

Variable Scope Recap

First lets be more 'ruby like'

Java: `myVar = 1`

js: `myVar = 1`

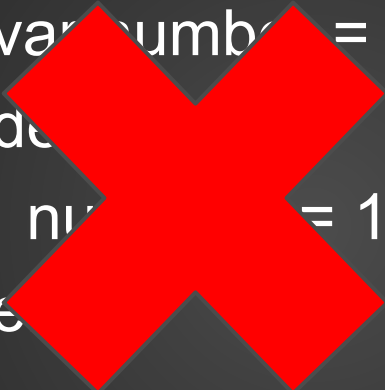
C#: `myVar = 1`

ruby: `my_var = 1`

variables and method names have the same
naming convention

Local Scope

```
var number = 0
if number > 0
  #some code
else
  #some code
end
```



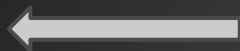
```
var number = 0
def add_one(num)
  num += 1
end
```

```
var number = 0
def add_one(num)
  num += 1
end

add_one(number)
```

Local Scope (cont)

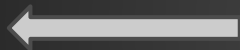
```
number = 2
```



These are not the same
variables in memory even
though they have the same
name

```
def add_one
```

```
  number = 0
```



```
  number += 1
```

```
end
```

```
number_2 = add_one
```

```
puts number      => 2
```

```
puts number_2    => 1
```

Local Variables on the fly

```
x = true
```

```
if x
```

```
    y = "It's true"
```

Does this code work?

```
else
```

```
    y = "It's a lie"
```

```
end
```

```
puts y
```

Local Variables on the fly (cont)

```
x = true
if x
  y = "It's true"
else
  y = "It's a lie"
end
puts y
```

```
def puts_y
  puts y
end
```



```
x = true
if x
  y = "It's true"
else
  y = "It's a lie"
end
puts y
```

```
def puts_y(my_variable)
  puts my_variable
end

puts puts_y(y)
```

```
x = true
if x
  @y = "It's true"
else
  @y = "It's a lie"
end
puts @y

def puts_y
  puts @y
end
```

SCOPE!!!

```
x = 2
```

```
puts x           => 2
```

```
def manipulate_x
```

```
  x = 0
```

```
  x += 1
```

```
end
```

```
puts manipulate_x  => 1
```

```
puts manipulate_x(x)  => 3
```

```
def manipulate_x(x)
```

```
  x += 1
```

```
end
```