Readable Regexps With TRegexp

github.com/iangreenleaf

@iangreenleaf

blog.iangreenleaf.com

self.inspect

Ian Young

Used to do PHP

Rails is much better

work for AMCO Online (huzzah)

2 SDRuby meetings (including this one)

TRegexp



Because I can

Because I love regexps

...but...

 $/^{(+d)*}s*((d{3}))s*)*d{3}(-{0,1}|s{0,}$

What if we used words?

less compact

but friendlier

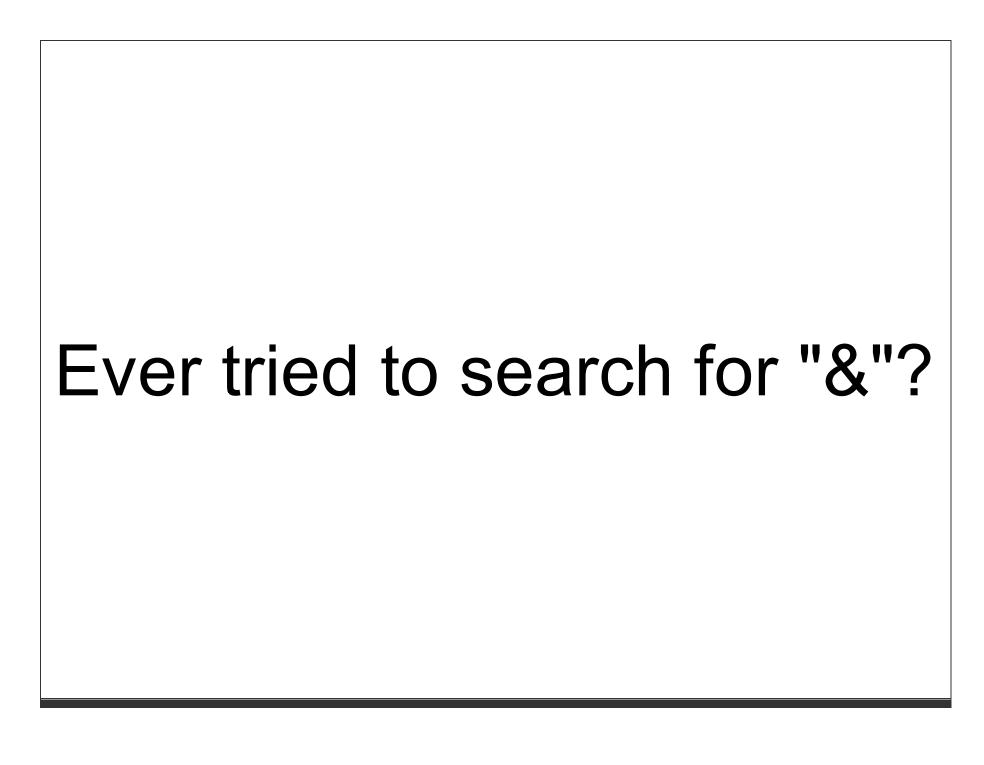
Examples!

"Your string" =~ /string/

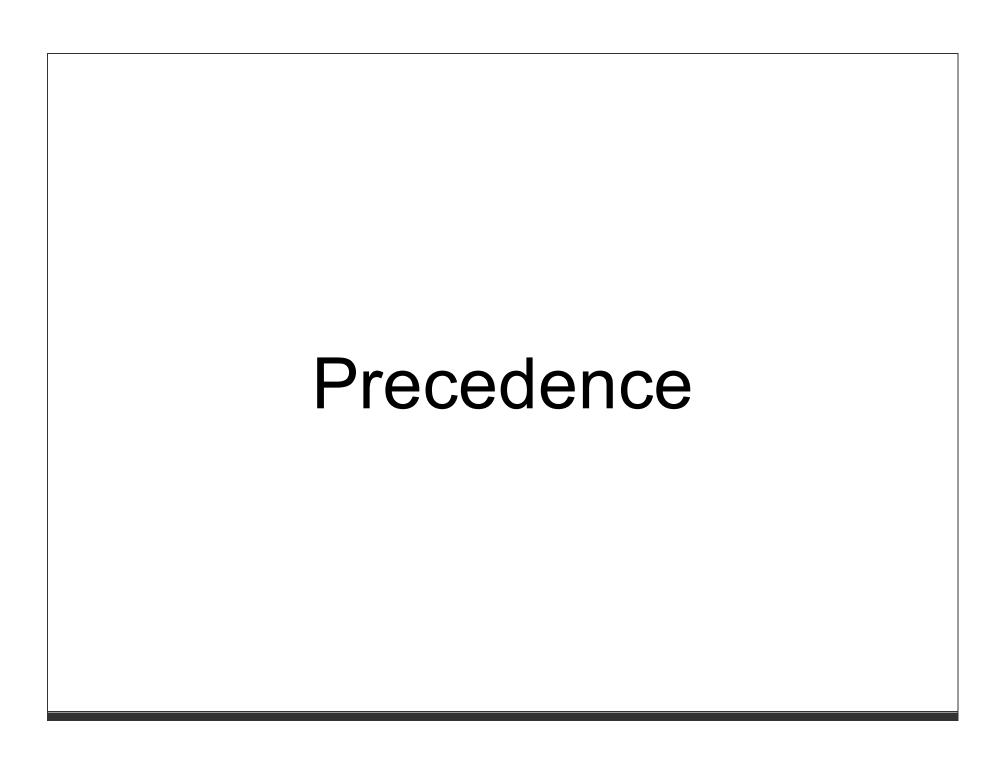
"Your string".rmatch? do "string" end

```
"abc".rmatch? { "ab" + "c" }
```

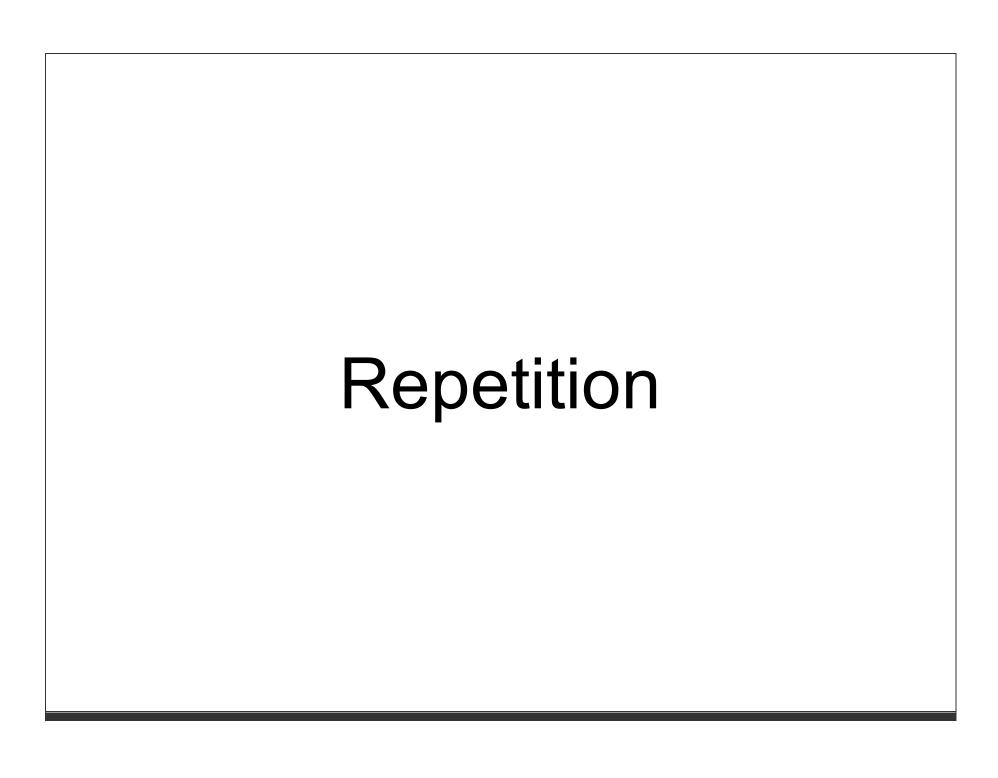
```
"abc".rmatch? { "xyz".or "abc" }
```



```
"symbols.*&+galore".rmatch? { ".*&+" }
```



```
"abc".rmatch? { "ab" + ( "z".or "c" ) }
```



```
"aaa".rmatch? { 1.or_more "a" }
"aaa".rmatch? { 5.or_less "a" }
"aaa".rmatch? { 3.exactly "a" }
"aaa".rmatch? { (1..5).of "a" }
```

```
"aaa".rmatch? { 0.or_more "a" }
"aaa".rmatch? { any "a" }
```

```
"aaa".rmatch? { 1.or_more "a" }
"aaa".rmatch? { some "a" }
```

Character sets

```
"abc1234.&*".rmatch? { 10.exactly any_char }
```

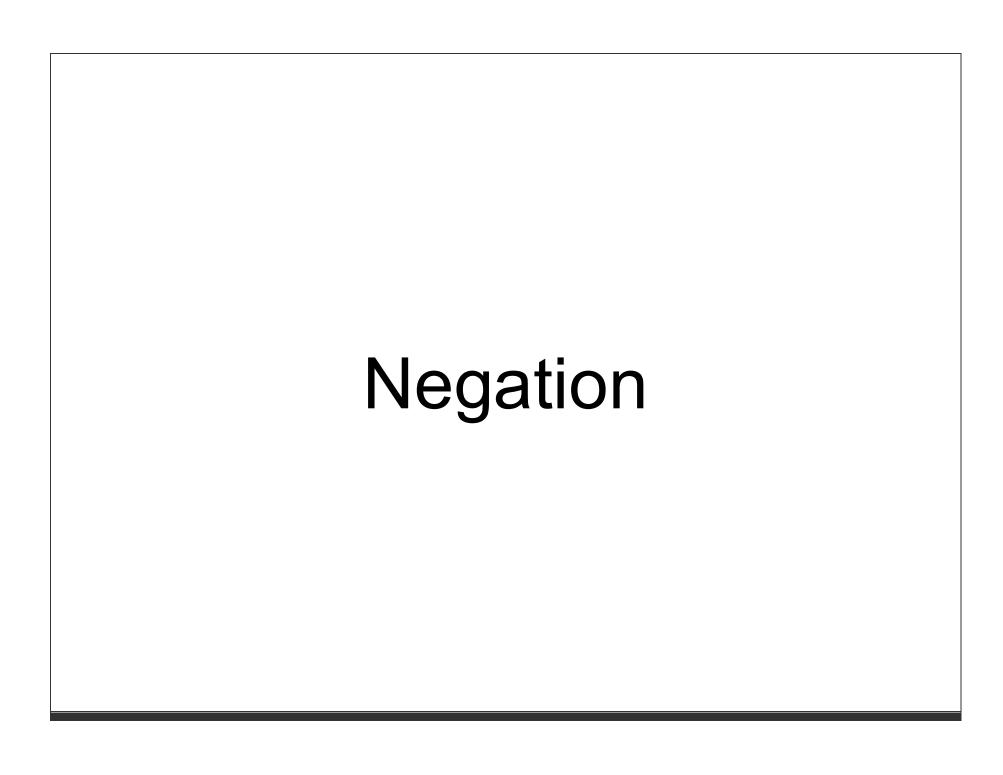
```
"abc".rmatch? { 3.exactly letter }
```

```
"1234".rmatch? { 4.exactly digit }
```

```
"abc_123".rmatch? { 7.exactly word_char }
```

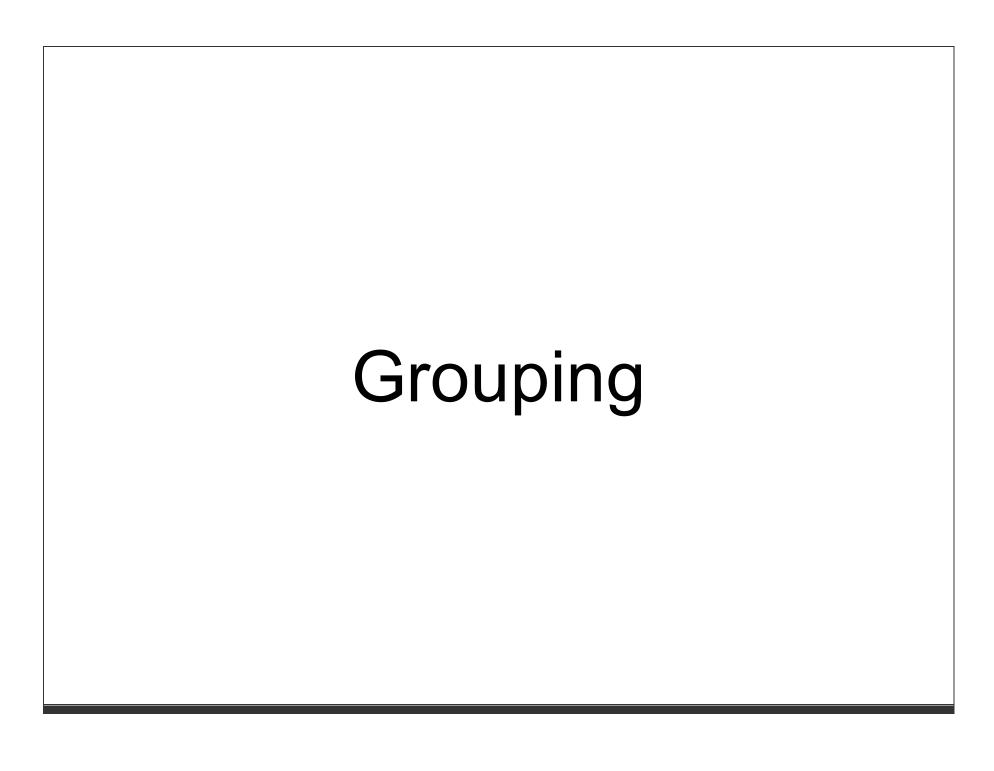
```
" ".rmatch? { whitespace }
```

```
"abc".rmatch? { 3.exactly "a".."c" }
```



```
"x".rmatch? { word_char.not "x" } # => nil
"y".rmatch? { word char.not "x" }
```

```
"x".rmatch? { _not "x" } # => nil
"y".rmatch? { _not "x" }
```



Works the old way

```
match = "abc".rmatch do
  group( "a" + group( "b" ) ) + "c"
end

match.to a #=> [ "abc", "ab", "b" ]
```

Named groups

```
match = "abcde".rmatch? do
  group(:word, any(word_char))
end

match[:word] #=> "abcde"
```

Or use blocks

```
match = "abcde".rmatch? do
  group :word do
  any word_char
  end
end

match[:word] #=> "abcde"
```

```
order = "1234567890 Central Processing"
match = order.rmatch? do
  group :serial do
    some digit
  end \
  + some whitespace + \
  group :source do
   any any char
  end
end
puts "Order #" + match[ :serial ]
puts "Shipped from " + match[ :source ]
```

Apply brakes

"experimental"

Monkey patches one or two teeny tiny

unimportant little core libs

fixnum.rb
range.rb
string.rb

Refinements will make this awesome



It turned out pretty cool

Need a better name

Could add more syntax

Taking suggestions

def zero_width_positive_lookbehind end

No.

Kiddie pool?

Other applications?

kthx

github.com/iangreenleaf

@iangreenleaf

blog.iangreenleaf.com