Ruby MasterClass

Gearing Up

by Francisco Bach

- **Regular Expression - Literal Characters**

"The quick brown fox jumps over the lazy dog."

/x/ *

"The quick brown fox jumps over the lazy dog."

/lazy/ *

* Within a pattern, all characters match themselves except . | ( ) [ ] { } + \ ^ $ * and ? you need to escape any of these characters with a backslash if you want them to be treated as regular characters to match

- **Regular Expression - Meta Characters**

"The quick brown fox jumps over the lazy dog."

"In England we write 'colour', in America we write 'color' !"

- Single Characters

  \d => 0 - 9 (\D)*
  \w => A-Za-z0-9 (\W)**
  \s => whitespace (\S)***
  . => any character except a new line

- Quantifiers

  * => 0 or more
  + => 1 or more
  ? => 0 or 1 (optional)
  {min, max} => range
  {n} => Single number n

- Position

  ^ => beginning of a line
  $ => end of a line
  \A => beginning of the string
  \z => end of the string
  \b => word boundary

- Examples:

  /\w/ outputs => "T", "h", "e", "q", "u", "i", "c", "k", "b", "r", "o", "w", "n", "f", "o", "x", "j", "u", "m", "p", "s", "o", "v", "e", "r", "t", "h", "e", "l", "a", "z", "y", "d", "o", "g"

  /\w+/ outputs => "The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"

  /./ outputs => "T", "h", "e", " ", "q", "u", "i", "c", "k", " ", "b", "r", "o", "w", "n", " ", "f", "o", "x", " ", "j", "u", "m", "p", "s", " ", "o", "v", "e", "r", " ", "t", "h", "e", " ", "l", "a", "z", "y", " ", "d", "o", "g"

  /\b\w{3}\b/ outputs => "The", "fox", "the", "dog"

  /\w+$/ outputs => "dog"

  /\b\w{3,4}\b/ outputs => "The", "fox", "over", "the", "lazy", "dog"

  /colou?r/ outputs => "colour", "color"

* Capital version of "\d" means, anything that it's not a digit from 0 to 9.
** Capital version of "\w" means, anything that it's not a the characters defined by the lowercase version.
*** Capital version of "\s" means, anything that it's not a whitespace.

- **Regular Expression - Meta Characters**

Single Characters

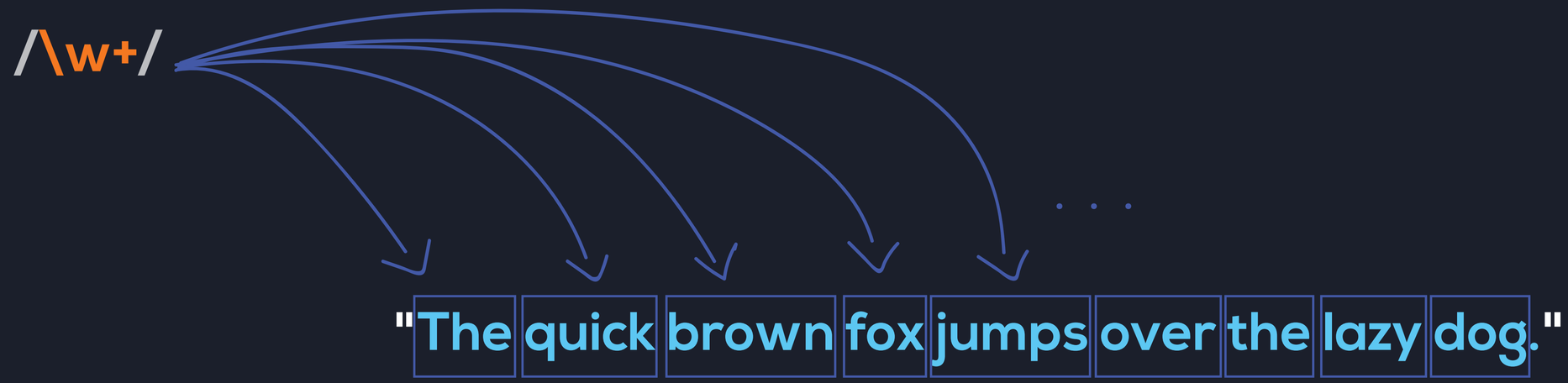
"The quick brown fox jumps over the lazy dog."

Matches every character defined by the meta character.
Excludes whitespace.

outputs => "T", "h", "e", "q", "u", "i", "c", "k", "b", "r", "o", "w", "n", "f", "o", "x", "j", "u", "m", "p", "s", "o", "v", "e", "r", "t", "h", "e", "l", "a", "z", "y", "d", "o", "g"

- **Regular Expression - Meta Characters**

Single Characters + Quantifier

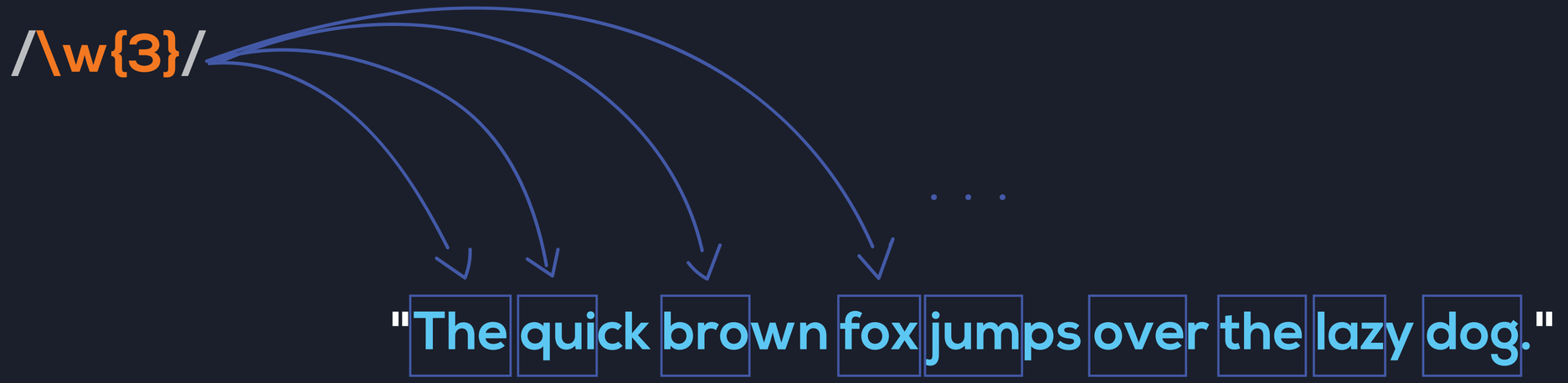


The quantifier “+” matches one or more character in a row.
It modifies the previous meta character behaviour.
Does not match whitespaces.

outputs => "The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"

- **Regular Expression - Meta Characters**

Single Characters + Quantifier



The quantifier “{n}“ matches “n” characters in a row.
It modifies the previous meta character behaviour.

outputs => "The", "qui", "bro", "fox", "jum", "ove", "the", "laz", "dog"

- **Regular Expression - Meta Characters**

Single Characters + Quantifier + Position

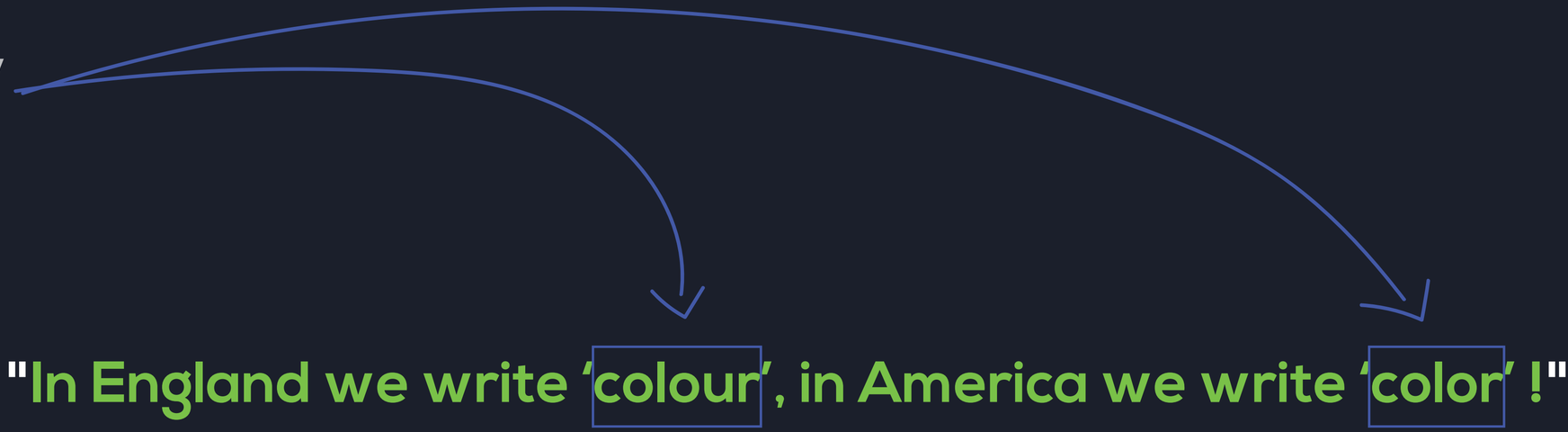
The meta character “\b” (boundary) sets a word boundary.
In this example matches only three letters word.

outputs => "The", "fox", "the", "dog"

- **Regular Expression - Meta Characters**

Single Characters + Quantifier

/colou?r/



"In England we write 'colour', in America we write 'color' !"

The meta character "?" matches 0 or 1 of the preceding character.
In this example makes the literal character "u" optional in the query.

outputs => "colour", "color"

- **Regular Expression - Character Classes**

  "This is a sample Text. And this is a sample Phone number 999-689-5550, and another phone number (000)899.321. My email is test.me@ruby.com"

- [characters]

  A character class is a set of characters between square brackets.

  [abc], matches “a” or “b” or “c”.

  [-.], matches a “dash” or a “dot”.

  The dash character (-) if its not in the very beginning of a character class, turns to be a special character. Ex.: [A-Z] matches all uppercase letters from A to Z. The same idea with [a-c], [0-9].

  The caret(^) character, negate the pattern defined. It will match anything that were not defined.

- Examples:

  /\(?\d{3}[-.)]\d{3}[-.]\d{3,4}/ outputs => "999-689-5550", "(000)899.321"

  /[A-Z]/ outputs => "T", "T", "A", "P", "M"

  /[abc]/ outputs => "a", "a", "a", "a", "b", "a", "a", "b", "a", "b", "c"

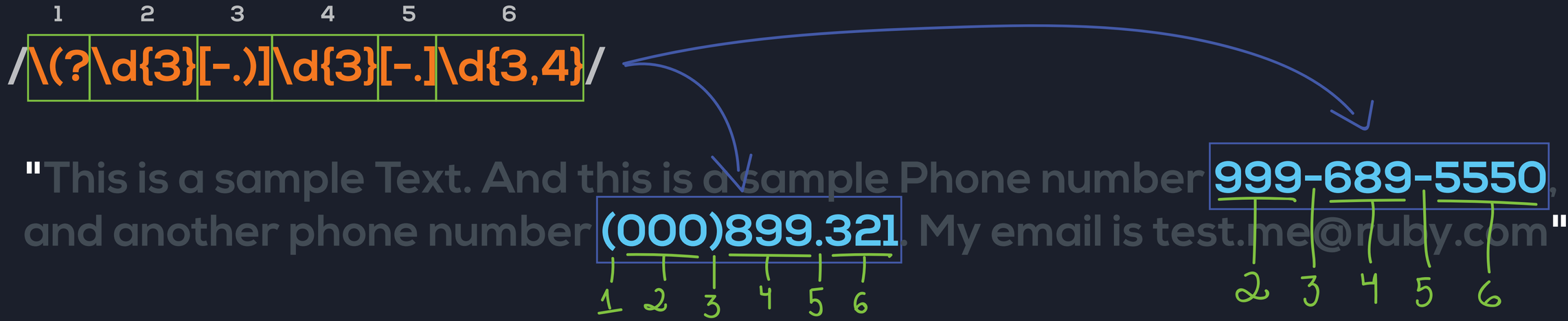
  /\b[A-Za-z]{4}\b/ outputs => "This", "Text", "this", "test", "ruby"

  /\b[A-Z][a-z]*\b/ outputs => "This", "Text", "And", "Phone", "My"

  /[\w.]+@\w+\.(net|com)/ outputs => "test.me@ruby.com"

- **Regular Expression - Character Classes**

Single Characters + Quantifier + Character Class

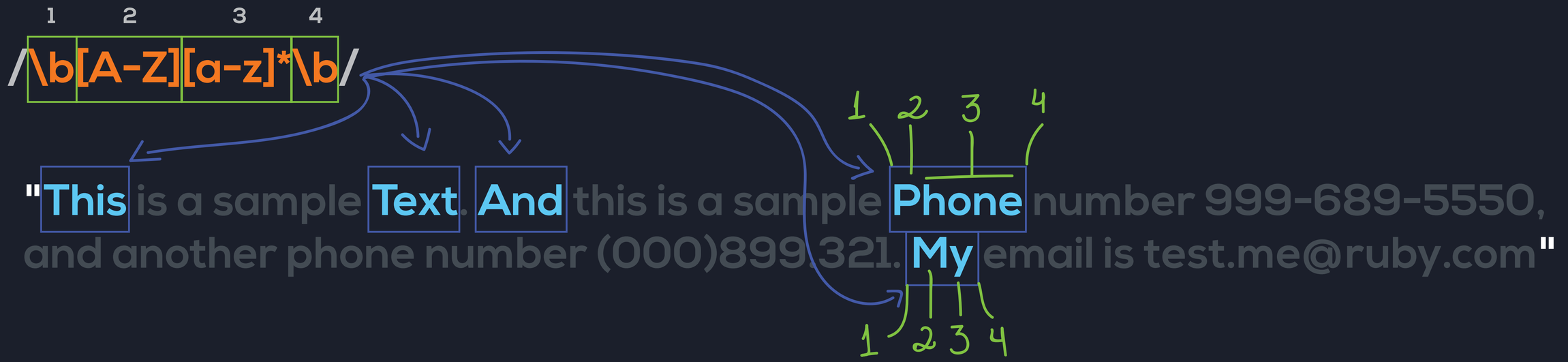


1. The left hand side parenthesis was escaped. \(, right after we make it optional using the question mark "?" meta character.
2. Followed by three digits in a row.
3. And then we created a character class, with the intent to match a "-" or a "." or a ")".
4. And then followed again by three digits in a row.
5. Followed by a "-" or a ".".
6. And then followed by a range of three to four digits in a row.

outputs => "999-689-5550", "(000)899.321"

- **Regular Expression - Character Classes**

Single Characters + Quantifier + Position + Character Class



1. The meta character “\b” sets a boundary.
2. Right after we set a character class to begin the boundary with any capital letter from A to Z.
3. Followed by 0 or more lowercase letters from a to z.
4. And then we close the boundary using the meta character “\b”.

outputs => "This", "Text", "And", "Phone", "My"