REGULAR EXPRESSIONS

FRIEND OR FOE?

INTRODUCTION TO REGULAR EXPRESSIONS

INTRODUCTION

"In computing, a regular expression provides a concise and flexible means to "match" (specify and recognize) strings of text, such as particular characters, words, or patterns of characters."

- Wikipedia

"A regular expression is a set of pattern matching rules encoded in a string according to certain syntax rules."

- About.com

HISTORY

- Originated in the Unix world
- Many flavors
  - Perl, PCRE (PHP, Delphi), .NET, Java, JavaScript, Python, Ruby, Posix …

USAGE

- Testing (matching)
- Searching
- Replacing
- Splitting

LIMITATIONS

- Slow(ish)
- Can use lots of time and memory
- Unsuitable for some purposes
  - HTML parsing
- UTF-8

TOOLS

- Editors
- grep/egrep/fgrep
- Online tools
  - regex.larsolavtorvik.com
- RegexBuddy, RegexMagic
  - www.regexbuddy.com/regexmagic.html

DELPHI

- RegularExpressions, RegularExpressionsCore
  - Since XE
- TPerlRegex
  - Up to 2010
- PCRE flavor

EXAMPLE

- Search for "Handel", "Händel", and "Haendel"
  - `H(ä|ae?)ndel`
  - `Handel|Händel|Haendel`

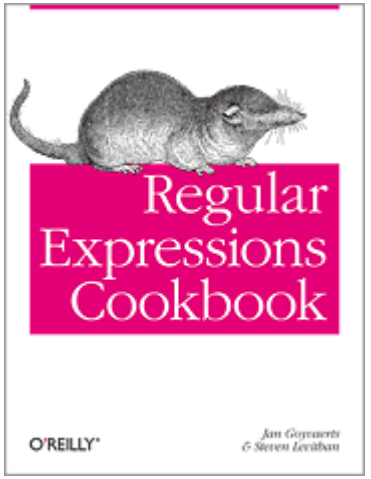
```
if TRegEx.IsMatch(s,'H(ä|ae?)ndel') then
```

SYNTAX

TUTORIALS

- www.regular-expressions.info/tutorial.html
- www.regular-expressions.info/delphi.html

- Jan Goyvaerts, Steven Levithan –
  Regular Expressions Cookbook (Amazon, O'Reilly)



LITERALS AND METACHARACTERS

- Metacharacters
  - `$()*+.?[\^{|`
- Literals
  - Everything else
- Escape
  - `\`
    - `\$ => $`
- Nonprintable
  - `\n, \r`

CHARACTER CLASS, ALTERNATIVES, ANY

- One-of
  - `[abc]`
  - `[a-fA-F0-9]`
  - `[^a-fA-F0-9]`
- Alternatives
  - `Delphi|Prism|FreePascal|Lazarus|SmartMS`
- Any
  - `.`

SHORTHANDS

- \d, \D
  - `[0-9]`, `[^0-9]`
- \w, \W
  - `[a-zA-Z0-9_]`, `[^a-zA-Z0-9_]`
- \s, \S
  - `[ \t\r\n]`, `[^ \t\r\n]`

ANCHORS

- Start of line/text
  - ^, \A
- End of line/text
  - $, \Z, \z
- Word boundary
  - \b, \B

UNICODE

- Single grapheme
  - `\X`
- Unicode codepoint
  - `\x{2122}` ™
- \p{category}
  - `\p{N}`
- \p{script}
  - `\p{Greek}`

GROUPS

- Capturing group
  - `(\d\d\d)`
- Noncapturing group
  - `(?:\d\d\d)`
- Named group
  - `(?P<digits>\d\d\d)`

GROUP REFERENCES

- Unnamed reference
  - `\1`, `\2`, … `\99`
- Named reference
  - `(?P=digits)`
- Example
  - `(\d\d\d)\1`

REPETITIONS

- Exact
  - `{42}`
- Range
  - `{17,42}`
  - `[a-fA-F0-9]{1,8}`
- Open range
  - `{17,}`

REPETITION SHORTCUTS

- `?`
  - `{0,1}`
- `+`
  - `{1,}`
- `*`
  - `{0,}`

REPETITION VARIATIONS

- Non-greedy
  - `*?`, `+?`
- Possesive
  - `*+`, `++`, `?+`, `{1,3}+`
  - `No backtracking`
  - `Allow regex to fail faster`

MODIFIERS

- Case-insensitive
  - `(?i)`, `(?-i)`
- Dot matches line breaks (‘single-line’)
  - `(?s)`, `(?-s)`
- ^ and $ match at line breaks (‘multi-line’)
  - `(?m)`, `(?-m)`

SEARCH & REPLACE

- `\1..\99` reference to a group
- `\0` all matched text
- `(?P=group)` reference to a named group
- `\`` left context
- `\'` right context

EXAMPLES

- Username
  - `[a-z0-9_-]{3,16}`
- Email (simplified)
  - `([a-z0-9_\.-]+)@([\da-z\.-]+)\.([a-z\.]{2,6})`
- IP (dotted, v4)
  - `([0-9]{1,3}\.){3}[0-9]{1,3}`

BAD EXAMPLES

- File name
  - (?i)^(?!^(PRN|AUX|CLOCK\$|NUL|CON|COM\d|LPT\d|\..*)(\..+)?$)
    [^\\\./:*\?\"<>\|][^\\/:*\?\"<>\|]{0,254}$
- Parsing HTML with RegEx
- Catastrophic backtracking
  - (x+x+)+y

DELPHI IDE

SEARCH & REPLACE

- Different flavor
- docwiki.embarcadero.com/RADStudio/XE7/en/Regular_Expressions
- Groups
  - {}
- ?, | are not supported

SEARCH & REPLACE DEMO

- Find {$IFDEF} and {$IFNDEF}
  - \$IFN*DEF
- Replace {$IFN?DEF WIN64} with {$IFN?DEF CPUX64}
  - \{\$IF(N*)DEF WIN64\}
  - \{$IF\1DEF CPUX64\}

CODE EXAMPLES

QUESTIONS?