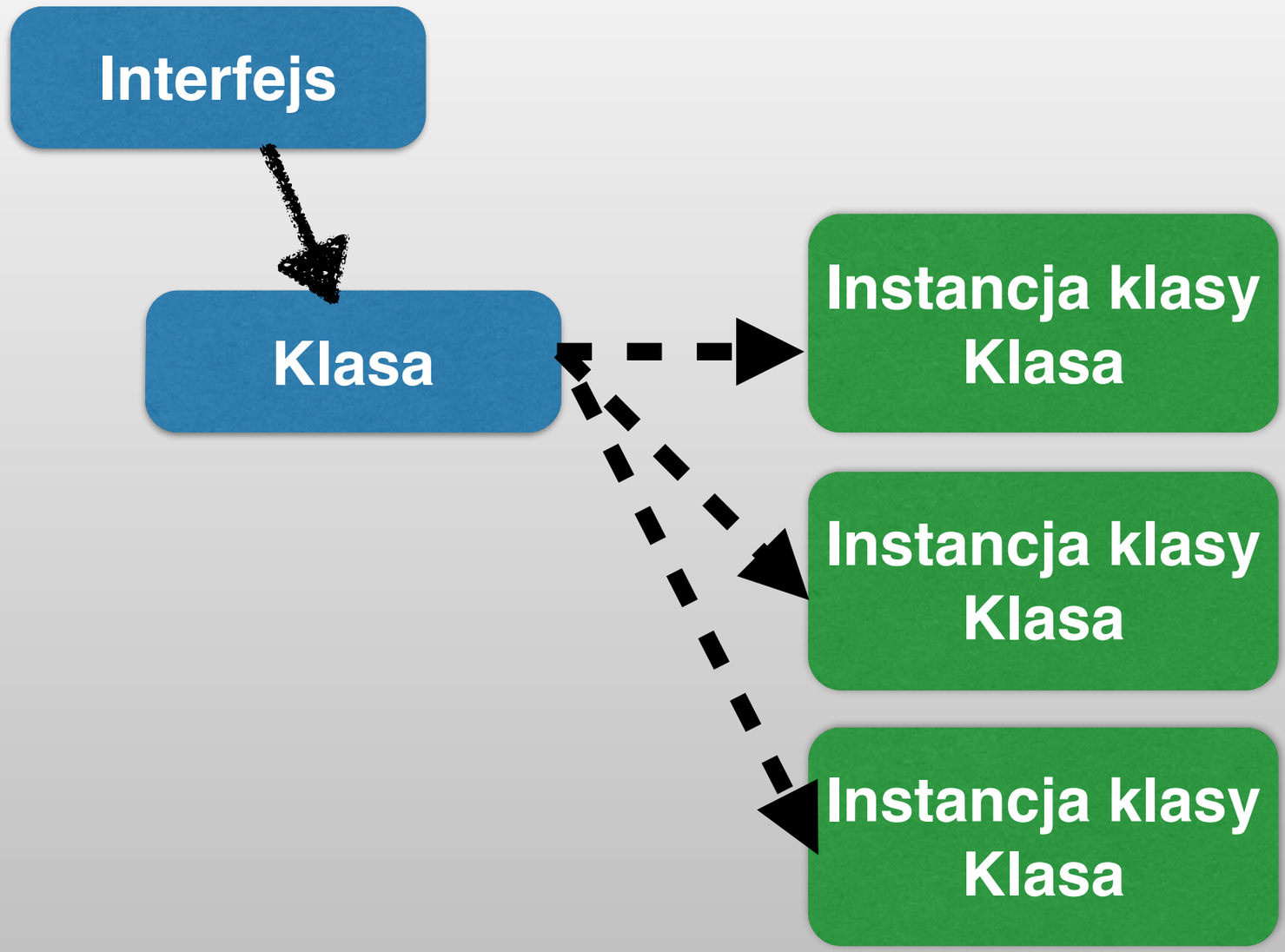
Języki prototypowalne

- **JavaScript** (ECMAScript)
  - Każdy powinien go znać ;-)
  - Występuje we wszystkich znanych przeglądarkach,
  - Posiada implementacje dla JVM oraz CLR.NET
  - Można też użyć do bezpośrednio do tworzenia szybkich sieciowych aplikacji przy użyciu Node (http://nodejs.org)
- **Lua**
  - Posiada implementacje dla JVM, CLR.NET, C/C++
    (i wiele innych: http://lua-users.org/wiki/LuaImplementations)
- **Io** (http://iolanguage.org/)
  - Minimalna składnia, bardzo mała wirtualna maszyna, osadzalna w C, JVM, CRL.NET (używany między innymi przez studio Pixar)

Języki prototypowe

- ji

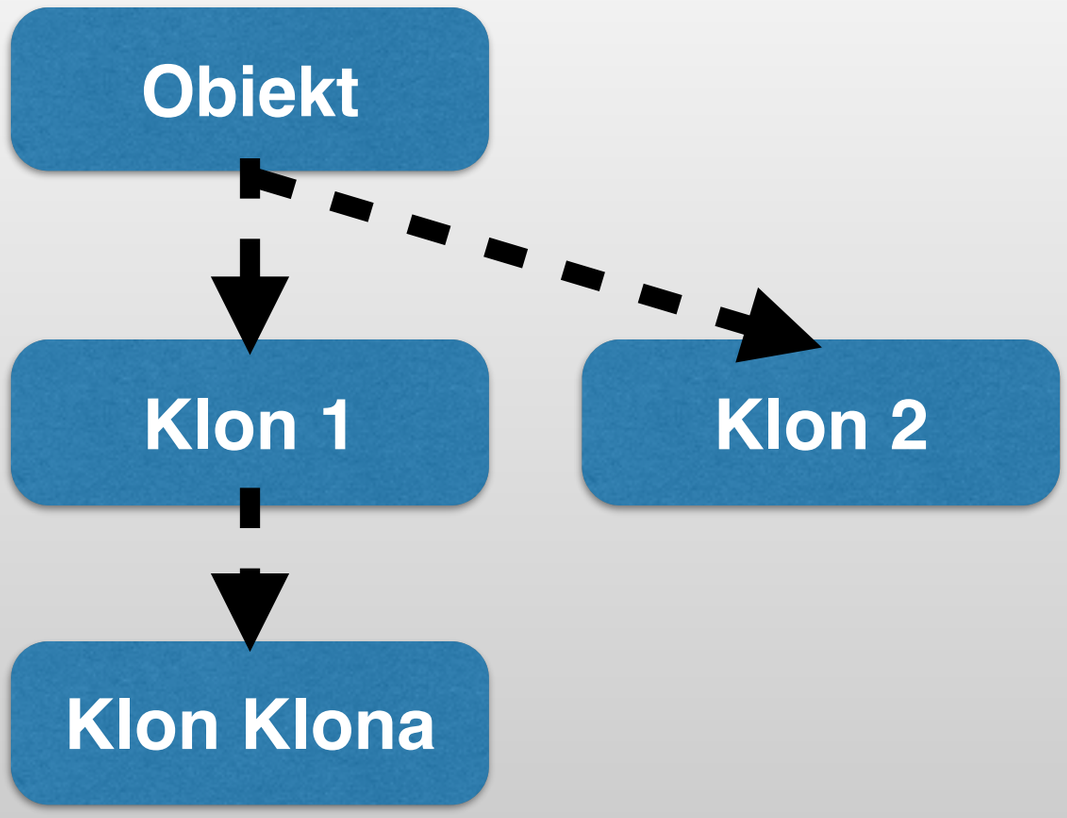
Dziedziczenie w językach obiektowych



- Minimalna składnia, bardzo mała wirtualna maszyna, osadzalna w C, JVM, CRL.NET (używany między innymi przez studio Pixar)

Języki prototypowe

Dziedziczenie w językach prototypowych



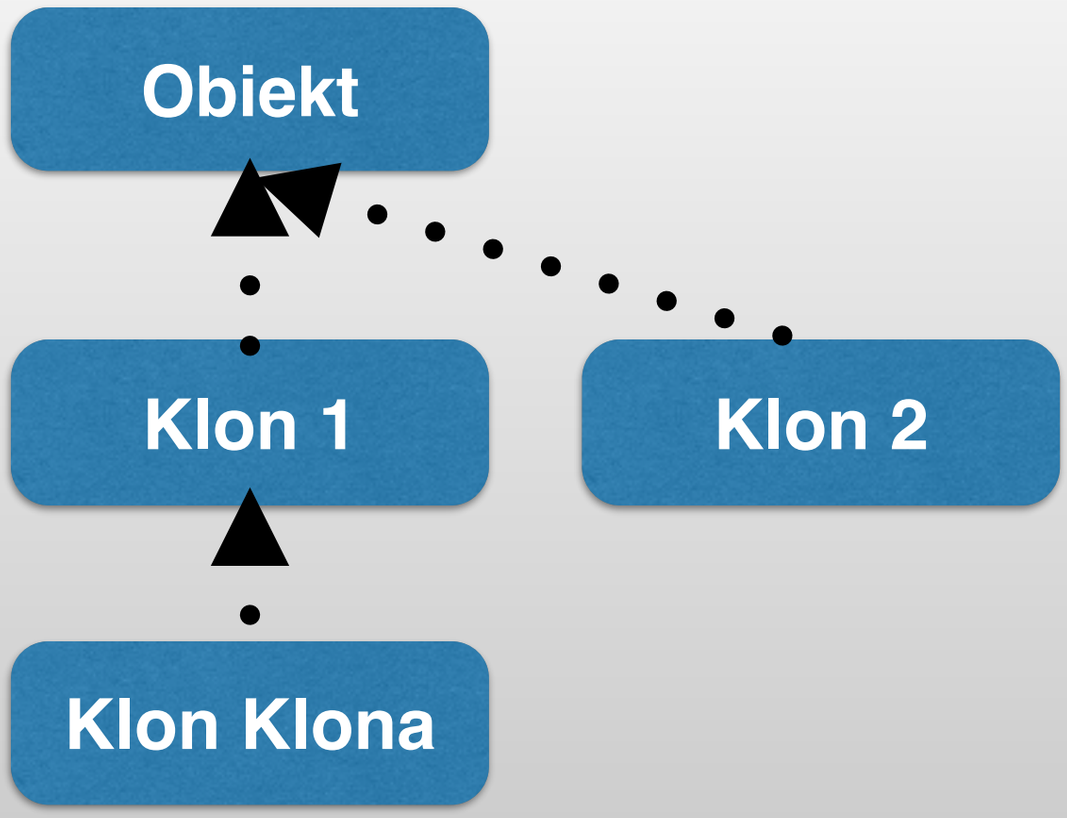
ji

- Minimalna składnia, bardzo mała wirtualna maszyna, osadzalna w C, JVM, CRL.NET (używany między innymi przez studio Pixar)

Języki prototypowe

Dziedziczenie w językach prototypowych

Każdy obiekt “zna” swój prototyp.



- Minimalna składnia, bardzo mała wirtualna maszyna, osadzalna w C, JVM, CRL.NET (używany między innymi przez studio Pixar)

Języki prototypowalne

- **JavaScript** (ECMAScript)
  - Każdy powinien go znać ;-)
  - Występuje we wszystkich znanych przeglądarkach,
  - Posiada implementacje dla JVM oraz CLR.NET
  - Można też użyć do bezpośrednio do tworzenia szybkich sieciowych aplikacji przy użyciu Node (http://nodejs.org)
- **Lua**
  - Posiada implementacje dla JVM, CLR.NET, C/C++
    (i wiele innych: http://lua-users.org/wiki/LuaImplementations)
- **Io** (http://iolanguage.org/)
  - Minimalna składnia, bardzo mała wirtualna maszyna, osadzalna w C, JVM, CRL.NET (używany między innymi przez studio Pixar)

Języki prototypowalne

- **JavaScript** (ECMAScript)
  - Każdy powinien
  - Występuje we w
  - Posiada implem
  - Można też użyć
    przy użyciu Nod
- **Lua**
  - Posiada implem
    (i wiele innych:
- **Io** (http://iolangua
  - Minimalna składnia, bardzo mała wirtualna maszyna, osadzalna w C, JVM, CRL.NET (używany między innymi przez studio Pixar)

Erlang

- zaprojektowany w firmie Ericsson w roku 1986, wydany jako open source. Nazwany na cześć matematyka A. K. Erlanga.

- **język funkcyjny**, z automatycznym zarządzaniem pamięcią, zaprojektowany dla **współbieżności** oraz pracy w **środowisku rozproszonym**, wczesną ewaluacją argumentów wiadomości, wspiera **hot-swap**owalność kodu, **kompilowany** do bytecode-u

- Używany przez Amazon (simpleDB), Facebook (chat), T-Moble (sms, autoryzacja użytkowników), Motorola oraz oczywiście Ericsson.

- Większe projekty open source używające Erlanga: ejabberd, **CouchDB**, **Riak**, RabbitMQ (Advanced Message Queuing Protocol)

Erlang

- zaprojektow
  source. Naz

- **język funkc**
  zaprojektow
  rozproszon
  wspiera **hot**

- Używany pr
  (sms, autory
  Ericsson.

- Większe pro
  CouchDB, I
  Protocol)

http://bi.gazeta.pl/im/ef/a9/ed/z15575535Q,Zamieszki-w-Caracas--W-Wenezueli-zginelo--juz-18-o.jpg

Erlang

Erlang:

```erlang
fac(0) -> 1;
fac(N) when N > 0  -> N*fac(N-1).
```

Ruby:

```ruby
def fac(n)
  n < 1 ? 1 : n*fac(n-1)
end
```

Erlang

Erlang:

```
1> c(fac).
{ok,fac}
2> fac:fac(100000).
28242294079603478742934215780245355184774949260912248505789180865429779509010630178725517714138311636107136117373619629514749961831
2
[…]
```

Ruby:

```
irb(main):071:0> fac(9000)
SystemStackError: stack level too deep
  from /System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0/irb/workspace.rb:86
```

Erlang

```
8> Zmienna = 3.
3
9> Zmienna.
3
10> Zmienna = 5.
** exception error: no match of right hand side value 5
```

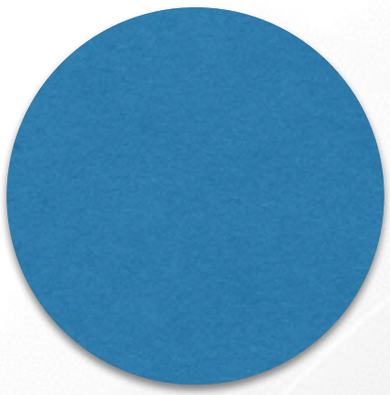
Erlang

```
8> Zmienna = 3.
3
9> Zmienna.
3
10> Zmienna = 5.
** exception error: no match of right hand side value 5
```

Nie znajdziemy w nim również
If-ów ani While-ów.

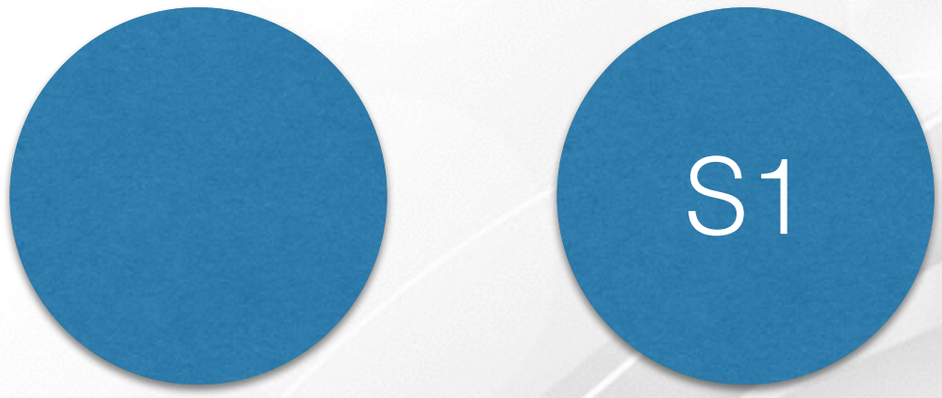
Erlang

```
S1 = spawn(modul, funkcja, []).
```



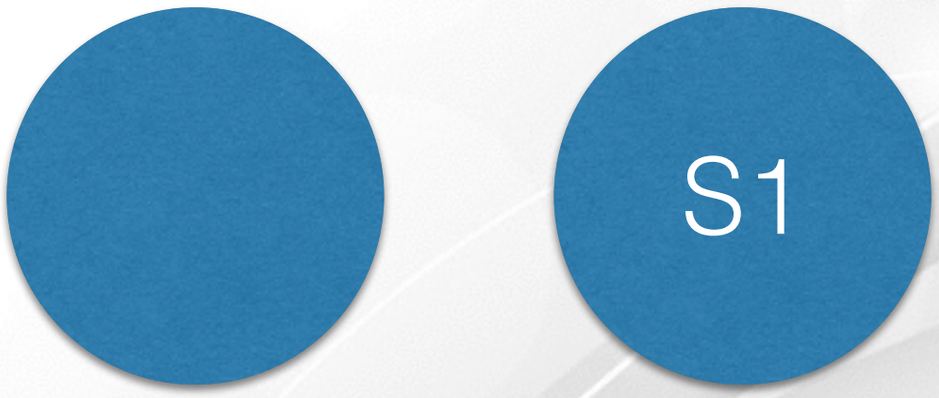
Erlang

```
S1 = spawn(modul, funkcja, []).
```



Erlang

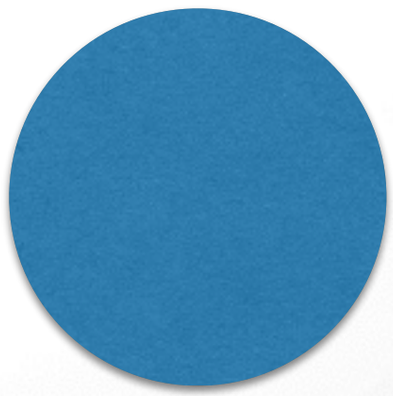
```
S1 = spawn(modul, funkcja, []).
```



Nowe procesy są “lekkie”,
w praktyce użycie pamięci
dla nowego procesu zaczyna
się w okolicach 1.5 KiB.

Erlang

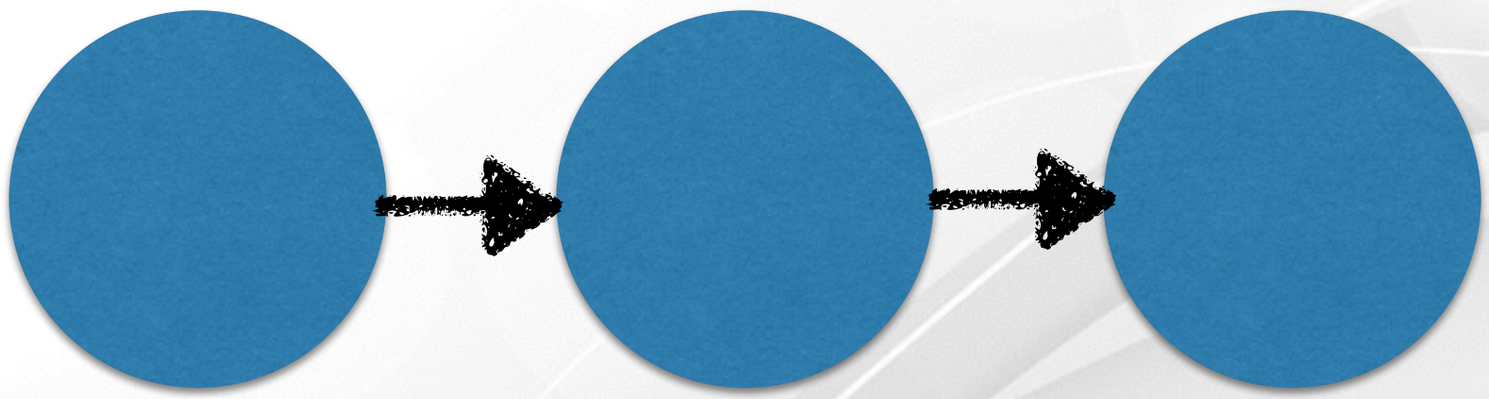
```
S2 = spawn_link(fun() … end).
```



```
{Pid, Ref} = spawn_monitor(fun() … end).
```

Erlang

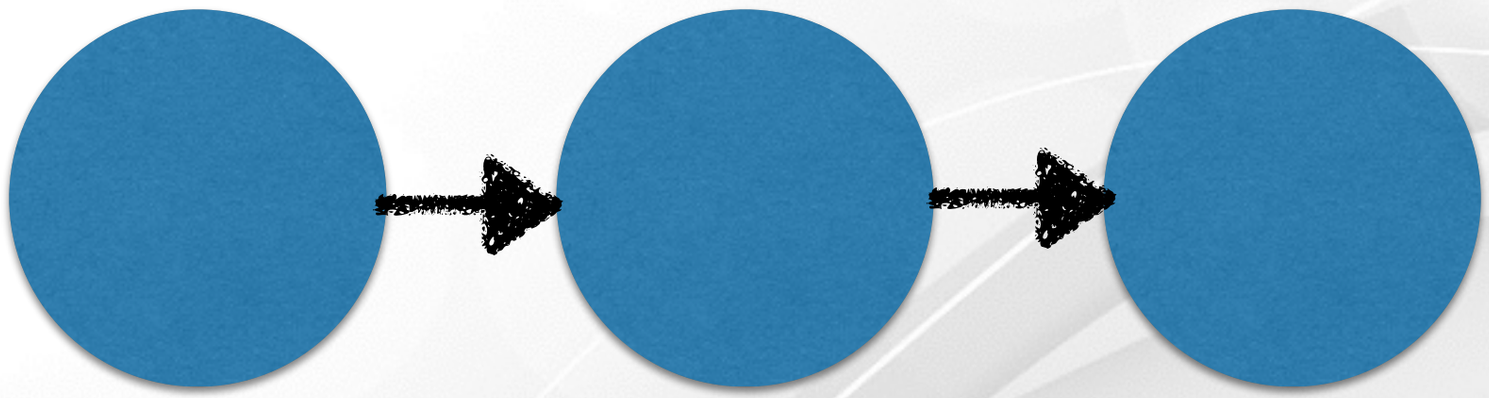
```
S2 = spawn_link(fun() … end).
```



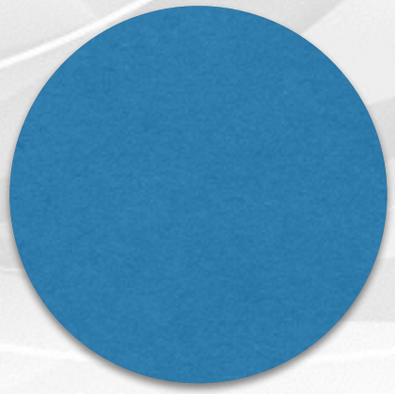
```
{Pid, Ref} = spawn_monitor(fun() … end).
```

Erlang

```
S2 = spawn_link(fun() … end).
```

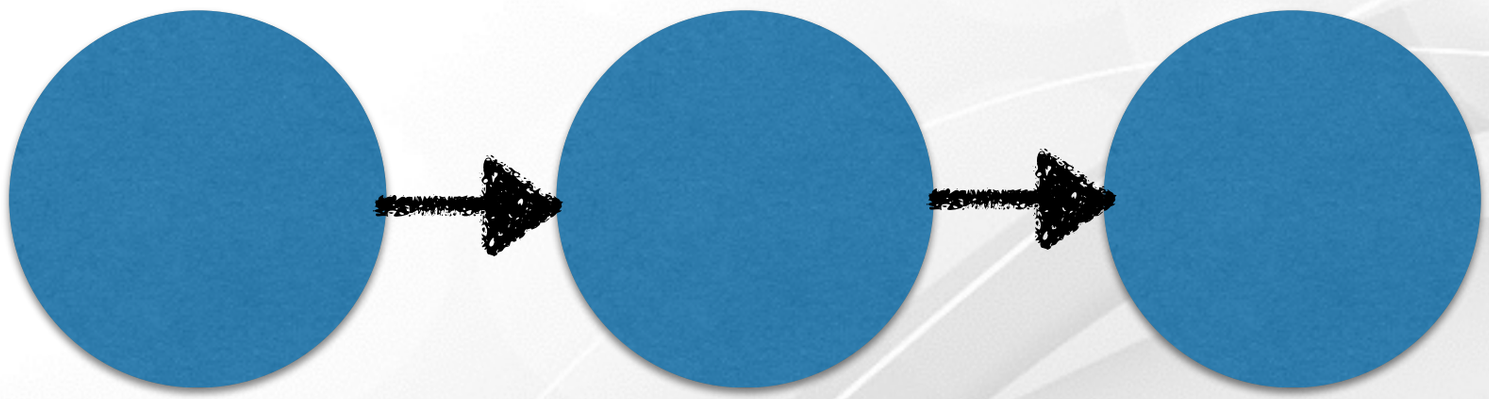


```
{Pid, Ref} = spawn_monitor(fun() … end).
```

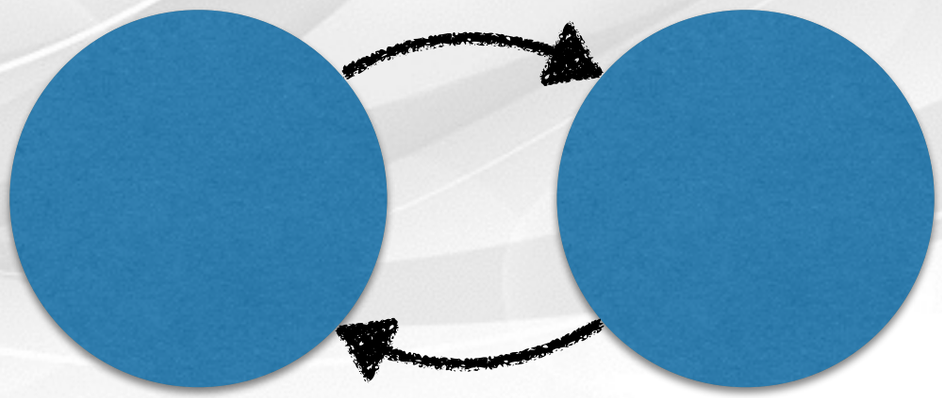


Erlang

```
S2 = spawn_link(fun() … end).
```



```
{Pid, Ref} = spawn_monitor(fun() … end).
```



Erlang

- Erlang, oficjalna strona języka (http://erlang.org)
- Try Erlang, interaktywny interpreter dostępny przez www (http://www.tryerlang.org/)
- Learn You Some Erlang for Great Good, tutorial dla Erlanga (http://learnyousomeerlang.com/)
- Ciekawy case przepisania serwerowego oprogramowania w C++ na Erlanga, ostatecznie codebase skrócił się o ~75% (http://www.metabrew.com/article/rewriting-playdar-c-to-erlang-massive-savings)

SOFTWARE DEVELOPER komp

Stworzony do przetwarzania współbieżnego oraz rozproszonego dla ogromnych ilości klientów/połączeń (lekkie procesy zamiast wątków). Funkcyjność wraz z niezmiennymi strukturami danych chronią przed najczęstszymi problemami systemów współbieżnych. Sprawdzony w systemach posiadających >100 milionów użytkowników.

Składnia języka jest obca dla większości programistów, nie posiada implementacji na innych wirtualnych maszynach (tj. JVM), przez co trudniej go zastosować.

SOFTWARE DEVELOPER komp

Lisp

```
(def LISP
  (fn [ & args ]
    (println args)))
```

```
(LISP
    '(- rodzina języków programowania)
    '(- zaprojektowany w 1958 przez Johna McCarthy-ego)
    '(- jeden z najstarszych języków programowania
        wysokiego poziomu, starszy jest tylko Fortran)
  '(- bazuje na rachunku lambda (lambda calculus))
  '(- istnieje wiele dialektów, odmian Lisp-a
     '( najpopularniejsze to Common Lisp oraz Scheme)
     '( a aktualnie również bazujący na JVM/CRL/JS Clojure)))
```

(def LISP

(fn [& args]

(p ln args)))

Lisp

(LISP

'(- rodzina języków programowania)

'(- zaprojektowany w 1958 przez Johna Mc thy-ego)

'(- jeden z **najstarszych** języków programow ia

wysokiego poziomu, starszy jest tylko Fortra

'(- bazuje na rachunku lambda (lambda calculus))

'(- istnieje wiele dialektów, odmian Lisp-a

'(najpopularniejsze to Common Lisp oraz Scheme)

'(a aktualnie również bazujący na JVM/CRL/JS Clojure)))

Lisp

(LISP

'(- posiada zarządzanie pamięcią (GC), od ~1960)

'(- wsparcie dla closures (po "akademicku" domknięcie),
wprowadził jako pierwszy język na początku lat 70)

'(- makra, które jako pierwszy wprowadził w latach 60,
w praktyce występują tylko w językach funkcyjnych)

'(- Lots of Irritating Superfluous Parentheses)

'(- LISP Is Simply Perfect))

Clojure

- Dialekt Lisp zaimplementowany w całości na JVM oraz CLR i JavaScript.
- Posiada dobrą interoperacyjność z Javą/.NET/JS i może wykorzystywać wszystkie ich biblioteki.
- Używany przez Akamai, Citi Bank, TypeWire, AutoDesk oraz coraz więcej startup-ów na całym świecie.
- Paul Graham stworzył w czystym Common Lispie projekt, który stał się później platformą Yahoo! Store, dzięki Lispowi twierdzi, że jego zepół mógł wyprzedzać całą konkurencję jaką wtedy posiadali (http://paulgraham.com/avg.html)

SOFTWARE DEVELOPER komp

Clojure

```
(+ 2 3) 5
(/ 3 4) 3/4
(= 4 (+ 2 2)) true

(defn factorial [n]
    (reduce *
        (range 1N (+ 1N n)))) #'LISP/factorial

(factorial 100000) 2824229407960347874293421578024535510
```

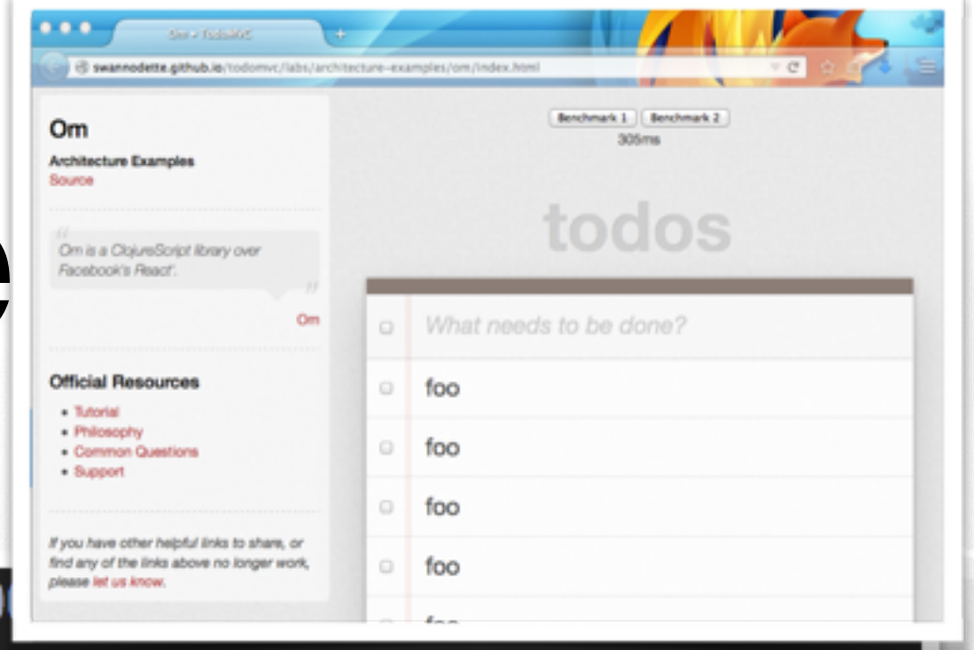
SOFTWARE DEVELOPER komp

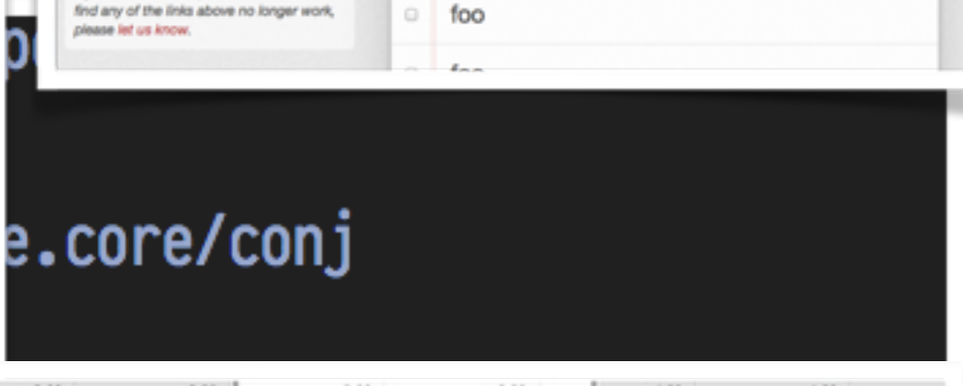
Clojure

```
;; funkcje otrzymują argumenty przetworzone (poddane ewaluacji),
;; makra otrzymują argumenty jako dane.

;; http://clojuredocs.org/clojure_core/clojure.core/conj
(defmacro postfix-notation
  [expression]
  (conj (butlast expression) (last expression))) #'LISP/postfix-notation

(postfix-notation (1 1 +)) 2

(butlast '(1 1 +)) (1 1)
(last '(1 1 +)) +
(conj '(1 1) '+) (+ 1 1)
```

SOFTWARE DEVELOPER komp

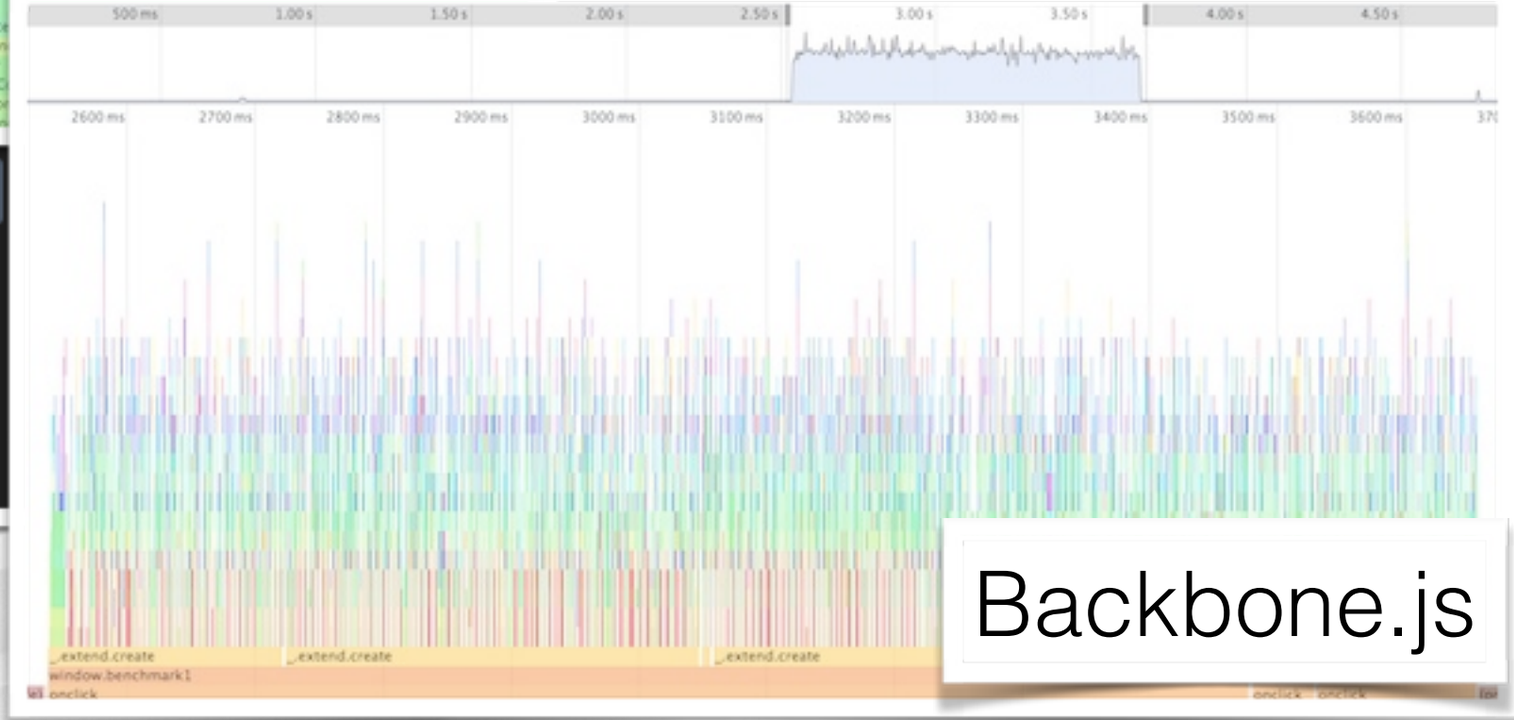
Clojure





Clojure/Om

```
(postfix-notation (1 1 +)) 2

(butlast '(1 1 +)) (1 1)
(last '(1 1 +)) +
(conj '(1 1) '+) (+ 1 1)
```



Backbone.js

http://swannodette.github.io/2013/12/17/the-future-of-javascript-mvcs/

Lisp/Clojure Scheme

- Książki:
  - Land of lisp (http://landoflisp.com)
  - Realms of Ratchet, dla dialektu Scheme (http://realmofracket.com/)
- Dr. Racket, współczesny dialekt Scheme (http://racket-lang.org/)

- Clojure (http://clojure.org)
- Lista interpreterów Common Lispa, w tym implementacja dla JVM (http://articulate-lisp.com/env:lisp-system.html)
- LightTable, eksperymentalne IDE dla Clojure (http://www.lighttable.com/)
- Interaktywny tutorial dla Clojure i IDE LightTable (https://github.com/swannodette/lt-cljs-tutorial)

SOFTWARE DEVELOPER komp

Tworząc programy pisze się sporo kodu, który wygląda podobnie, dzięki mechanizmom takim jak makra w lispie można sprawić aby ten kod miał trzy linie.

Nie znana składnia oraz wymagana miłość do nawiasów ;-)

> “Lisp is worth learning for the profound enlightenment experience you will have when you finally get it; that experience will make you a better programmer for the rest of your days, even if you never actually use Lisp itself a lot.”
>
> — Eric Raymond

Game Dev

- C#
  - Unity 3D (https://unity3d.com/)
- C++ / własne
  - UDK (http://udk.com/)

- Java (subset)
  - Android
- iOS
  - ObjC (XCode dostępny z Mac OS X)

Języki “ezoteryczne”

Języki “ezoteryczne”

Whitespace

https://github.com/hostilefork/whitespacers/

Języki “ezoteryczne”

B****fuck

http://esolangs.org/wiki/Brainfuck

```
++++++++[>++++[>++>+++>+++>+<<<<-]>+>+>->>+[<]<-]>>.>---.+++++++..
+++.>>.<-.<.+++.------.--------.>>+.>++.
```

Inne warte poznania języki

- **Python**, dynamiczny język, posiadający sporą ilość bibliotek, implementacje osobne dla JVM oraz CLR.NET (http://python.org)
- **Groovy**, dynamiczny język na JVM, zainspirowany Smalltalkiem, Rubym i Pythonem (http://groovy.codehaus.org/). Posiada odpowiednik ruby on rails: Grails (http://grails.org/)
- **Scala**, funkcyjno-obiektowy język na JVM (http://scala-lang.org/)
- **Go**, język ze stajni Google (http://go-lang.org/)
- **Haskell**, czysto funkcyjny język, używany przez Google, Facebook, Bank of America
- **F#** funkcyjny na CLR.NET (http://fsharp.org/)
- **Smalltalk**, implementacja Smalltalka Squeak (http://www.squeak.org/)
- **AspectJ**, nie do końca język sam w sobie ale pozwala na pewną dozę meta programowania w Javie (http://eclipse.org/ajdt/)

Na zakończenie…

Na zakończenie…