

language
aesthetics

Sam Aaron



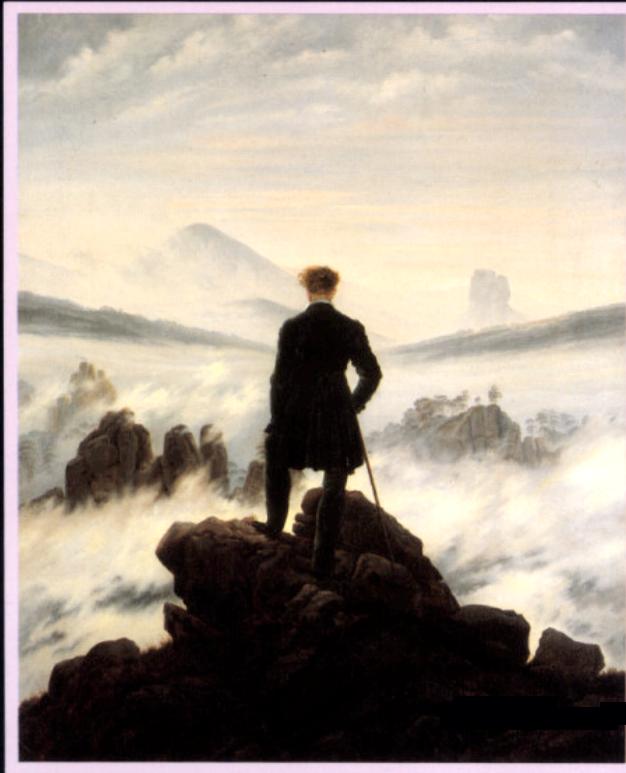
INNOVATION FACTORY







THE IDEOLOGY OF THE AESTHETIC



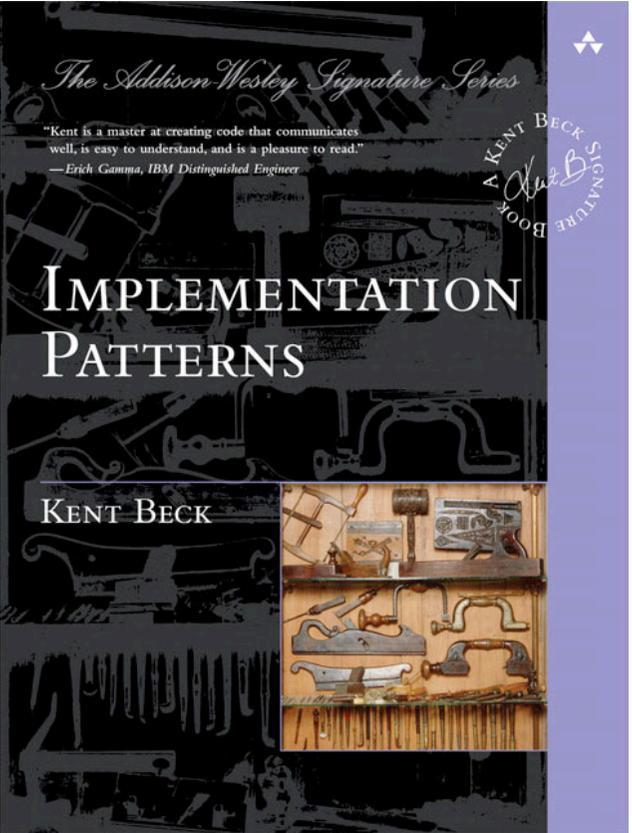
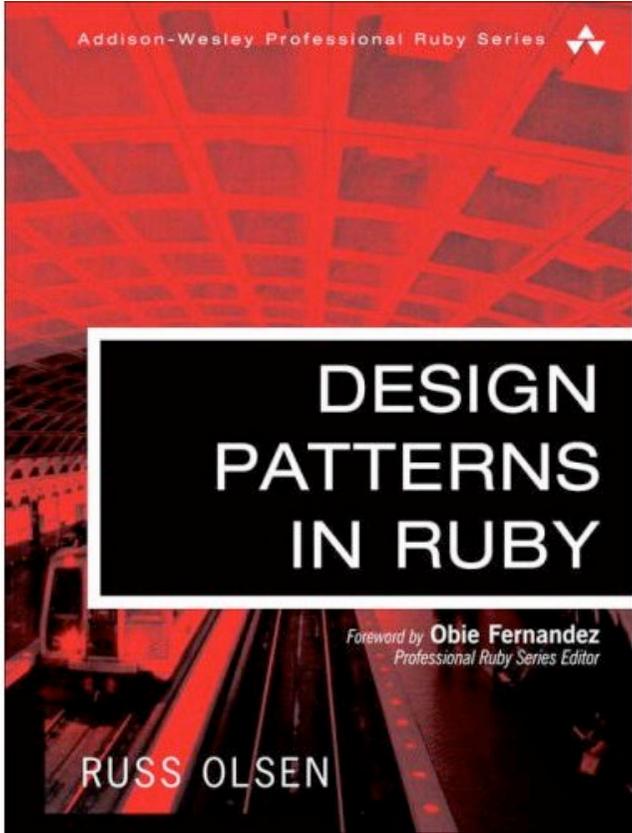
Terry Eagleton

ΤΕΡΡΥ ΕΓΚΛΕΤΟΝ



"The aesthetic is at once, as I try to show, the very secret prototype of human subjectivity in early capitalist society, and a vision of human energies as radical ends in themselves which is the implacable enemy of all dominative or instrumentalist thought".

Terry Eagleton

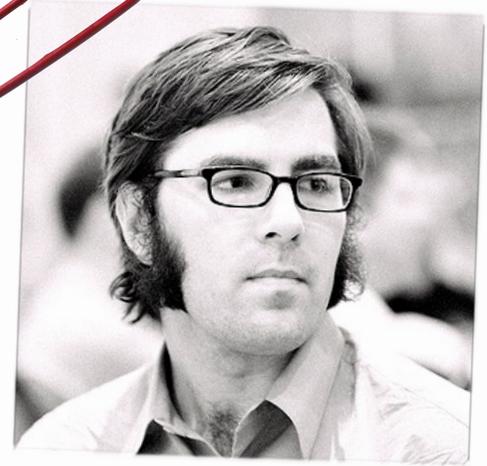


aes•thet•ic | es' θ etik | (also **es•thet•ic**)

adjective

concerned with beauty or the appreciation of beauty : *the pictures give great aesthetic pleasure.*

- giving or designed to give pleasure through beauty; of pleasing appearance.

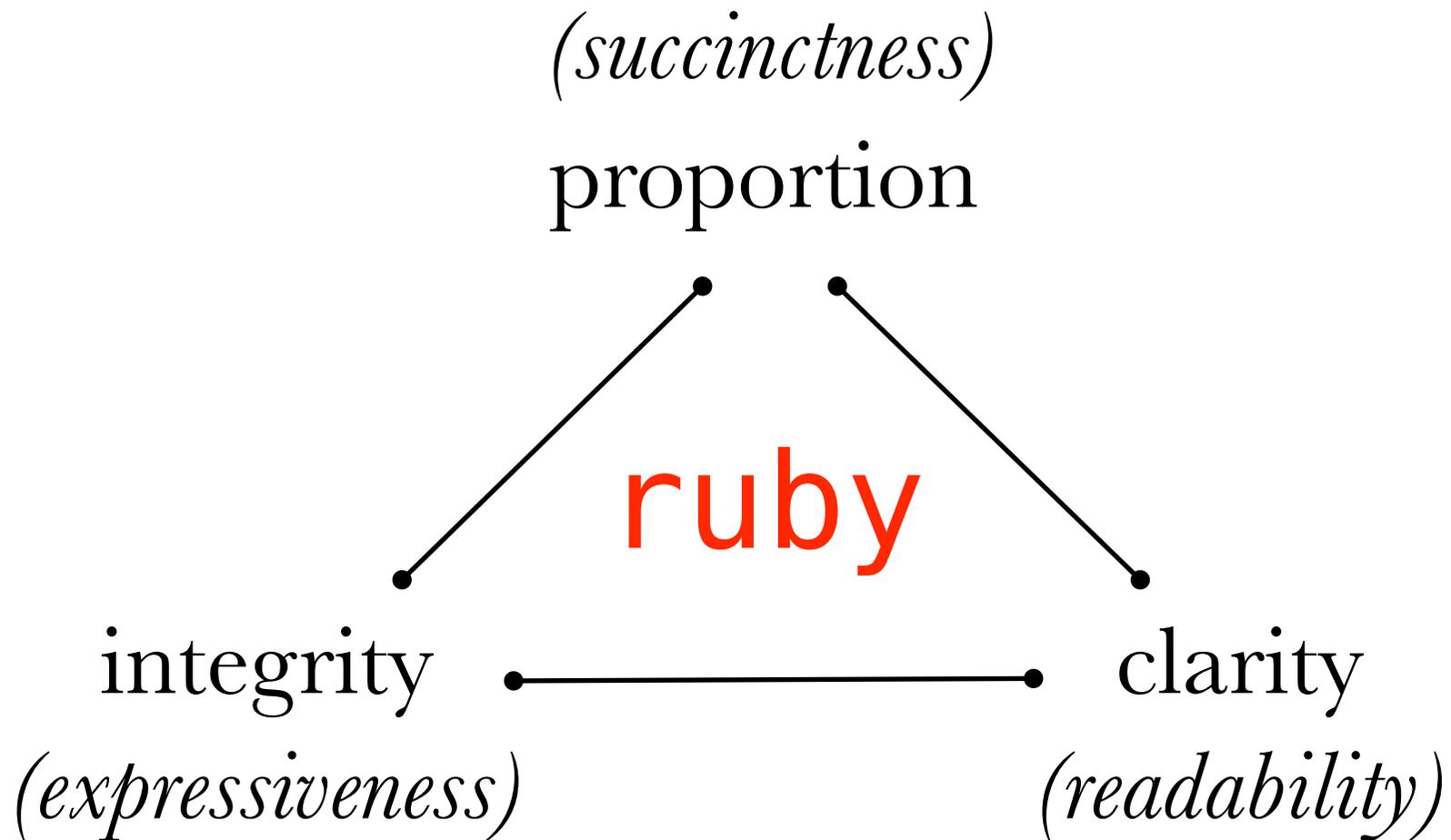


“Good software = beautiful software”
Marcel Molina Jr

Thomas Aquinas



Thomas Aquinas: *definition of beauty*



λ

λ







disoriented?

why the lucky stiff
is a fledgling freelance professor, one who
will die young and make no lasting
impression.

except there was that time when I
vacuumed all of Greenland for them.

email? here.

Sunday, April 17, 2005

Seeing Metaclasses Clearly

If you're new to metaprogramming in Ruby and you'd like to start using it, perhaps these four methods could give you a bit more vision.

```
class Object
  # The hidden singleton lurks behind everyone
  def metaclass; class << self; self; end; end
  def meta_eval &blk; metaclass.instance_eval &blk; end

  # Adds methods to a metaclass
  def meta_def name, &blk
    meta_eval { define_method name, &blk }
  end

  # Defines an instance method within a class
  def class_def name, &blk
    class_eval { define_method name, &blk }
  end
end
```

Books:

- [Why's \(Poignant\) Guide to Ruby](#) (Blbxy Tees)

Blogs:

- [quiet](#)
- [Hackety Org](#)
- [RedHanded](#) COMPLETED!

Stories:

- [The Bobby Wolves](#)
- [Stunt Runner](#)

Comics:

- [Holy Bible: one, two, three, four, five, six, seven, eight, nine, ten, eleven, twelve, thirteen, fourteen, fifteen, sixteen](#)
- [The Open Window: one, two, three, four, five, six, seven, eight, nine](#)
- [Lost Children, Lost Children, Lost Children, See Such Feverish Castles: one](#)

[Incidents: \(35\) Adventure of the Apple's Mom \(34\) A Magic Milk \(33\) The Secret Sandwich \(32\) A Smart Curtsy \(31\) The Hand Which Fell Apart \(30\) Gentle & Pailey \(29\) The Grieving Boar \(28\) The Story Life Doesn't Explain \(27\) The Life Guy \(26\) The Jump Wanter \(25\) Kimothy's Mouth \(24\) Speaking Of Flutes \(23\) The Advisor \(22\) Wristwatches \(21\) The Queen-Sized \(20\) The Tandem Bicycle \(19\) The Little Piece of Cloth \(18\) Milk Powers \(17\) Javek and the Candle \(16\) The Skier \(15\) The Man Who Happened to Have Legs \(14\) The Man Who Happened](#)

```
end
end
class_eval { define_method name, &blk }
def class_def name, &blk
```



“This is some serious mojo”

Harm Aarts

```
:require 'open-uri'; require 'rubygems'; require 'hpricot'
url, save_as = *ARGV
KEEPVID = URI(Hpricot(URI("http://keepvid.com/").read).
  at("form")['action'])
```

```
Net::HTTP.start(KEEPVID.host, KEEPVID.port) do |http|
  doc = Hpricot(http.post(KEEPVID.path, "url=#{url}&site=aa").body)
  url = URI((doc/:%a).detect{|x|x.next_node.to_s.index("%flv - Flash")}['href'])
end
```

```
Net::HTTP.start(url.host, url.port) do |http|
  http.request_get(url.request_uri, {'User-Agent' => 'ruby', 'Host' => url.host}) do
    |r|
```

```
File.open(save_as, 'wb') do |f|
```

```
  len = 0
  r.read_body do |chunk|
```

```
    f.write(chunk)
```

```
    len += chunk.length
```

```
    print "%30s / %d of %d / %0.1f%%\r" %
```

```
      [save_as, len, r.content_length, ((len * 1.0) / r.content_length) * 100]
```

```
  end
```

```
end
```

```
end
```

```
end
```

```
$ruby.is_a?(Object){|oriented| language}
```

```
def you are  
  false  
end
```

```
enjoy life while you /can/
```

```
"eyes".scan /the_darkness/  
catch( :in_the_wind ) { ?a.round; "breath" \  
  or "a".slice /of_moon/ }
```

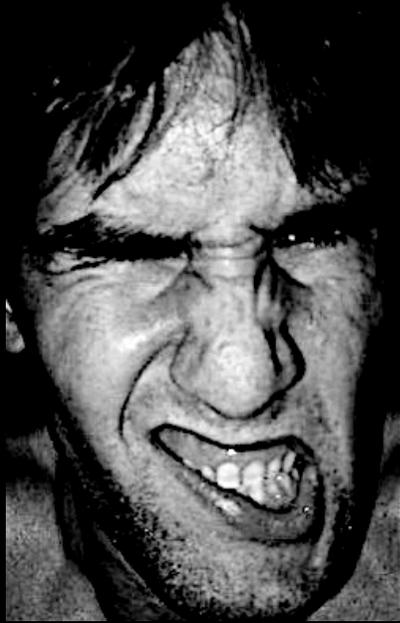
```

%w[tempfile uri].map{|l|require l};class Object;def meta_def m,&b;(class<<self
self end).send(:define_method,m,&b end end;module Camping;C=Self
S=IO.read(__FILE__)rescue nil;P=<h1>Cam\ping Problem!</h1><h2>4s</h2>
class H<flash
def method_missing m,*a;m.to_s==~/self\$/?a[0]:a==[]?self[m.to_s]:super end
alias u merge!;undef id,type;end;module Helpers;def R c,*g
p,h=/(.+?)/.g.grep(Hash);g->h;raise"bad route"unless u=c.urls.find{|x|
break x if x.scan(p).size==g.size&&/%{x}\$/!==(x.g.inject(x){|x,a|
x.sub p,C.escape({a[a.class.primary_key]rescue a})}})
h.any?? u+"?"+h[0].map{|x|x.map{|z|C.escape z}=="?":?":u end;def / p
p[/^\/\/?@root+p:p end;def URL c=/'/:a;c=R(c,*a) if c.respond_to?(:urls
c=self/c;c=~/^+@env.HTTP_HOST+c if c[/^\/\/?@URI c end end;module Base
attr_accessor :input,:cookies,:env,:headers,:body,:status,:root;Z="\r\n"
def method_missing *a,&b;a.shift if a[0]==:render;m=Mab.new({},self)
s=m.capture(send(*a,&b));s=m.capture(send(:layout){s})if/^_/_/!-a[0].to_s and
n.respond_to?(:layout);s end;def r s,b,h={};@status=s;headers.u(h);@body=b
end;def redirect *a;r 302,'','Location'=>URL(*a)end;def r404 p=env.PATH
r 404,P+"#{p} not found"end;def r500 k,m,x
r 500,P+"#{k}.#{m}"+<h3>#{x.class} #{x.message}: <ul>#{x.
backtrace.map{|b|<li>#{b}</li>}</ul></h3>"end;def r501 m=@method
r 501,P+"#{m.upcase} not implemented"end;def to a
[status,body,headers]end;def initialize r,e,m;@status,@method,@env,@headers,
@root=200,m,e,H['Content-Type','text/html'],e.SCRIPT_NAME.sub(/\/\$/,'')
@k=C.kp e.HTTP_COOKIE_TYPE;q=C.qsp e.QUERY_STRING;@in=r;case e.CONTENT_TYPE
when/\r\multipart/form-.*boundary=\^7\{[";,:]+\}n
b=/(?:\r?\n|\A)#(Regexp.quote"--#%1"?(?:--)?\r?$/;until@in.eof?;fh=H[]
for l in@in;case l;when Z;break;when/^Content-D.+?: form-data;/
fh.u H["$".scan(/(?:\s\w+)=("[^"]+)"\)/).flatten]
when/^Content-Type: (.+?)(\r?\n)/m: fh.type = $1 end end;fn=fh.name
o=if fh.filename;o=fh.tempfile=Tempfile.new(:C);o.binmode;else;fh="";end;s=@192
k='';l=@in.read(s*2);while l;if(k<<l)==b;o<<$'.chomp
@in.seek(-$'.size,IO::SEEK_CUR);break end;o<<k.slice(0..s);l=@in.read(s) end
C.qsp(fn,'&';',fh,q)if fn;fh.tempfile.rewind if fh.is_a?H end;when
"application/x-www-form-urlencoded": q.u(C.qsp(@in.read))end
@cookies,@input=@k.dup,q.dup end;def service *a;@body=send @method,*a
headers['Set-Cookie']=cookies.map{|k,v|"%{k}=%{C.escape(v)}; path=#{self/
}/"}*if v!=@k[k]-[nil];self end;def to_s;"Status: #{@status}#{Z+{headers.inject{|
|}{|a,o|[%o[1]].map{|v|a<<[o[0],v]}:" if v&&v.to_s.any?};a}Z}+Z}#{@body}end
end;X=module Controllers;@r=[];class<<self;def r;@r end;def R *u;r=@r
Class.new(meta_def(:urls){u};meta_def(:inherited){|x|r<<x})end
def D p,m;r.map{|k|k.urls.map{|x|return(k.instance_method(m)rescue nil)?
[k,m,"$-[1..-1]":[I,'r501',m]if p~/^%{x}\$/};[I,'r404',p]
end;def M;def M;end;constants.map{|c|k=const_get(c)
k.send(:include,C::Base::Helpers::Models;@r=[k]+r if r-[k]==r
k.meta_def(:urls){["/%{c.downcase}"]}if k.respond_to?(:urls)end end;class I<R()
end;self end;class<<self;def goes m
eval S.gsub(/Camping/,m.to_s).TOPLEVEL_BINDING end;def escape s
s.to_s.gsub(/[\^ \w.-]+/n){'%'+(%$>unpack('H2'%$>.size)**%$>).upcase}.tr' ','+'
end;def un s;s.tr(' ','').gsub(/%{[\da-f]{2}}/in){[%$>].pack'H2'}end
def qsp q,d='&';y=nil,z=R[];m=proc{|_,o,n|o.u(n,&m)rescue(/%><m)}
(q.to_s.split(/[#>]+/n)-[""]).inject({b,z,H[]}{|h,p|k,v=un(p).
split='',2;h.u k.split(/[\]\[\]+/).reverse.inject(y){|v|{x,1|H[i,x]};&m}end
def kp s;c=qsp s,'';end;def run r=$stdin,e=ENV;X=M;e=H[e.to_hash];k,m,*e=X.D e.
PATH_INFO=un("/#{@e.PATH_INFO}").gsub(/\/+/, '/'),
(e.REQUEST_METHOD||'get').downcase
k.new(r,e,m).Y.service(*a);rescue=>x;X:I.new(r,e,'r500').service k,m,x
end;def method_missing m,c,*a;X.M;k=X.const_get(c).new StringIO.new,
H['HTTP_HOST'],'',SCRIPT_NAME','','HTTP_COOKIE',''],m.to_s
H[a.pop].each{|e,f|k.send"@{e}=",f}if Hash==a[-1];k.service(*a)end end
module Views;include X::Helpers end;module Models;autoload:Base,'camping/db'
def Y;self;end end;autoload:Mab,'camping/mab'end

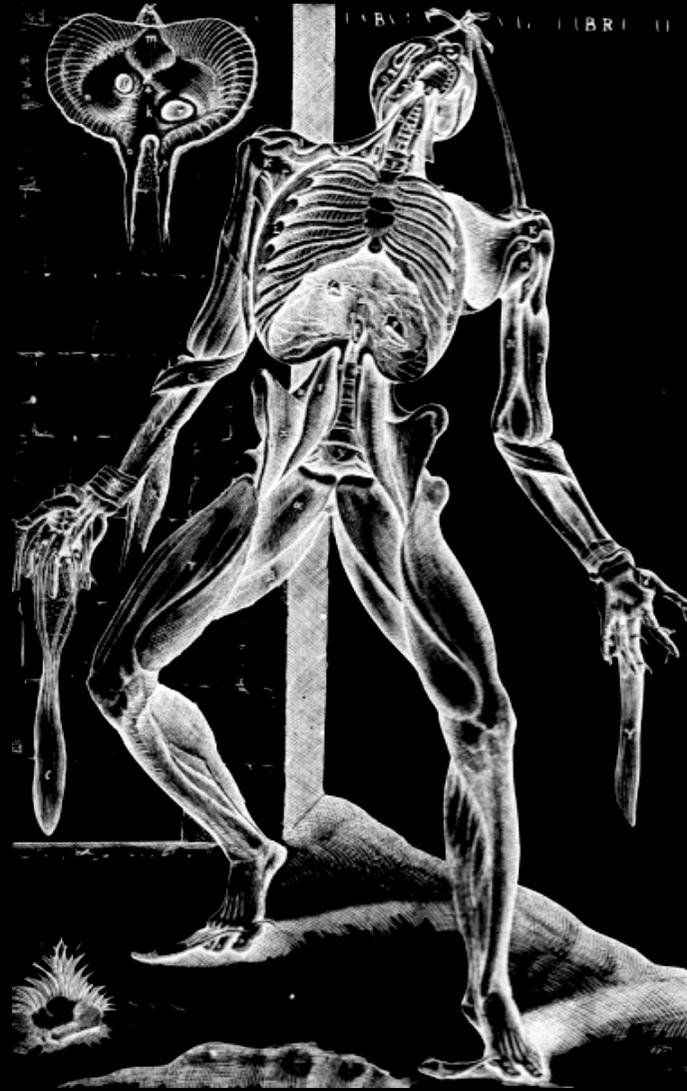
```

```
kw[tempfile uri].map{|l|require l};class Object;def meta_def m,&b;(class<<self
self end).send(:define_method,m,&b end end;module Camping;C=self
S=IO.read( __FILE__ )rescue nil;P=<html><html><ping Problem1</html><html2><html2>"
class H<hash
method_missing m,*a;m.to_s=~/$/?self[$]=a[0]:a=[]?self[m.to_s]:super end
alias u merge!;undef id,type;end;module Helpers;def R c,*g
p,h=/(.+?)/,g.grep(Hash);g-=h;raise"bad route"unless u=c.urls.find{|x|
break x if x.scan(p).size==g.size&&/#{x}\/?/==(x=g.inject(x){|x,a|
x.sub p,C.escape((a[a.class.primary_key]rescue a))}})-
h.any?? u+"?"+h[0].map{|x|x.map{|z|C.escape z}**"="}*"&":u end;def / p
p[/\//]?@root+p:p end;def URL c='/*a;c=R(c,*a) if c.respond_to?:urls
c=self/c:="//"+@env.HTTP_HOST+c if c[/\//]:URI c end end;module Base
attr_accessor:input,:cookies,:env,:headers,:body,:status,:root;Z="\r\n"
def method_missing *a,&b;a.shift if a[0]==:render;m=Web.new({},self)
s=m.capture{send(*a,&b)};s=m.capture{send(:layout){s}}if/^/l~a[0].to_s and
m.respond_to?:layout;s end;def r s,b,h={};@status=s;headers.u(h);@body=b
end;def redirect *a;r 302, '', 'Location'=>URL(*a)end;def r404 p=env.PATH
r 404,P%#{p} not found"end;def r500 k,m,x
r 500,P%#{k}.#{m}"+<h3>#{x.class} #{x.message}: <ul>#{x.
backtrace.map{|b| "<li>#{b}</li>"}</ul></h3>"end;def r501 m=@method
r 501,P%#{m.upcase} not implemented"end;def to_a
[status,body,headers]end;def initialize r,e,m;@status,@method,@env,@headers,
@root=200,m,e,H['Content-Type','text/html'],e.SCRIPT_NAME.sub(/\$/,'')
@k=C.kp e.HTTP_COOKIE;q=C.qsp e.QUERY_STRING;@in=r;case e.CONTENT_TYPE
when%r|\Amultipart/form-. *boundary="\"?([\^";,]+)|n
b=/(?:\r?\n|A)#Regexp.quote"--#1"?(?:-)?\r$/;until@in.eof?;fh=H[
for l in@in:case l;when Z;break;when/^Content-D.+?: form-data:/
fh.u H[*$'.scan(/(?:\s\w+)=("[\^"]+)/).flatten]
when/^Content-Type: (.+?) (\r$|Z)/m: fh.type = $1 end end;fn=fh.name
o=if fh.filename:o=fh.tempfile=Tempfile.new(:C):o.binmode;else;fh="";end;s=8192
k='';l=@in.read(s*2);while l;if(k<l)->b;o<<$`.chomp
@in.seek(-$.size,IO::SEEK_CUR);break end;o<<k.slice!(0..s);l=@in.read(s) end
C.qsp(fn,'&';fh,q)if fn;fh.tempfile.rewind if fh.is_a?H end;when
"application/x-www-form-urlencoded": q.u(C.qsp(@in.read))end
@cookies,@input=@k.dup,q.dup end;def service *a;@body=send @method,*a
headers['Set-Cookie']=cookies.map{|k,v|"*#{k}=##{C.escape(v)}; path=#{self/
"/"}"if v!=@k[k]}-[nil];self end;def to_s;"Status: #{@status}#{Z+(headers.inject([
]){|a,o|[*o[1]].map{|v|a<<[o[0],v]}*": "if v&&v.to_s.any?};a*Z)+Z+Z}#body"end
end;X=module Controllers;@r=[];class<<self;def r;@r end;def R *u;r=@r
Class.new{meta_def(:urls){u};meta_def(:inherited){|x|r<<x}}end
def D p,m;r.map{|k|k.urls.map{|x|return(k.instance_method(m)rescue nil)?
[k,m,*$-[1..-1]]:[I,'r501',m]if p=~/#{x}\/?$/}}:[I,'r404',p]
end;def M;def M;end;constants.map{|c|k=const_get(c)
k.send(:include,C,Base,Helpers,Models;@r=[k]+r if r-[k]==r
k.meta_def(:urls){|"/#{c.downcase}"}]if !k.respond_to?:urls}end end;class I<R()
end;self end;class<<self;def goes m
eval S.gsub(/Camping/,m.to_s),TOPLEVEL_BINDING end;def escape s
s.to_s.gsub(/[\^ \w.-]+/n){'%'+($$.unpack('H2'*$$.size)*'%').upcase}.tr ' ','+'
end;def un s;s.tr(' ','').gsub(/%([\da-f]{2})/in){[$1].pack'H*'}end
def qsp q,d='&';y=nil,z=H[];m=proc{[_ ,o,n|o.u(n,&m)rescue[*o]<<n]}
q.to_s.split(/#{d}+ */n)-[""].inject((b,z=z,H[])[0]){|h,p|k,v=un(p).
split '=' ,2;h.u k.split(/[\]\[\]\+\/).reverse.inject(y|v){|x,i|H[i,x]},&m}end
def kp s;c=qsp s,'';end;def run r=$stdin,e=ENV;X.M;e=H[e.to_hash];k,m,*a=X.D e.
```

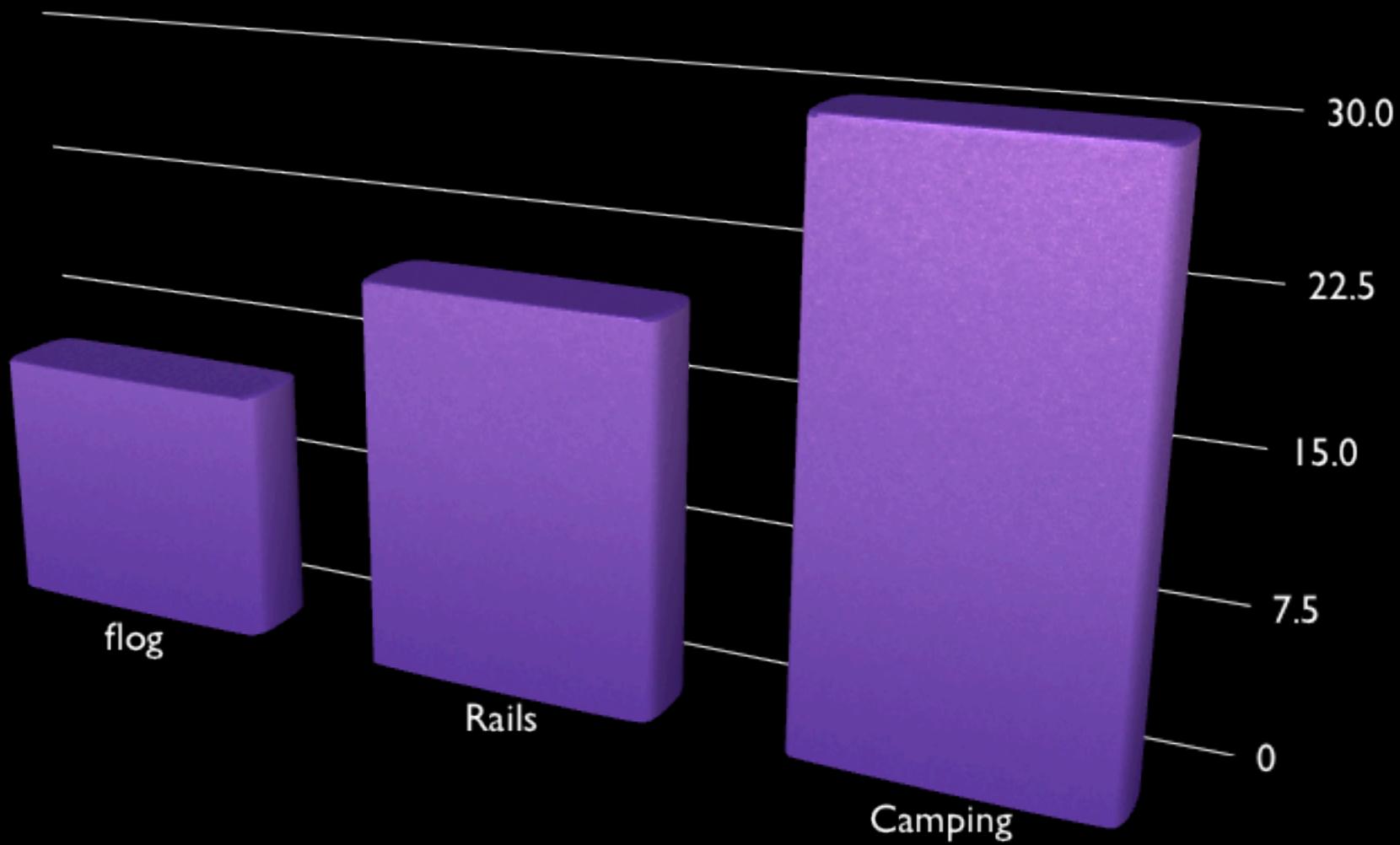
perspective



Ruby Sadists



sudo gem install flog



/THEORY/IN/PRACTICE

Beautiful Code

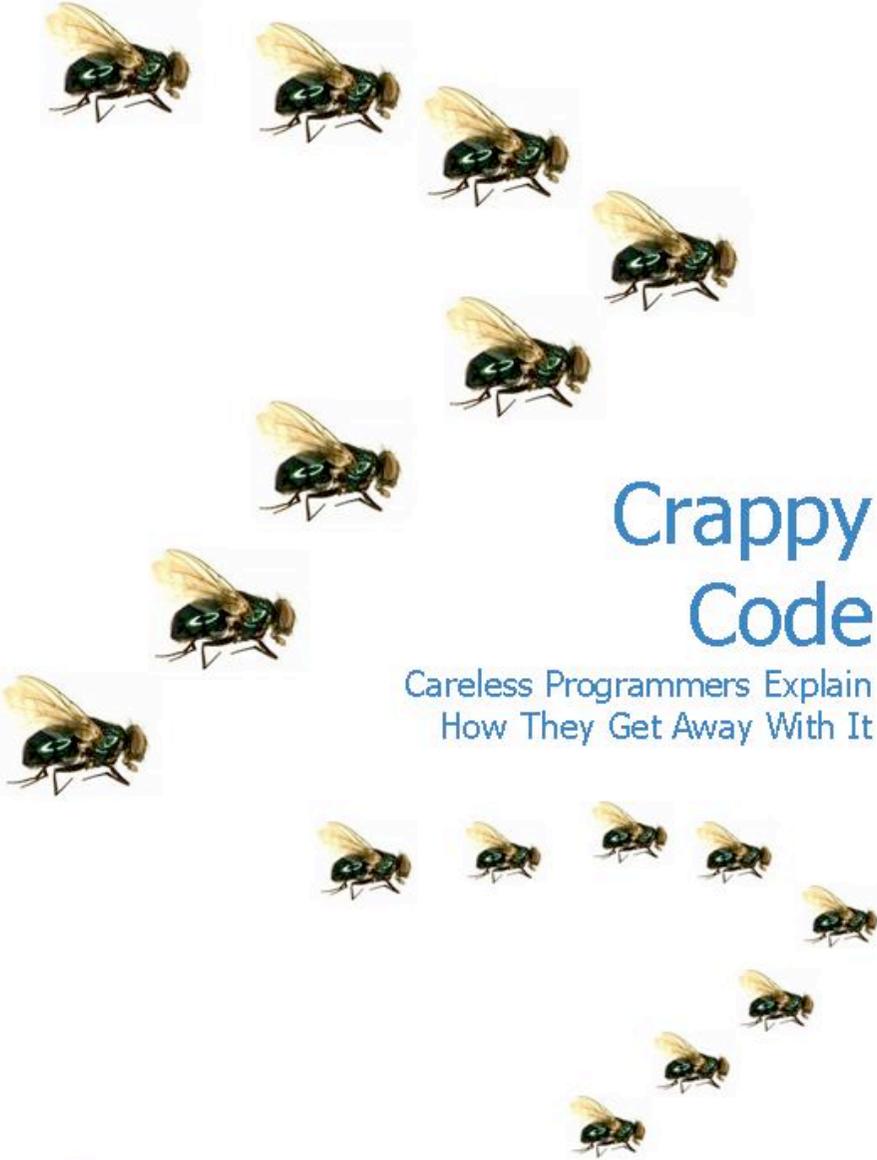
Leading Programmers Explain How They Think

O'REILLY®

Edited by Andy Oram & Greg Wilson

O'REILLY®

Edited by Andy Oram & Greg Wilson



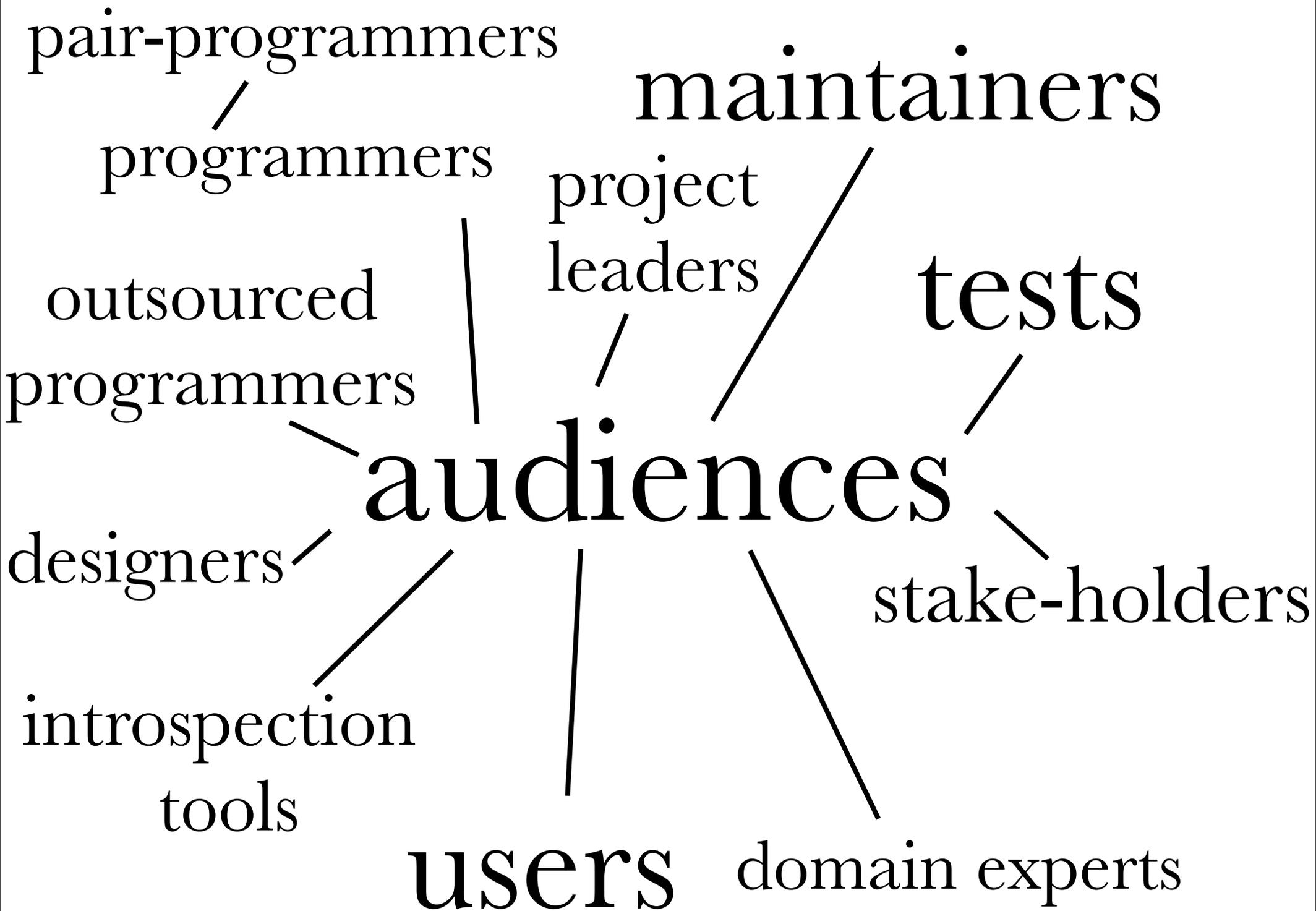
Crappy Code

Careless Programmers Explain How They Get Away With It

O'REALLY?™

O'REALLY?™

audiences



ORIGIN late 18th cent. (in the sense [relating to perception by the senses]): from Greek *aisthētikos*, from *aisthēta* '*perceptible things*,' from *aisthesthai* '*perceive*.' The sense [concerned with beauty] was coined in German in the mid 18th cent. and adopted into English in the early 19th cent., but its use was controversial until late in the century.

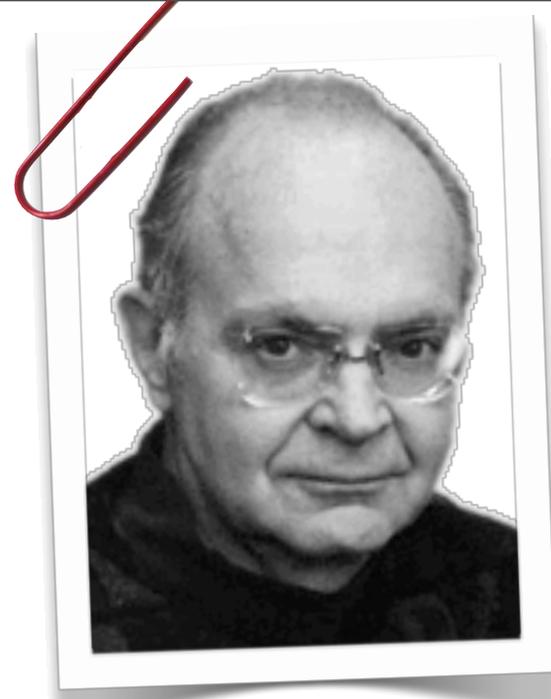
"Programmers spend so much of their time in their own heads that trying to look at the world from someone else's viewpoint is a big shift"

Kent Beck



“The main idea is to treat a program as a piece of literature, addressed to human beings rather than to a computer”.

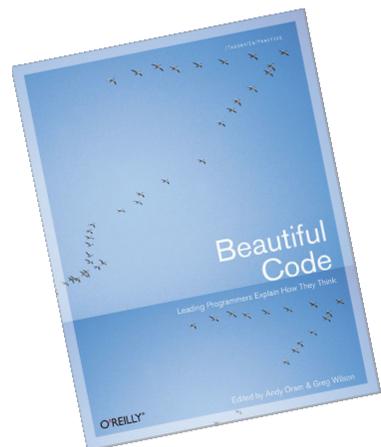
Donald Knuth

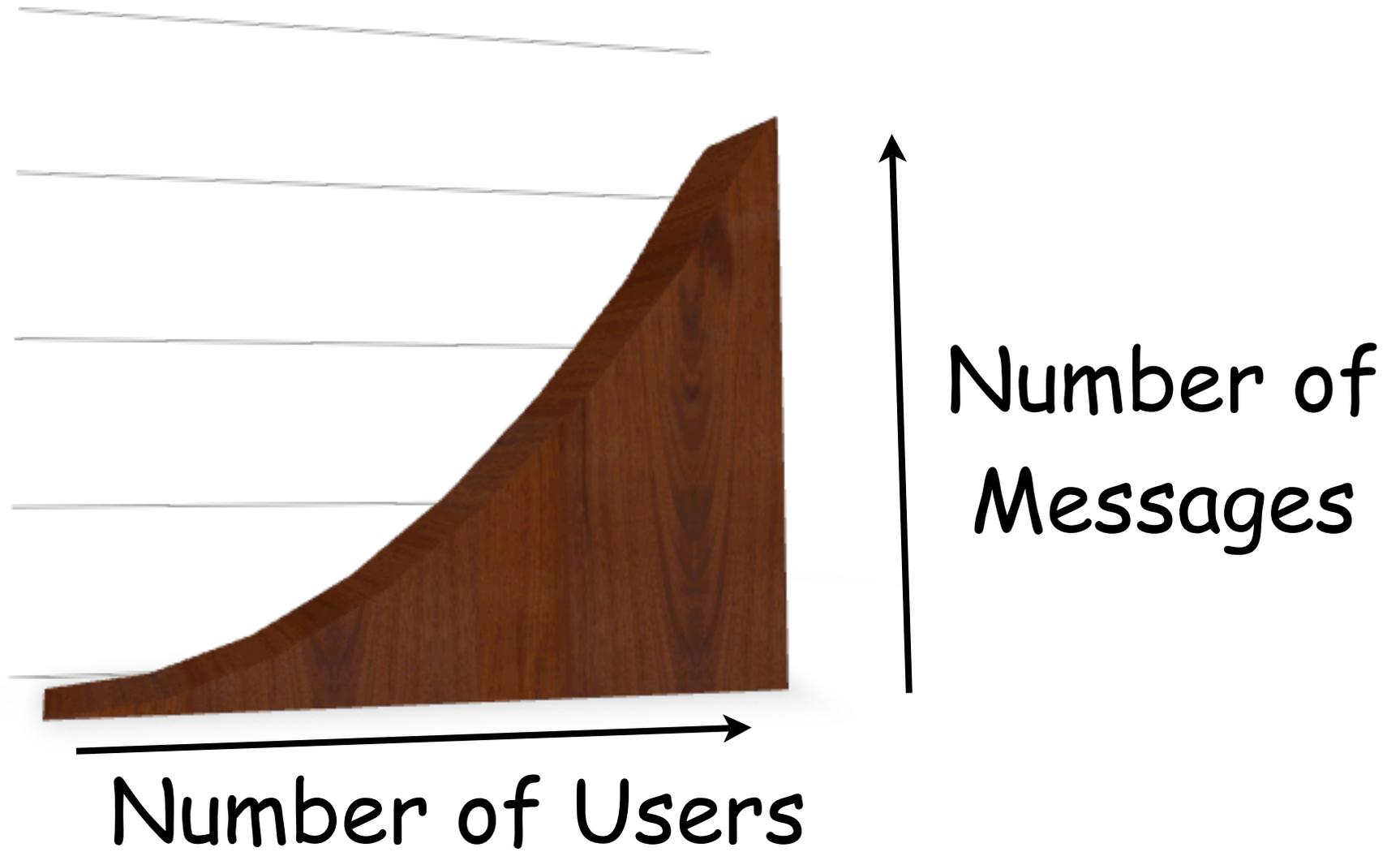




“Treating Code as an Essay”

Yukihiro Matsumoto, 松本行弘, Matz



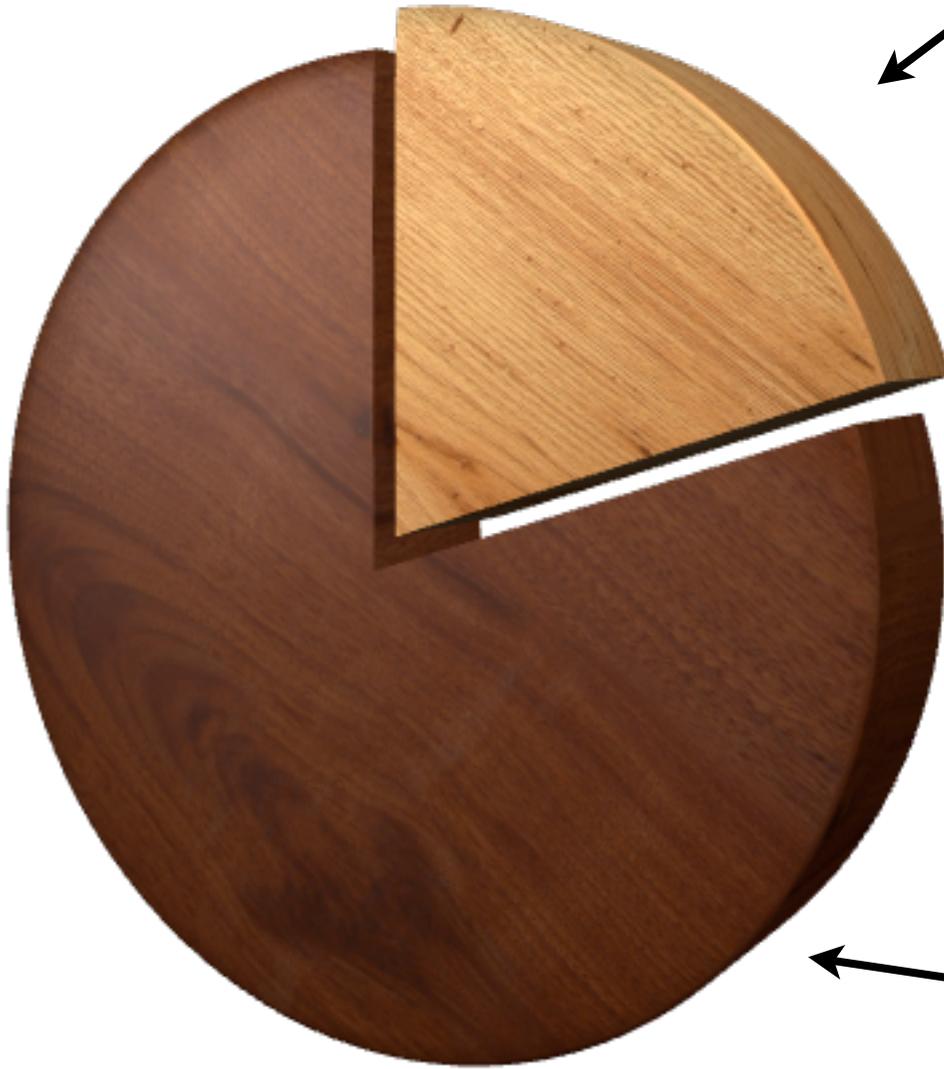




Everything



Interesting



Boring

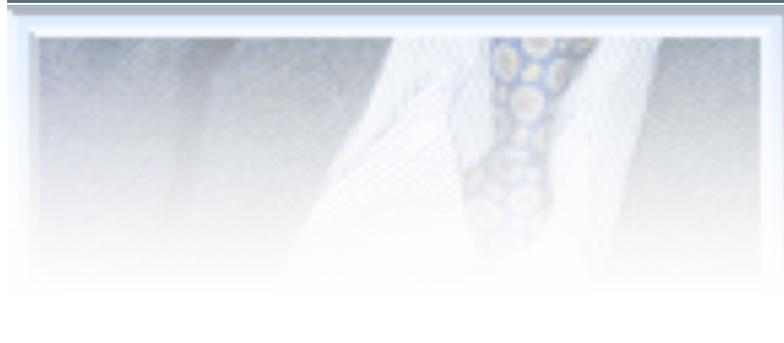
```
select * from artefacts
      where
      (colour = 'red')
```

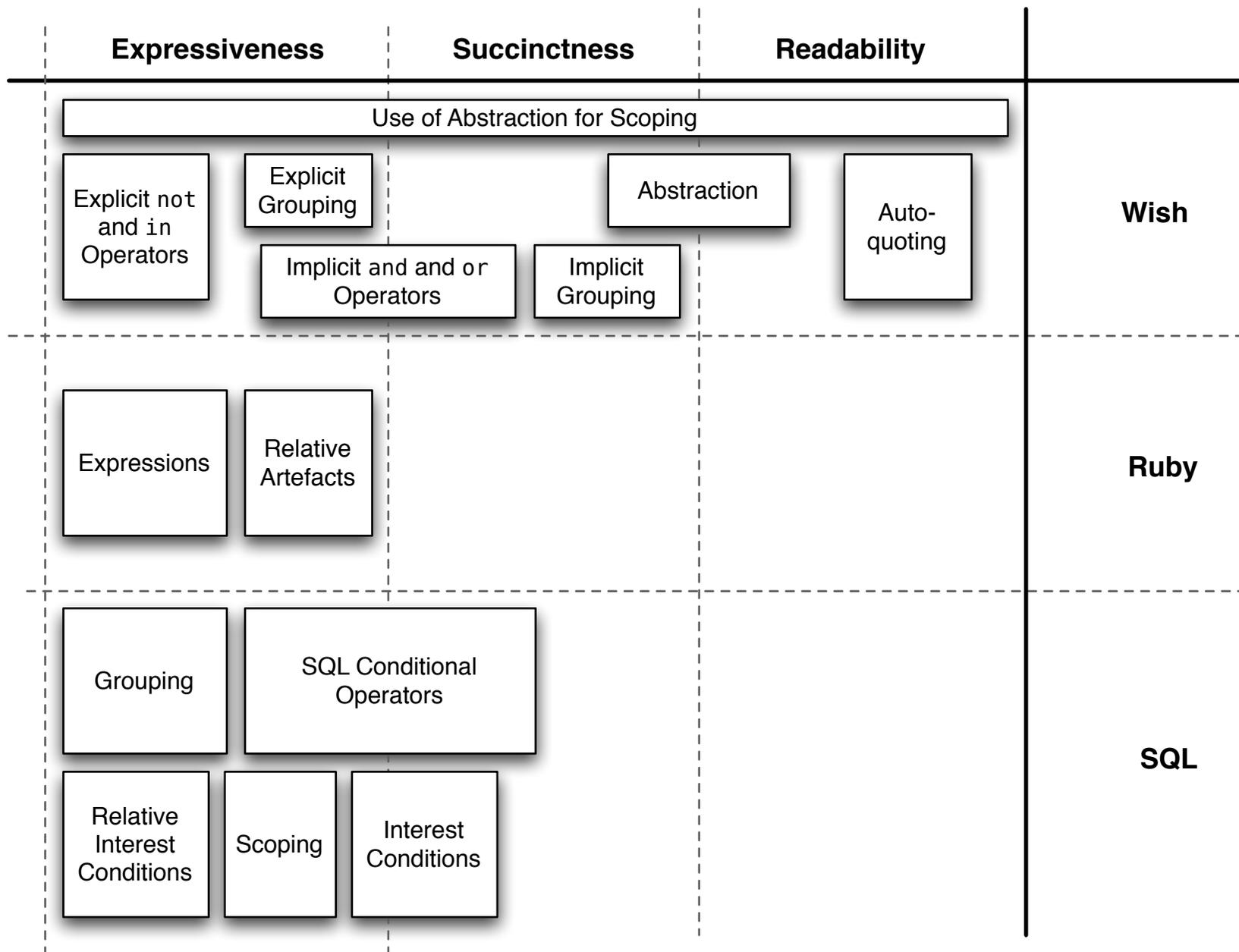
```
select * from artefacts where (virtual = false
and ((colour = 'red') and (category = 'player'
and
((virtual = false and (category = 'player' and
(name in (select name from artefacts where
(((5.0 + radius > sqrt(pow((x_coord - 27.0),
2) + pow((y_coord - 13.0), 2)))) and (category
= 'aura'))))))))
and ((9.5 > sqrt(pow((x_coord - 25.0), 2) +
pow((y_coord - 10.0), 2)))))))))
```

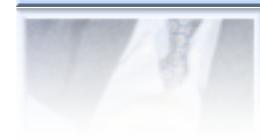
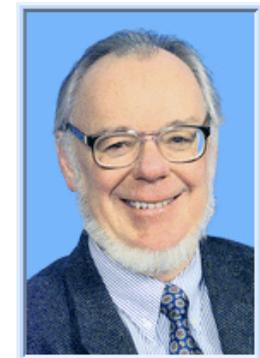
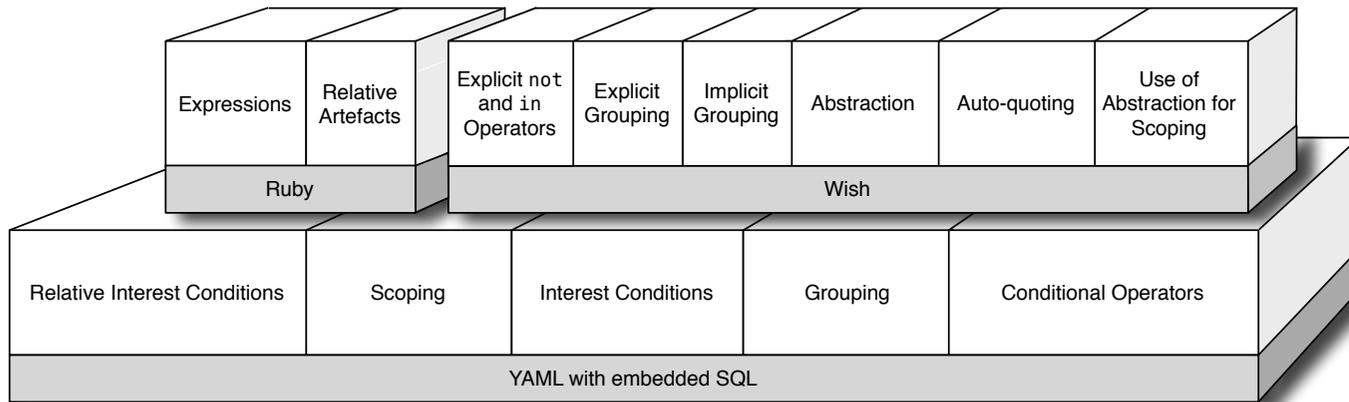


Complexity of
Interest

Complexity of
SQL







Domain Specific Languages

```
class CatalogueSubgroup < ActiveRecord::Base  
  has_many :catalogue_groupings  
  has_many :items, :through => :catalogue_groupings  
  belongs_to :catalogue_group  
end
```

```
class GroupingMerger < Merger
  for_model Grouping
  listen_to 'ItemPDBSubgroup'

  primary_key :item_number, :group_code, :subgroup_code

  map 'ItemNo' => :item_number
  map 'PDBGroupCode' => :group_code
  map 'PDBSubgroupCode' => :subgroup_code
  map 'Position' => :position, :type => :integer
  map 'Description' => :description
end
```

```
it "should put expression tags around a word containing multiple .s " do
  wish = "attribute = 10.next.next"
  post_parse = "(attribute = 12)"
  @interest.parse_interests(wish, "").should == post_parse
end
```

```
class LegacyAssociations < AssociationLinker

  link "user belongs to a contact" do
    from User => :contact_number
    to Contact => :number
  end

  link "favourite belongs to an item" do
    from Favourite => :item_number
    to Item => :number
  end

  link "item_metadata belongs to an item" do
    from ItemMetadata => :item_number
    to Item => :number
  end

end
```

"Not just 'what will the computer do with this code?' but 'How can I communicate what I am thinking to people?'"

Kent Beck

