

CS4550 Web Development

Regular Expressions (RE)

Thang Nguyen

Outline

1. History of RE <
2. RE
3. Applications


History

- Credited to mathematician Stephen Kleene (Σ^*) in the 1950s

History

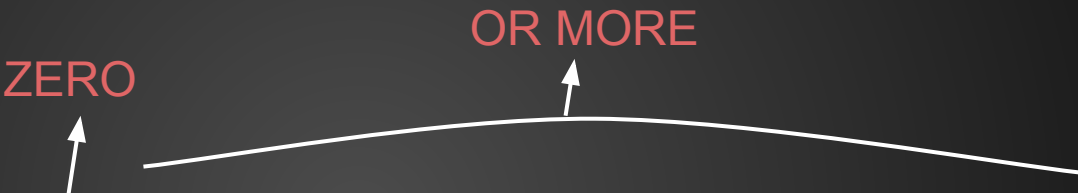
- Credited to mathematician Stephen Kleene (Σ^*) in the 1950s
- Example:
 $\{"ab", "c"\}^* = \{\epsilon, "ab", "c", "abab", "abc", "cc", \dots\}$

History

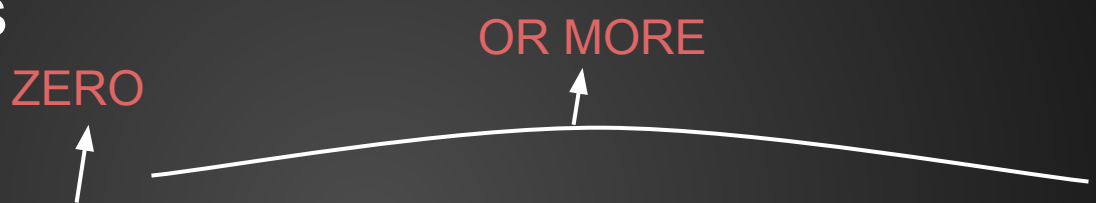
- Credited to mathematician Stephen Kleene (Σ^*) in the 1950s
- Example:  $\{\text{"ab"}, \text{"c"}\}^* = \{\epsilon, \text{"ab"}, \text{"c"}, \text{"abab"}, \text{"abc"}, \text{"cc"}, \dots\}$

History

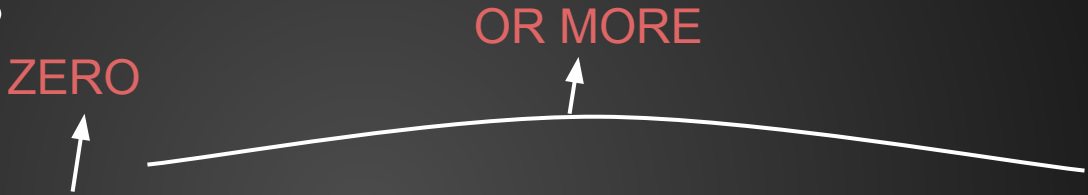
- Credited to mathematician Stephen Kleene (Σ^*) in the 1950s

- Example:
 $\{\text{"ab"}, \text{"c"}\}^* = \{\epsilon, \text{"ab"}, \text{"c"}, \text{"abab"}, \text{"abc"}, \text{"cc"}, \dots\}$
- 

History

- Credited to mathematician Stephen Kleene (Σ^*) in the 1950s
- Example:  $\{\text{"ab"}, \text{"c"}\}^* = \{\epsilon, \text{"ab"}, \text{"c"}, \text{"abab"}, \text{"abc"}, \text{"cc"}, \dots\}$
- Ken Thompson built support for RE into QED as a means to match patterns in text files

History

- Credited to mathematician Stephen Kleene (Σ^*) in the 1950s
- Example: 
 $\{"ab", "c"\}^* = \{\epsilon, "ab", "c", "abab", "abc", "cc", \dots\}$
- Ken Thompson built support for RE into QED as a means to match patterns in text files
- Added this capability to Unix which led to the popular search tool `grep`

Outline

1. History of RE
2. RE <
3. Applications

RE

- RE are basically a specific pattern that "matches" strings of text or patterns of characters

RE

- RE are basically a specific pattern that "matches" strings of text or patterns of characters
- The simplest type of RE is a literal match

RE

- RE are basically a specific pattern that "matches" strings of text or patterns of characters
- The simplest type of RE is a literal match

| RE | Matches | Doesn't Match |
|----|---------|---------------|
| at | at | it, a-t, At |

RE

- RE are basically a specific pattern that "matches" strings of text or patterns of characters
- The simplest type of RE is a literal match

| RE | Matches | Doesn't Match |
|----|---------|---------------|
| at | at | it, a-t, At |

- RE are case sensitive unless a modifier or options (-i) are specified

RE

- Wildcard, ".", can match any single character

RE

- Wildcard, ".", can match any single character

| RE | Matches | Doesn't Match |
|-------|--------------|--------------------------|
| t...s | trees, teens | trucks, trains, beans |

RE

- Wildcard, ".", can match any single character

| RE | Matches | Doesn't Match |
|-------|--------------|--------------------------|
| t...s | trees, teens | trucks, trains, beans |

- Escaping Metacharacters (i.e. \,], ?, ^)

RE

- Wildcard, ".", can match any single character

| RE | Matches | Doesn't Match |
|-------|--------------|--------------------------|
| t...s | trees, teens | trucks, trains, beans |

- Escaping Metacharacters (i.e. \,], ?, ^)

| RE | Matches | Doesn't Match |
|------------------|----------------|------------------------------------|
| c:\\readme\\.txt | c:\\readme.txt | c:\\readme.txt, c: \\readme_txt |

RE

- Parentheses "()" are often used to group characters and expressions

| RE | Matches | Doesn't Match |
|-------------|-------------------|---------------|
| (foo){2, 3} | foofoo, foofoofoo | foo, foooo |

RE

- Parentheses "()" are often used to group characters and expressions

| RE | Matches | Doesn't Match |
|-------------|-------------------|---------------|
| (foo){2, 3} | foofoo, foofoofoo | foo, foooo |

- Character classes are a mini-language within RE, defined by enclosing hard braces []

| RE | Matches | Doesn't Match |
|--------------|----------------------------|-------------------|
| ^b[aeiou]t\$ | bat, bet, bit, bot, but | beau, beat, beaut |

RE

- Quantifiers specify how many instances of the preceding element must appear in order to match

RE

- Quantifiers specify how many instances of the preceding element must appear in order to match
- As mentioned before Kleene Star is widely used in RE and is denoted by `*'

RE

- Quantifiers specify how many instances of the preceding element must appear in order to match
- As mentioned before Kleene Star is widely used in RE and is denoted by `*'

| RE | Matches | Doesn't Match |
|---------------------------|---|--|
| <code>www\my.*\com</code> | <code>www.my.com</code> , <code>www.mypage.com</code> , <code>www.mysite.com</code> , <code>www.mysite.com</code> some text goes here <code>cs4550.com</code> | <code>www.oursite.com</code> , <code>mypage.com</code> |

RE

QUANTIFIER SUMMARY

| Quantifier | Matches... | RE | Matches |
|------------|---------------------------------------|-----------------|-----------------|
| ? | any preceding element 0 or 1 times | colou?r | color, colour |
| * | the preceding element 0 or more times | www\.my.*\.com | www.mysite.com |
| + | the preceding element 1 or more times | bob5+@foo\.com | bob5555@foo.com |
| {n} | the preceding element n times | w{3}\.site\.com | www.site.com |

Outline

1. History of RE
2. RE
3. Applications <

Applications

- Data validation

Applications

- Data validation
 - US ZIP Code: `\d{5}(-\d{4})?`

Applications

- Data validation

- US ZIP Code: `\d{5}(-\d{4})?`

- Date (mm\dd\yyyy):

`^(0[1-9]|1[012])[-/.] (0[1-9]|12)[0-9]|3[01])[-/.] (19|20)\d\d$`

Applications

- Data validation

- US ZIP Code: `\d{5}(-\d{4})?`

- Date (mm\dd\yyyy):

`^(0[1-9]|1[012])[-./](0[1-9]|12)[0-9]|3[01])[-./](19|20)\d\d$`

- Email:

`^\w+([-+.] \w+)*@ \w+([-.] \w+)*\ . \w+([-.] \w+)*$`

Applications

- heavily used in web applications to verify, parse, manipulate, and format data

Applications

- heavily used in web applications to verify, parse, manipulate, and format data
- Search and replace text in a document

Applications

- heavily used in web applications to verify, parse, manipulate, and format data
- Search and replace text in a document
- Java and Oracle Database
 - `Pattern.compile("regex", Pattern.CASE_INSENSITIVE | Pattern.MULTILINE)`
 - `REGEXP_INSTR(email, '\w+@\w+(\.\w+)+') > 0`

Applications

- heavily used in web applications to verify, parse, manipulate, and format data
- Search and replace text in a document
- Java and Oracle Database
 - `Pattern.compile("regex", Pattern.CASE_INSENSITIVE | Pattern.MULTILINE)`
 - `REGEXP_INSTR(email, '\w+@\w+(\.\w+)+') > 0`
- Bioinformatics to assist with IDing DNA and protein sequences

Questions?

References

1. http://msdn.microsoft.com/en-us/library/ms972966.aspx#regexnet_topic9
2. http://docs.oracle.com/cd/B19306_01/appdev.102/b14251/adfns_regexp.htm#i1011021
3. http://www.gnu.org/software/emacs/manual/html_node/emacs/Regexp.html
4. <http://www.oracle.com/technetwork/database/focus-areas/application-development/twp-regular-expressions-133133.pdf>
5. <http://www.zytrax.com/tech/web/regex.htm>