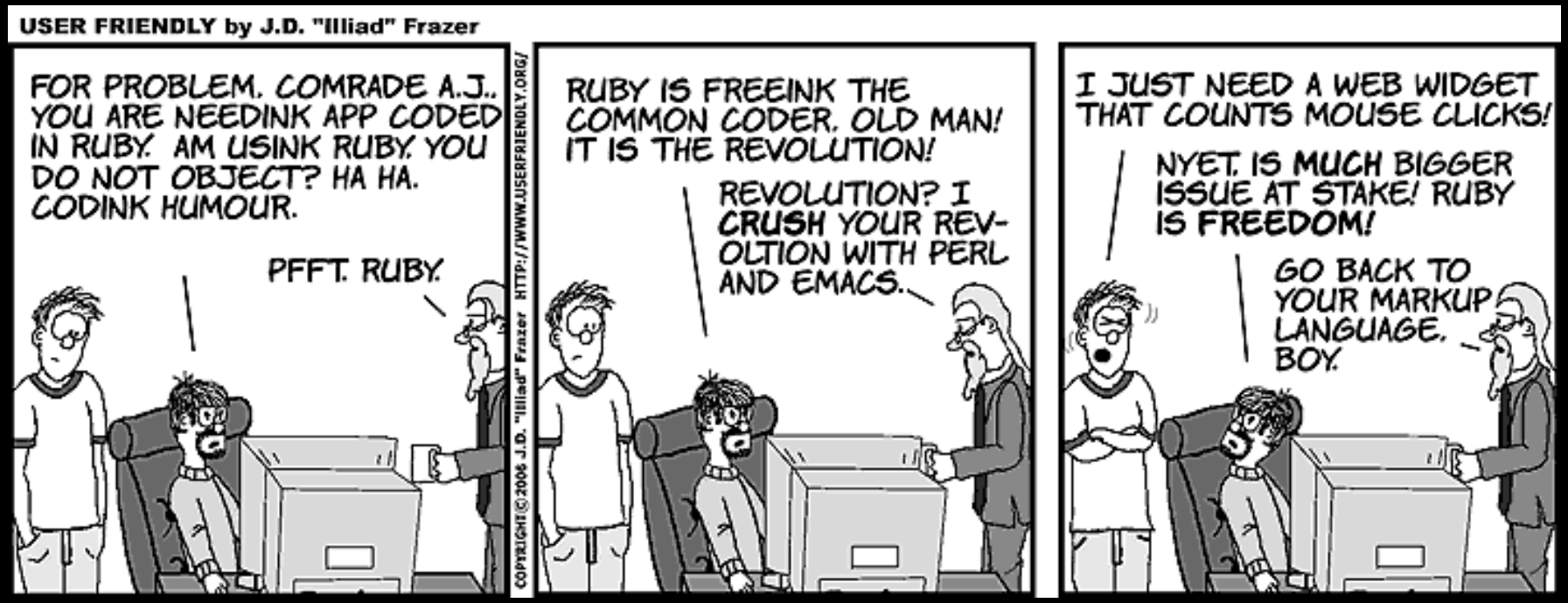


http://ars.userfriendly.org/cartoons/?id=20080627

CS 152: *Programming Language Paradigms*

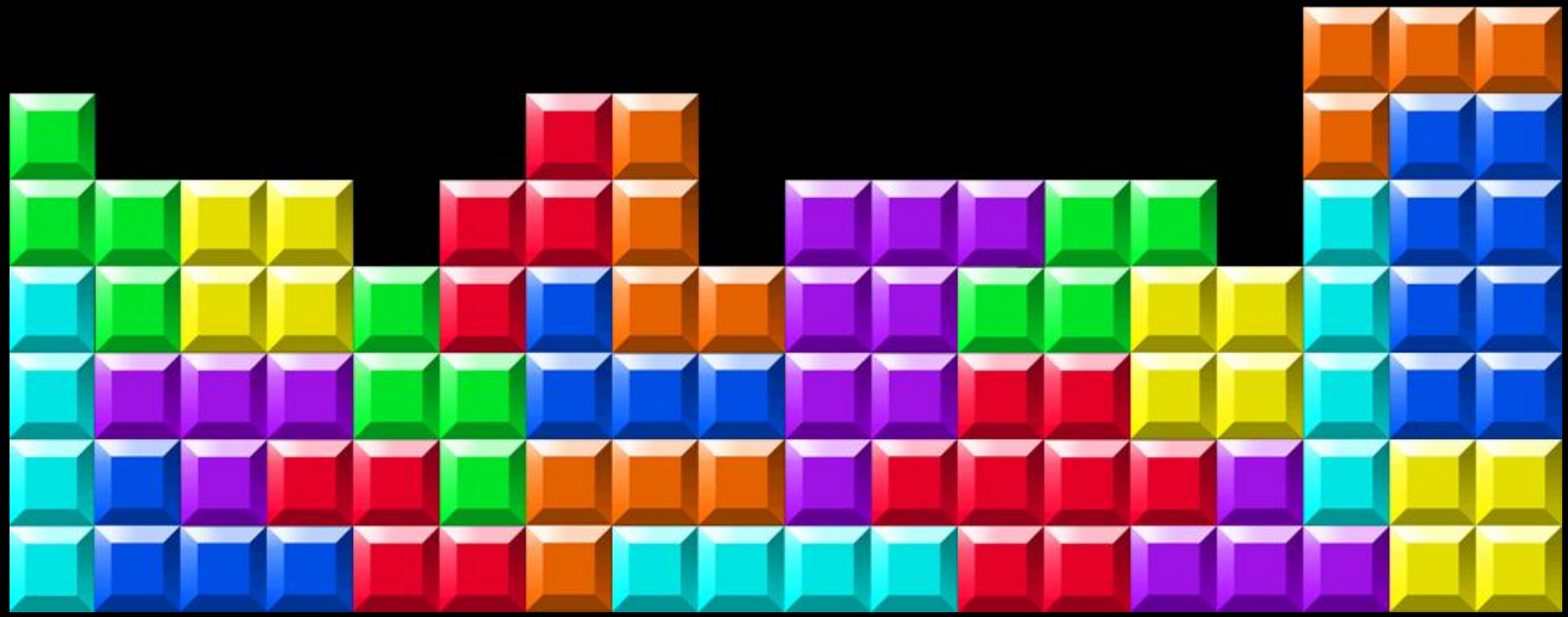
Blocks and Message Passing

Prof. Tom Austin
San José State University

Smalltalk influences on Ruby

- Everything is an object
- Blocks
- Message passing

Blocks



File I/O

```
file = File.open(
           'temp.txt', 'r')
file.each_line do |line|
  puts line
end
file.close
```

File I/O with blocks

```
File.open('file','r') do |f|
  f.each_line { |ln| puts ln }
end
```

Creating custom blocks

Ruby methods can accept blocks to

- create custom **control structures**
- eliminate **boilerplate** code

with_prob

- Accepts:
  - a probability (between 0 and 1)
  - a block
- Executes block probabilistically
  - `with_prob 0 { puts "hi" }`
  - `with_prob 1 { puts "hi" }`
  - `with_prob 0.5 { puts "hi" }`

```
def with_prob (prob)
  yield if (Random.rand < prob)
end

with_prob 0.42 do
  puts "There is a 42% chance "
    + "that this code will
print"
end
```

Single-line `if` statements come after the body

```
def with_prob (prob, &blk)
  blk.call if (Random.rand < prob)
end

with_prob 0.42 do
  puts "There is a 42% chance "
    + "that this code will print"
end
```

Note that there is no '&' here.

We can explicitly name the block of code

```
def with_prob (prob, &blk)
  blk.call if (Random.rand < prob)
end

def half_the_time (&block)
  with_prob(0.5, &block)
end
```

Explicitly naming the block is useful if we wish to pass it to another method

Conversion table example

(in class)

Blocks are *closures*, though there are some differences between JavaScript functions and Ruby blocks.

Let's see how the two compare…

Writing `withProb` in JavaScript

```
function withProb(prob, f) {
  if (Math.random() < prob) {
    return f();
  }
}
```

JavaScript uses callbacks rather than blocks

What is the difference?

```
def coin_flip
  with_prob 0.5
  do
    return "H"
  end
  return "T"
end
```

```
function coinFlip() {
  withProb(0.5,
    function() {
      return "H";
    });
  return "T";
}
```

Singleton classes

JavaScript & prototypes

```
function Employee(name, salary) {
  this.name = name;
  this.salary = salary;
}
var a = new Employee("Alice", 75000);
var b = new Employee("Bob", 50000);

b.signingBonus = 2000;
console.log(a.signingBonus);
console.log(b.signingBonus);
```

Can we do the same thing in Ruby?

In Ruby, every object has a special *singleton class*.

This class holds methods unique to that object.

Singleton Class Example

(in-class)


Message Passing

Message passing (object interaction)

- Sender sends
  - message: method name
  - data: method parameters
- Receiver
  - processes the message
  - (optionally) returns **data**

If receiver doesn't understand message?

```
irb> "hello".foo
NoMethodError: undefined
method `foo' for
"hello":String
    from (irb):2
    from /usr/bin/irb:12:in
`<main>'
irb>
```

`method_missing`

- You can override this behavior
  - Smalltalk: `doesNotUnderstand`
  - Ruby: `method_missing`
- This method is called when an unknown method is invoked

```
class Person
  attr_accessor :name
  def initialize(name)
    @name = name
  end
  def method_missing(m)
    puts "Didn't understand #{m}"
  end
end
```

```
bob = Person.new "Robert"
class << bob
   def method_missing m
    phrase = m.to_s.sub(
         /say_(.*)/, '\1')
     puts phrase
   end
end
```

Rails ActiveRecord example

```
Person.find_by(first_name: 'David')

Person.find_by_first_name "John"

Person.find_by_first_name_and_last_name \
  "John", "Doe"
```

Record example
(in class)

Lab: Ruby Metaprogramming

Today's lab explores blocks and `method_missing`.

Download tree.rb from the course website. The lab description is available in both Canvas and the course website.