```
var first = 3;
var second = 4;

var sum = function(a, b) {
  a + b;
}

console.log("Sum = ", sum(first, second));
```

This is ruby

@kkuchta

Ruby is the best javascript

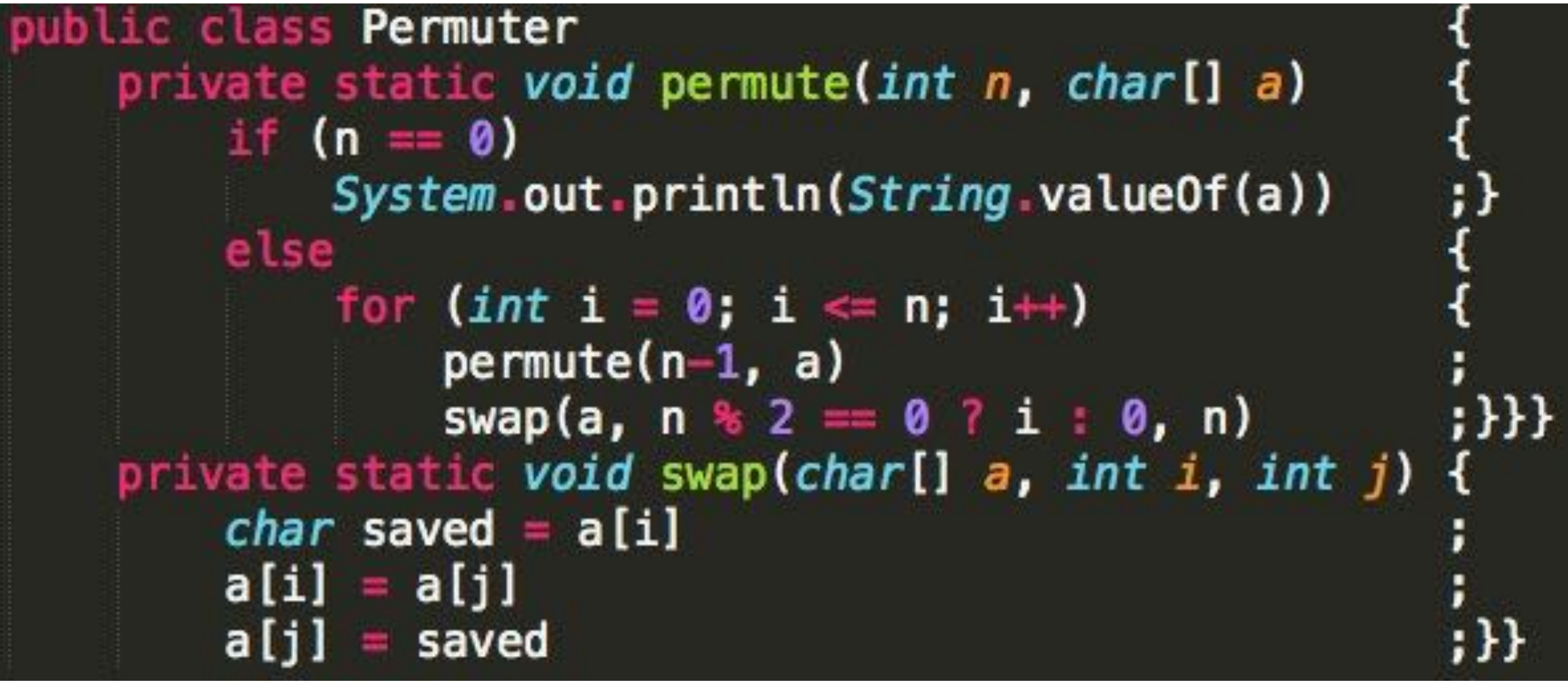
Kevin Kuchta | he/him | Joyable Inc

Good ideas

will not feature heavily in this talk

- Opal: ruby -> js
- Therubyracer: js in ruby
- Our thing: ruby == js

Inspiration



```
public class Permuter                                          {
    private static void permute(int n, char[] a)               {
        if (n == 0)                                            {
            System.out.println(String.valueOf(a))              ;}
        else                                                   {
            for (int i = 0; i <= n; i++)                       {
                permute(n-1, a)                                ;
                swap(a, n % 2 == 0 ? i : 0, n)                 ;}}}
    private static void swap(char[] a, int i, int j)           {
        char saved = a[i]                                      ;
        a[i] = a[j]                                            ;
        a[j] = saved                                           ;}}
```

@kkuchta

Taking it further

```
var first = 3;
var second = 4;

var sum = function(a, b) {
  a + b;
}

console.log("Sum = ", sum(first, second));
```

```
def var(_);end

var(first = 3)
var second = 4;
```

@kkuchta

```
class Console
  def log(input)
    puts input
  end
end
console = Console.new()
console.log("something")
```

Splat: *

```
def some_func(*my_args)
  puts my_args[0]
  puts my_args[1]
  puts my_args[2]
end
some_func('first arg',
'second arg', 'third arg')
```

@kkuchta

Splat: *

```ruby
def log(*args)
  puts args.join(',')
end

log('Hey', 'world')
```

Classes

```
class Foo
  ...
end
bar = Foo.new
```

Classes

```
bar.class # Foo
Foo.class # Class
Class.class # Class
```

@kkuchta

Anonymous Classes

```
Foo = Class.new do
  ...
end
bar = Foo.new
```

Anonymous Classes

```
bar = (Class.new {
  ...
}).new
```

@kkuchta

Console.foo

```
console =
  (Class.new {
    def log(*x); puts x.join(""); end
  }).new

console.log('hello', 'world')
```

@kkuchta

Anonymous Functions

```
// Javascript
var sum = function(a,b){
  return a + b
}
sum(3, 4)

# ruby
sum = Proc.new { |a, b| a + b }
sum.call(3, 4)
sum[3, 4]
```

@kkuchta

Functions

```
var sum = function(a, b) {
  return a + b;
}

def function(*args, &block)
  ... block stuff here ...
end
```

@kkuchta

Global Object

```
self # main
self.class # Object
```

@kkuchta

Global Object

```
class Object
  def foo
    'bar'
  end
end

foo # bar
3.foo # bar
```

@kkuchta

MethodMissing

```
class Foo
  def method_missing(*args)
    args[0]
  end
end
bar = Foo.new
bar.nonexistent
# returns :nonexistent
```

@kkuchta

GlobalMissing

```
class Object
  def method_missing(*args)
    return args[0]
  end
end
a # :a
b # :b
function(a, b) { return a + b }
```

@kkuchta

Caveat

```
def method_missing(*args)
  skip_methods = %i(
    to_a to_hash to_io to_str to_ary to_int
  )
  return nil if skip_methods.include?(args[0])
  return args[0]
end
```

Anonymous Functions

```
var sum =
  function(a, b) { |a, b|
    return a + b;
  }
```

@kkuchta

Send

```
some_obj = [ 'a', 'b', 'c']

some_obj.first # 'a'

some_obj.send('first') # 'a'
```

@kkuchta

Equals

```
obj = Class.new {
  attr_accessor(:foo)
}.new

obj.foo = 3
obj.foo=(3)
obj.send('foo=', 3)
```

@kkuchta

InstanceEval

```
obj = Class
  .new { attr_accessor :a, :b }
  .new
obj.a = 3
obj.b = 4
obj.instance_eval { a + b }
```

@kkuchta

Anonymous Functions

```
          [:a, :b]
             ↘
def function(*args, &block)
  klass = Class.new { attr_accessor *args }
  function_block = Proc.new { |*arg_values|
    obj = klass.new
    args.zip(arg_values).each { |arg, arg_value|
      obj.send(:"#{arg}=", arg_value)
    }
    obj.instance_eval(&block)
  }
  return function_block
end
```

@kkuchta

So Close

```
var sum = function(a, b) { a + b }
sum.call(3, 4) # 7
```

@kkuchta

Local Variables

```
a = 1
b = 2
local_variables
# [:b, :a]
```

@kkuchta

Define_method

```
class Foo
  [1,2,3,4].each do |i|
    define_method("get_#{i}") { i }
  end
end

Foo.new.get_1 # 1
Foo.new.get_2 # 2
```

@kkuchta

So Close-er

```
var func_1234 = function(a, b) { a + b }
define_method(:sum, &func_1234)
puts sum(3, 4) # 7
```

Tiny Text Success

```ruby
def function(*args, &block)
  func_name = :"func_#{rand(1000000)}"

  klass = Class.new { attr_accessor *args }
  function_block = Proc.new { |*arg_values|
    obj = klass.new
    args.zip(arg_values).each {|arg, arg_value| obj.send(:"#{arg}=", arg_value) }
    obj.instance_eval(&block)
  }

  define_method(func_name, &function_block)

  func_name
end

define_method(:var) { |random_function_name|
  var_name = local_variables.find do |local_var|
    local_var != :random_function_name && eval(local_var.to_s) == random_function_name
  end
  define_method(var_name) { |*args|
    send(random_function_name, *args)
  }
}
```

@kkuchta

Success!

```
var sum = function(a, b) {
  a + b
}
puts sum(3, 4) # 7
```

```
var first = 3;
var second = 4;

var sum = function(a, b) {
  a + b;
}

console.log("Sum = ", sum(first, second));
```

This is ruby

I'm @kkuchta