Strings and Symbols

Strings

- Single-quote literal strings are **very** literal
  - Allow escaping of ‘ with `\`
  - Show (almost) everything else as is
- Double-quoted strings
  - Interpret special characters like `\n` and `\t`
  - Allow string interpolation!
    - Embedding arbitrary expressions using `#{}`
    - Don’t bother with concatenating with `+`

Strings (Continued)

```
single_quoted = 'one line \n followed by what\'s the matter?`
double_quoted = "one line \n followed by another line"
puts single_quoted # => one line \n followed by what's the matter?
puts double_quoted # => one line
                   # => followed by another line

def feedback (some_param)
  some_param = [1,2,3] if some_param.nil?
  "The parameter is #{some_param}"
end
puts feedback (5)  # => The parameter is 5
puts feedback nil # => The parameter is [1, 2, 3]
```

Strings (Continued)

- String methods ending with `!` modify the existing string
- Most others just return a new string
- Can also use `%Q{some real long multiline string}`- same behavior as double-quoted string
- `nil` object `to_s` produces an empty string!
- Very important to master String API !

Strings (Continued)

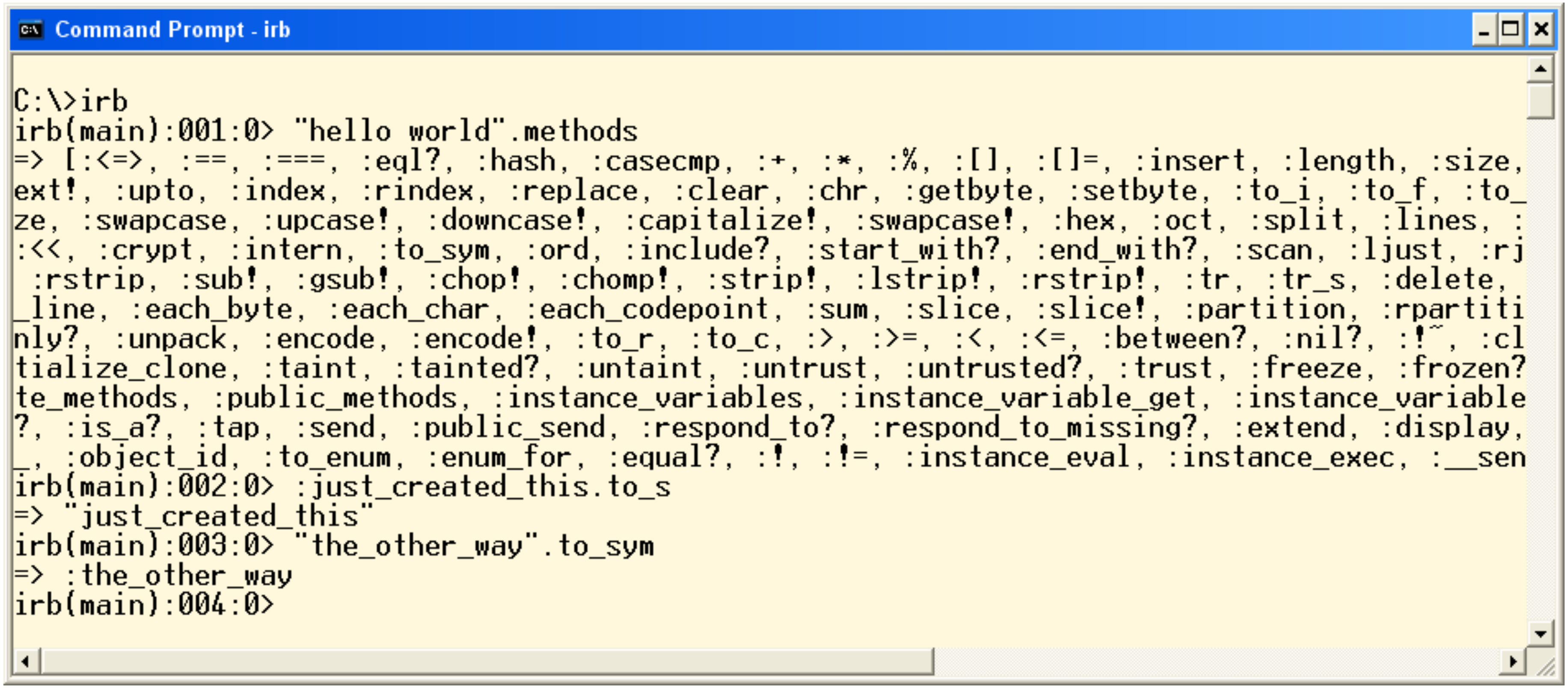
```
my_name = " tim"
puts my_name.lstrip.capitalize # => Tim
puts my_name # => tim
my_name.lstrip! # (destructive) remove the leading space
my_name[0] = 'K' # replace the fist character
puts my_name # => Kim

cur_weather = %Q{It's a hot day outside
                 Grab your umbrellas… }
cur_weather.lines do |line|
  line.sub! 'hot', 'rainy' # substitute 'hot' with 'rainy'
  puts "#{line.strip}"
end # => It's a rainy day outside
         Grab your umbrellas...
```

Symbols

- `:something` - highly optimized strings
- Constant names that you don't have to predeclare – "stands for something" string
- Guaranteed to be unique and immutable
- Can be converted to a `String` with `to_s`
- Often used as keys to `Hash`es (see later)
  - Makes retrieval very fast

Symbols (Continued)



```
C:\>irb
irb(main):001:0> "hello world".methods
=> [:<=>, :==, :===, :eql?, :hash, :casecmp, :+, :*, :%, :[], :[]=, :insert, :length, :size,
ext!, :upto, :index, :rindex, :replace, :clear, :chr, :getbyte, :setbyte, :to_i, :to_f, :to_
ze, :swapcase, :upcase!, :downcase!, :capitalize!, :swapcase!, :hex, :oct, :split, :lines, :
:<<, :crypt, :intern, :to_sym, :ord, :include?, :start_with?, :end_with?, :scan, :ljust, :rj
 :rstrip, :sub!, :gsub!, :chop!, :chomp!, :strip!, :lstrip!, :rstrip!, :tr, :tr_s, :delete,
_line, :each_byte, :each_char, :each_codepoint, :sum, :slice, :slice!, :partition, :rpartiti
nly?, :unpack, :encode, :encode!, :to_r, :to_c, :>, :>=, :<, :<=, :between?, :nil?, :!~, :cl
tialize_clone, :taint, :tainted?, :untaint, :untrust, :untrusted?, :trust, :freeze, :frozen?
te_methods, :public_methods, :instance_variables, :instance_variable_get, :instance_variable
?, :is_a?, :tap, :send, :public_send, :respond_to?, :respond_to_missing?, :extend, :display,
_, :object_id, :to_enum, :enum_for, :equal?, :!, :!=, :instance_eval, :instance_exec, :__sen
irb(main):002:0> :just_created_this.to_s
=> "just_created_this"
irb(main):003:0> "the_other_way".to_sym
=> :the_other_way
irb(main):004:0>
```