Ruby Tricks

Devpoint Labs - Dave Jungst / Jake Sorce

Splat

```
def my_method(a,*b)
  return a, b
end


my_method(1, 2, 3, 4 ) => [1, [2,3,4]]
my_method(1) => [1, []]
my_method => error
```

```
def my_method(a,*b, **c)
  return a, b, c
end


my_method(1, 2, 3, 4, { a: '1', b: '2' }) => [1, [2,3,4], { a: '1', b: '2' }]
my_method(1) => [1, [], {}]
```

Default params

```
def my_method(a, b = false)
  return a,b
end
```

my_method(1) => [1, false]

my_method(1, true) = > [1, true]

```
def my_method(a, def1 = true, def2 = 'blue')
  return a, def1, def2
end
```

my_method(1) => [1, true, 'blue']

my_method(1, false) => [1, false, 'blue']

my_method(1, 'red') => [1, 'red', 'blue']

my_method(1, true, 'blue') => [1, true, 'blue']

Anonymous methods

Lambda Literal

```
x = -> { 1 + 1 }

x.call => 2

x = -> (num) { num + 1 }

x.call(5) => 6
```

Proc

```
square = Proc.new { |n| n ** 2 }

square.call(2) => 4


def my_method(num, sq)
  puts “#{num} squared is #{sq.call(num)}”
end

my_method(2, square) => “2 squared is 4”
```

Collections

Array.new(3) => [nil,nil,nil]

Array.new(3, 1) => [1,1,1]

Array.new(2) { rand(6) + 1 } => [5, 1]

roll = -> { Array.new(2) { rand(6) + 1}}

roll.call => [2,4]

val1, val2 = [‘A’, ‘B’] => [‘A’, ‘B’]
val1 => ‘A’
val2 => ‘B’

values = [ ]
values[0..2] = [‘A’, ‘B’, ‘C’]